An Efficient Collusion Resistant Security Mechanism for Heterogeneous Sensor Networks

Sajid Hussain
Jodrey School of Computer Science, Acadia University, Wolfville, NS, B4P 2R6, Canada
Firdous Kausar and Ashraf Masood
Department of Information Security, College of Signals, NUST, Rawalpindi, Pakistan
Jong Hyuk Park
Department of Computer Science and Engineering, Kyungnam University, Masan, Korea

Abstract
Purpose - Random key pre-distribution schemes are vulnerable to collusion attacks. The purpose of this research is to propose an efficient collusion resistant security mechanism for heterogeneous sensor networks (HSN). As large-scale homogeneous networks suffer from high costs of communication, computation, and storage requirements, the HSNs are preferred because they provide better performance and security solutions for scalable applications in dynamic environments.

Design/methodology/approach - We consider an heterogeneous sensor networks (HSN) consist of a small number of powerful high-end H-sensors and a large number of ordinary low-end L-sensors. However, homogeneous sensor networks (MSN) consists of only L-sensors. Since the collusion attack on key pre-distribution scheme mainly takes advantage of the globally applicable keys, which are selected from the same key pool, we update the key ring after initial deployment and generate new key rings by using one-way hash function on nodes’ IDs and initial key rings. Further, in the proposed scheme, every node is authenticated by the BS in order to join the network.

Findings - The analysis of the proposed scheme shows that even if a large number of nodes are compromised, an adversary can only exploit a small number of keys near the compromised nodes, while other keys in the network remain safe.

Originality/value - The proposed key management scheme outperforms the previous random key pre-distribution schemes by: a) considerably reducing the storage requirement, b) providing more resiliency against node capture and collusion attacks.

Keywords: Key distribution, Heterogeneous sensor networks, Collusion attack, Security, Symmetric key cryptography
1 Introduction

Wireless sensor networks are formed by a large number of sensor nodes. Each sensor node contains a battery-powered embedded processor and a radio, which enables the nodes to self-organize into a network, communicate with each other and exchange data over wireless links. Wireless sensor networks are ideal candidates for a wide range of applications, such as nuclear, biological and chemical attack detection and protection, home automation, battlefield surveillance and environmental monitoring (Akyildiz, et al. 2002).

An important area of research interest is a general architecture for wide-area sensor networks that seamlessly integrates homogeneous and heterogeneous sensor networks. Heterogeneous sensor networks have different types of sensors, with a large number of ordinary sensors in addition to a few powerful sensors. Further, as sensor devices are typically vulnerable to physical compromise and they have very limited power and processing resources, it is unacceptable to completely trust the results reported from sensor networks, which are deployed outside of controlled environments without proper security.

In order to provide secret communication in a sensor network, shared secret keys are used between communicating nodes to encrypt data. Traditionally, security is provided through public-key based protocols. However, these protocols require large memory, bandwidth and complex algorithms. The limited resources of WSNs make this type of security schemes unsuitable for implementation. Hence, asymmetric cryptography such as RSA or Elliptic Curve cryptography (ECC) is unsuitable for most sensor architectures due to high energy consumption and increased code storage requirements. Several alternative approaches have been developed to perform key management on resource constrained sensor networks without involving the use of asymmetric cryptography such as single network-wide key, pairwise key establishment, trusted base station, and random key pre-distribution schemes (Xiao, et al. 2007).

Key Management Schemes can also be classified into homogeneous or heterogeneous schemes with regard to the role of network nodes in the key management process. All nodes in a homogeneous scheme perform the same functionality; on the other hand, nodes in a heterogeneous scheme are assigned different roles. Homogeneous schemes generally assume a flat network model, while heterogeneous schemes are intended for both flat and clustered networks.

In random key pre-distribution (RKP) schemes, a large key pool of random symmetric keys is generated along with the key identifiers. All nodes are given a fixed number of keys randomly selected from a key pool. In order to determine whether or not a key is shared, each node broadcasts its keys’ identifiers. The, neighbors sharing a key associated with one of those identifiers, issue a challenge/response to the source. If two nodes do not share keys directly, they can establish a session key with the help of neighbors with which a key is already shared.
It is highly likely that all nodes in the network will share at least one key if the following are carefully considered: a) the network density, b) the size of the key pool, and c) the number of keys pre-configured in each sensor node.

While pre-distributing pairwise keys does protect confidentiality, it still loads nodes with a large number of globally-applicable secrets. By eliminating the eavesdropping attack, the pairwise scheme makes another type of malicious behavior more attractive. As several nodes possess the same keys, any node can make use of them by simply combining the keys obtained from a few nodes, which greatly increases the attacker’s chances of sharing keys with other nodes. A collusive attacker can share its pairwise keys between compromised nodes by enabling each node to present multiple ‘authenticated’ identities to neighboring nodes while escaping detection (Moore 2006). Colluding nodes can grow their knowledge about the network security measures. Therefore, it is conceivable that few compromised nodes can collude and reveal all the keys employed in the network to an adversary. Such scenario is considered as capturing the entire network since the adversary would be capable of revealing all encrypted communications in the network.

An adversary who obtains compromised nodes’ keys can inject malicious sensor nodes elsewhere in the network since the pool keys that were obtained are always valid and are used to authenticate each node. As a result, RKP is unable to protect the sensor network against collusion attack. In order to counter the collusion attacks, nodes should discard unused keys from the node’s memory after the initialization phase; however, it means that new nodes can no longer join the system after the initial network deployment. The other possible way to prevent collusion attacks is updating the preloaded keys in order to prevent the compromised and revoked nodes from launching a collusive attack in which they pool together their keys with the goal of jeopardizing the secure channels between other nodes. Without key updating, both the performance and security of the system will degrade greatly with the number of compromised nodes.

To address this issue, we present an efficient key management scheme based on random key pre-distribution for heterogeneous sensor networks. We focus to address the collusion problem in key management schemes that use RKP as an efficient means for key management. The goal of our key distribution approach is to update the initial key rings in such a way, that even though compromised nodes may collude and share their key rings, an adversary would not be able to access all the keys of the network.

A good security practice is to use different keys for different cryptographic operations; this prevents potential interactions between the operations that might introduce weaknesses in a security protocol. Therefore we are using different keys for encryption and authentication. The rest of the paper is organized as follows. Section 2 provides the related work and Section 3 describes the network and threat model. In Section 4, the proposed scheme is described in detail. Section 6 gives the results and performance evaluation. Finally, Section 7 concludes the paper.
2 Related Work

There are many key management protocols which are proposed for WSN.

The simplest method of key distribution is to pre-load a single network-wide key onto all nodes before deployment. After deployment, nodes establish communications with any neighboring nodes that also possess the shared network key. This method requires minimal memory storage because only a single cryptographic key is needed to be stored in memory. The main drawback of the network-wide key approach is that the compromise of a single node causes the compromise of the entire network, since the network-wide key is now known to the adversary. (S. Basagni and Bruschi 2001) use this approach to design a secure routing protocol. No new nodes are ever added to the system after deployment. In this case, the sensor nodes use the network wide key to encrypt unique link keys which are exchanged with each of their neighbors.

(S. Zhu and Jajodia 2003) follow this approach and set up all keys from a single network-wide key during a short, initial phase after deployment, assuming that no nodes are compromised during this phase, and later all nodes erase the single network key. This approach, however, is vulnerable to compromise of a single node that misses the key setup period, and does not erase its key.

Another common approach for key distribution uses a trusted, secure base station as an arbiter to provide link keys to sensor nodes, e.g., similar to Kerberos (Miller, et al. 1987, Kohl and Neuman 1993). The sensor nodes authenticate themselves to the base station, after which the base station generates a link key and sends it securely to both parties. An example of such a protocol is part of the SPINS (Perrig, et al. 2001), a security infrastructure specifically designed for sensor networks. In SPINS, each sensor node shares a secret key with the base station. Two sensor nodes cannot directly establish a secret key. However, they can use the base station as a trusted third party to set up the secret key. Small memory is required in this approach because for every node, a single secret symmetric key shared with the base station is needed, as well as one unique link key for each one of its neighbors. This approach is not scalable and has significant communication overhead. If any two nodes wish to establish a secure communications, they must first communicate directly with the base station. In a large network, the base station may be many hops away, thus incurring a significant cost in communication. The base station can become a target for compromise.

(Eschenauer and Gligor 2002) propose a probabilistic key pre-distribution technique to bootstrap the initial trust between sensor nodes. The main idea is to have each sensor randomly pick a set of keys from a key pool before deployment. Then, in order to establish a pairwise key, two sensor nodes only need to identify the common keys that they share. (Chan, et al. 2003a) further extended this idea and propose the q-composite key predistribution.
(Chan et al. 2003a) propose the q-composite key pre-distribution, which allows two sensors to setup a pairwise key only when they share at least q common keys. The q-composite keys scheme is a modification to the basic scheme (Eschenauer and Gligor 2002), where q common keys (q > 1) are needed, instead of just one. By increasing the amount of key overlap required for key-setup, the scheme increases the resilience of the network against node capture. (Chan et al. 2003a) also develop a random pairwise keys scheme and multipath key reinforcement to defeat node capture attacks.

(Zhu, et al. 2003) adopted the similar mechanism which uses threshold secret sharing for key reinforcement. S_A generates a secret key $K^t_{A,B}$, $j - 1$ random shares $sk_i$, and $sk_j = K^t_{A,B} \oplus sk_1 \oplus \ldots \oplus sk_{j-1}$. S_A sends the shares through j disjoint secure paths. S_B can recover $K^t_{A,B}$ upon receiving all shares.

(Pietro, et al. 2003) provide further improvements to basic scheme (Eschenauer and Gligor 2002). Keys are assigned to a node according to the output of a pseudorandom generator with a public seed and the node’s ID as inputs. (Liu and Ning 2003) propose the Polynomial Pool Based Key Predistribution Scheme that offers several efficient features the other key predistribution schemes lack, including: (1) any two sensors can definitely establish a pair-wise key when there are no compromised sensors (2) even with some nodes compromised, the others in the network can still establish pairwise keys (3) a node can find the common keys to determine whether or not it can establish a pairwise key and thereby help reduce communication overhead. Advantages of this scheme include that it allows the network to grow to a larger size after deployment. Disadvantages of this scheme include t-collision resistance (compromising more than t polynomials leads to network compromise).

(Oliveira, et al. 2006) show how random key predistribution, widely studied in the context of flat networks, can be used to secure communication in hierarchical (cluster-based) protocols such as LEACH (Heinzelman, et al. 2000). They presented SecLEACH, a protocol for securing node-to-node communication in LEACH-based networks. These and some others (Chan, et al. 2003b), (Zhu et al. 2003), (Pietro et al. 2003), (Cheng and Agrawal 2005), (Ren, et al. 2006) efforts have assumed a deployment of homogeneous nodes, and have therefore suggested a balanced distribution of random keys to each of the nodes to achieve security. Most of these schemes suffer from high communication and computation overhead, and/or high storage requirement.

(S. Zhu and Jajodia 2003) Localized Encryption and Authentication Protocol (LEAP) is a complete key management framework for static WSNs. For key deployment each node has to store 4 kinds of keys: (1) an individual key, (2) a group key, (3) cluster keys, and (4) pair-wise shared keys. In addition to these keys: a node also has to store a one-way key chain it creates, the commitments of the key chains its neighbors create, and the commitment of the base station’s key chain.
Blundo, et al. 1993 propose several schemes that allow any group of t parties to compute a common key, while being secure against collusion between some of them. These schemes focus on saving communication costs, while memory constraints are not placed on group members. When t = 2, one of these schemes is actually a special case of Blom’s scheme (Blom 1985).

Availability of some information on the sensors deployment in the field assists to improve the security of the key pre-distribution schemes. Some location aware schemes are proposed in (Liu and Ning 2003) and (Wadaa, et al. 2004). These techniques divide the target field into non-overlapping square areas and randomly deploy the sensors in every area. The exact location of a sensor in any area is unknown, but there is knowledge about the identity of sensors in every area. This information helps to eliminate the dependency of keys between nonadjacent cells.

(Du, et al. 2007) propose the asymmetric pre-distribution (AP) scheme for heterogeneous sensor networks. They consider a small number of powerful high-end sensors and a large number of ordinary low-end sensors. The basic idea of the AP key management scheme is to pre-load a large number of keys in each H-sensor whereas only a small number of keys are pre-loaded in each L-sensor, in order to provide better security with low complexity and significant reduction in storage requirement. (Traynor, et al. 2006) demonstrate that a probabilistic unbalanced distribution of keys throughout the network that leverages the existence of a small percentage of more capable sensor nodes can not only provide an equal level of security but also reduces the consequences of node compromise.

(Lu, et al. 2006) propose a framework for key management schemes in distributed wireless sensor networks with heterogeneous sensor nodes. (Kausar, et al. 2008) present a key management scheme for heterogeneous sensor networks. They reduce the storage requirements by incorporating a key generation process, where instead of generating a large pool of random keys, a key pool is represented by a small number of generation keys. For a given generation key and a publicly known seed value, a keyed-hash function generates a key chain; these key chains collectively make a key pool.

3 Network Model

We consider a heterogeneous sensor network (HSN) consisting of a small number of high end (H-node) sensors and a large number low end (L-node) sensors. L-nodes are ordinary sensor nodes with limited computation, communication, and storage capability. H-nodes, however, are more powerful nodes and have higher computation, communication, energy supply and storage capability than L-nodes. The HSN includes a base station (BS) that is globally trusted and it receives data from all the nodes; the BS has unlimited resources.
We consider the hierarchical structure, where H-nodes act as cluster heads (CHs) and L-nodes act as cluster members. Clustering of sensors enable local data processing, which reduces communication load in the network in order to provide scalable solutions.

Most traffic in HSN can be classified into one of three categories:

i. Many-to-one: Multiple H-nodes and L-nodes send sensor readings to a BS or aggregation point in the network.

ii. One-to-many: A single node (either a BS or H-nodes) multicasts or floods a query or control information to several L-nodes.

iii. Local communication: Neighboring L-nodes and H-nodes send localized messages to discover and coordinate with each other. A node may broadcasts messages intended to be received by all neighboring nodes or unicast messages intended for a only single neighbor.

3.1 Threat Model

Sensor networks are often deployed in hostile environments, yet nodes cannot afford expensive tamper-resistant hardware. The threat model is assumed to be an adversary that tries to capture and compromise a number of nodes in the network. Also, there is no unconditional trust on any sensor node. An adversary may try to eavesdrop on the messages exchanged in the system, intercept these messages as well as inject false messages. If an adversary compromises a node, the memory of that node is known to the adversary; CHs can also be compromised. The goal of the adversary is to uncover the keys used in the network for secure communication. The nodes can collude with each other by sharing their keys with other attacker nodes. The main objective of node collusion is to incrementally aggregate the uncovered keys of individual nodes to a level that allows revealing all encrypting traffic in the network.

3.2 Collusion Attack

In collusion attacks two or more nodes cooperate with each other by sharing their knowledge of pre-deployed secrets and thus increasing their capabilities in overcoming the network security measures. In RKP a collusion attack can be possible in the scenarios: a) when compromised nodes are in the transmission range of one another b) when compromised nodes are not in the transmission range of one another.
In the latter case, for example, consider two compromised nodes $n_1$ and $n_2$ which are not in the transmission range of each other. Suppose $n_1$’s neighbors are $n_3$, $n_5$, and $n_6$ and it shares key with $n_5$ and $n_9$. Similarly $n_2$’s neighbors are $n_4$, $n_7$, and $n_9$ and it shares keys with $n_3$, $n_6$, $n_7$ as shown in Table 1.

![Table 1: Collusion Attack](image)

Accordingly, $n_1$ can communicate securely with $n_5$ and $n_2$ can communicate securely with $n_7$. If $n_1$ colludes with $n_2$ the resultant keys known to both of them would be $\text{Keys}(n_1) \cup \text{Keys}(n_2)$. As a result, $n_1$ can communicate with $n_6$ and $n_3$ masquerading as $n_2$ and similarly $n_2$ can communicate with $n_9$ masquerading as $n_1$. It can be seen that compromised nodes not in the communication range of each other can collude to launch an attack to uncover a large number of employed keys.

**Definition 1**  A pseudo-random function is an efficient (deterministic) algorithm which given an $h$-bit seed, $y$, and an $h$-bit argument, $x$, returns an $h$-bit string, denoted $f_y(x)$, so that it is infeasible to distinguish the responses of $f_y$ for a uniformly chosen $y$, from the responses of a truly random function.

**Definition 2**  A cryptographically secure one-way hash function $H$ has the following property: for $y = H(k, x)$, 1) given $x$, it is computationally infeasible to find $y$ without knowing the value of $k$; 2) given $y$ and $k$, it is computationally infeasible to find $x$.

**Definition 3**  (Key Graph) Let $V$ represent all the nodes in the sensor network. A Key-Sharing graph $G(V, E)$ is constructed in the following manner:

For any two nodes $i$ and $j$ in $V$, there exists an edge between them if and only if nodes $i$ and $j$ have at least one common key in their key ring. Note that $|V| = n$ for a WSN of size $n$, the key graph $G(V; E)$ is connected if and only if any two nodes $i$ and $j$ belonging to $V$ can reach each other via edge set $E$ only.

For convenience, a summary of notations and symbols used in the paper are given in Table 2.
### Table 2 Symbol Definition

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS</td>
<td>Base Station</td>
</tr>
<tr>
<td>CH</td>
<td>Cluster Head</td>
</tr>
<tr>
<td>(iL_i)</td>
<td>Identity of L-node i</td>
</tr>
<tr>
<td>(iH_i)</td>
<td>Identity of H-node i</td>
</tr>
<tr>
<td>N</td>
<td>A random number string</td>
</tr>
<tr>
<td>(R_{L_i})</td>
<td>Set of the keys in L-node i initial key ring</td>
</tr>
<tr>
<td>(R_{H_i})</td>
<td>Set of the keys in H-node i initial key ring</td>
</tr>
<tr>
<td>(R'_{L_i})</td>
<td>Set of the keys in L-node i new/update key ring</td>
</tr>
<tr>
<td>(R'_{H_i})</td>
<td>Set of the keys in H-node i new/update key ring</td>
</tr>
<tr>
<td>(CK_i)</td>
<td>Cluster key of i-th cluster</td>
</tr>
<tr>
<td>(AK_X)</td>
<td>Authentication key of node X</td>
</tr>
<tr>
<td>(K_{X,Y})</td>
<td>A shared key between X and Y</td>
</tr>
<tr>
<td>(MAC_K (m))</td>
<td>A MAC of message m calculated using key K</td>
</tr>
<tr>
<td>(&lt;m&gt;_K)</td>
<td>An encryption of message m with key K</td>
</tr>
</tbody>
</table>

### 4 Protocol

In this section we present a key management algorithm that increases the network resilience to collusion.

#### 4.1 Initial Deployment

Generate a large key pool \(P\) consisting of \(S\) number of random symmetric keys and their ids prior to network deployment. Before deploying the nodes, each node is loaded with its assigned key ring \(R\) as follows: each L-node is pre-loaded with \(\gamma\) number of keys and each H-node is pre-loaded with \(\rho\) number of keys, randomly selected from the key pool without replacement, where \(\rho >> \gamma\). As given in (Pietro,
et al. 2006), the assigning rules are as follows:

4.1.1 L-node:

for every key $k_i \in P$, where $P = (k_1, k_2, ..., k_S)$
compute $z = f_{k_i}(id_{Lx})$

if $z \equiv 0 \text{mod}(\frac{S}{\gamma})$ then
put $k_i$ into $R_{Lx}$, the key ring of L-node.
In addition to that every L-node is pre-loaded with an authentication key $AK_{Lx}$ shared with BS and public key of BS.

4.1.2 H-node:

for every key $k_i \in P$, where $P = (k_1, k_2, ..., k_S)$
compute $z = f_{k_i}(id_{Hx})$

if $z \equiv 0 \text{mod}(\frac{S}{\rho})$ then
put $k_i$ into $R_{Hx}$, the key ring of H-node.
In addition to that every H-node is pre-loaded with an authentication key $AK_{Hx}$ shared with BS.

4.2 Cluster Heads Authentication

Before entering into the cluster organization phase each H-node need to be authenticated by BS. Let H-node $H_a$ sends a request to BS consisting of its id, a random number nonce, and MAC is calculated on all these values using it authentication key $AK_{H_a}$ as shown in message 1 of Figure 1. BS authenticates $H_a$ by verifying the MAC. If authentication is successful, BS randomly selects a key, suppose $k_m$ from the key ring of $H_a$. BS then sends message 2 shown in Figure 1 consisting of the id of $k_m$, nonce, and $id_{H_a}$ encrypting with $AK_{H_a}$. $H_a$ gets all these values by decrypting the message 2 and then generates the shared secret key $K_{BS,H_a}$ between BS and $H_a$ by applying one-way hash function on $id_{BS}$, $id_{H_a}$, and $0$ using $k_m$ as shown in message 3. We use different keys for date encryption and message authentication, therefore $H_a$ generate the MAC key $K'_{BS,H_a}$ by applying one-way hash function on $id_{BS}$, $id_{H_a}$, and $1$ using $k_m$ as shown in message 4. After joining the network, $H_a$ deletes $AK_{H_a}$ from its memory.
After authentication by BS, H-nodes enter into the cluster organization phase. Let H-node \( h_a \) broadcasts an advertisement message \( \text{adv} \), consisting of its id \( \text{id}_{h_a} \) and nonce as shown in message 1 of Figure 2. The nearby L-nodes, suppose \( l_b \) upon receiving the \( \text{adv} \) message, determines whether it shares a common key with \( h_a \) as follows: for every key \( k_j \in R_{l_b} \), \( l_b \) computes \( H(k_j, \text{id}_{h_a} | 0) \). If \( z \equiv 0 \mod(S) \), it means that \( h_a \) also has a key \( k_j \) in its key ring i.e. \( R_{h_a} \cap R_{l_b} = k_j \).

As \( l_b \) could receive \( \text{adv} \) broadcast messages from several H-nodes, it would be possible that \( l_b \) shares a common key with more than one H-node. From these H-nodes, it will choose the H-node as its CH with whom it has the best received signal strength and link quality.

\( l_b \) sends the join request to the selected CH (say \( h_a \)) protected by MAC, using \( k_j \) and include the nonce from CH broadcast (to prevent replay attack), as well as the id of shared key \( \text{id}_{k_j} \) chosen to protect this link (so that the receiving CH knows which key to use to verify the MAC) as shown in message 2 of Figure 2.

\( h_a \) upon receiving the join request from \( l_b \), authenticates the \( l_b \) by verifying the MAC using \( k_j \). \( h_a \) generate the shared pairwise key \( K_{h_a,l_b} \) with \( l_b \) by applying one-way hash function on \( \text{id}_{h_a}, \text{id}_{l_b}, \) and 0 using \( k_j \) as shown in message 3. \( h_a \) generate the MAC key \( K'_{h_a,l_b} \) with \( l_b \) by applying one-way hash function on \( \text{id}_{h_a}, \text{id}_{l_b} \) and 1 using \( k_j \) as shown in message 4 and sends message 5 to \( l_b \) consisting of cluster key \( \text{CK}_a \) encrypted with \( K'_{h_a,l_b} \) along with MAC on \( \text{id}_{h_a}, \text{id}_{l_b}, \) nonce, and \( \text{CK}_a \) calculated using \( K'_{h_a,l_b} \).
L_h gets the cluster key CK_a by decrypting the message using K_{H_a,L_h}, verifies the MAC by using K'_{H_a,L_h} to ensure the message authenticity and integrity and hence join the cluster.

Each L-node also records other H-nodes from which it receives the adv messages and it has common key with them, as these H-nodes will serve as backup cluster heads in case the CH (H_a) fails.

1: H_a \rightarrow *: id_{H_a}, nonce
2: L_b \rightarrow H_a : id_{L_b}, id_{H_a}, id_{k_j}, nonce, MAC_{L_b}(id_{L_b}, id_{H_a}, id_{k_j}, nonce)
3: K_{H_a,L_b} = H(k_j, id_{H_a} | id_{L_b} | 0)
4: K'_{H_a,L_b} = H(k_j, id_{H_a} | id_{L_b} | 1)
5: H_a \rightarrow L_b : CK_a > K_{H_a,L_b} \times MAC_{K_{H_a,L_b}}(id_{H_a}, id_{L_b}, nonce, CK_a)

Fig. 2. Messages Transferred between sensor nodes and CHs.

4.3.1 Unsupervised Nodes

At the end of cluster organization phase, it is expected that a fraction of the L-nodes will not be matched with a CH because of key sharing constraints; these nodes are called unsupervised nodes. Suppose the unsupervised node L_x have best RSSI with H_a. L_x sends request to H_a consisting of its id, id of H_a, nonce and MAC is calculated on all these values using AK_{L_x} shown in message 1 of Figure 8. H_a forwards this message to BS. BS authenticate the L_x by verifying the MAC and select key k_j from the key ring of L_x. BS sends k_j and id_{k_j} to H_a encrypting with the key K_{BS,H_a} along with MAC on k_j, id_{k_j}, and nonce using key K'_{BS,H_a} as shown in message 3. H_a generates the shared pairwise key with L_x by applying one-way hash function on id_{H_a}, id_{L_x}, and 0 by using k_j as shown in message 4. H_a generates the MAC key with L_x by applying one-way hash function on id_{H_a}, id_{L_x}, and 1 by using k_j as shown in message 5. H_a sends message 6 to L_x consisting of its id_{H_a}, id_{L_x}, id of the key k_j to be used as common shared key, and cluster key encrypted with K_{H_a,L_x} and MAC on all these values using K'_{H_a,L_x}. L_x receives this message and calculate the K_{H_a,L_x} and K'_{H_a,L_x} by using k_j and use it to get cluster key and hence join the network.
1: \(L_x \rightarrow H_a: id_{L_x}, id_{H_a}, nonce, MAC_{AK_{L_x}}(id_{L_x}, id_{H_a}, nonce)\)
2: \(H_a \rightarrow BS: Forward \text{ Message 1 to BS}\)
3: \(BS \rightarrow H_a: id_{k_j}, < k_j >_{k_{BS,L_x}}\)
4: \(K_{id_{L_x}, L_x} = H(k_j, id_{H_a} \mid id_{L_x} \mid 0)\)
5: \(K'_{id_{L_x}, L_x} = H(k_j, id_{H_a} \mid id_{L_x} \mid 1)\)
6: \(H_a \rightarrow L_x: id_{k_j}, id_{L_x}, nonce, < CK_{L_x} >_{k_{L_x,L_x}}\)
\[\text{MAC}_{k_{L_x,L_x}}(id_{L_x} \mid id_{H_a} \mid id_{k_j} \mid nonce \mid CK_{L_x})\]

Fig. 3. Unsupervised nodes key establishment

There may be some L-nodes, suppose L_y in the network that may have common key shared with a CH H_a but have better RSSI with H_b then H_a. However, L_y do not have common key shared with H_b. In that case L_y can contact with H_b as unsupervised node to request key as explained above.

4.3.2 Direct Key Discovery Phase

After cluster organization phase, L-nodes learn their neighbors through the exchange of hello messages, and then attempt to establish keys with them. To accomplish this, L-nodes broadcast hello messages.

Consider an L-node, L_a, it broadcasts a hello message consisting of its id id_{L_a}. Then, it waits for hello messages from its neighboring L-nodes. Suppose, it receive hello message from one of its neighbor L_b, it extracts the node id from message i.e. id_{L_b}. For every key \(k_j \in R_{L_a}L_a\) computes \(z = f_{k_j}(id_{L_b})\). If \(z \equiv 0 \mod(\gamma)\), it means that node L_b also has a key \(k_j\) in its key ring i.e. \(R_{L_a} \cap R_{L_b} = k_j\). After discovering the common key in their key rings, they will generate the shared pairwise key by applying one-way hash function on id_{L_a} and id_{L_b} by using \(k_j\).

\[K_{L_a, L_b} = H(k_j, id_{L_a} \mid id_{L_b} \mid 0)\]

If L_a and L_b share more than one common keys in their key rings, the key with the least id would be used to generate the shared pairwise key.

4.3.3 Indirect Key Discovery Phase

L-nodes gather information about both types of neighbors: 1) nodes with which they share a key, and 2) nodes with which they do not share keys. When the direct key
discovery phase ends, the nodes would have discovered the common keys, if any, with their neighbors. L-nodes use the CH with which keys are already shared to assist it in establishing secure connections with the neighboring L-nodes with which common keys are not found.

Let L-nodes $L_x$ and $L_y$ are neighboring nodes in the same cluster; however, they do not share a common key in their key rings, $R_{L_x} \cap R_{L_y} = \emptyset$. The L-node $L_x$, having already established a link with its CH ($H_a$), transmits a message to $H_a$, as shown in Figure 4, requesting to transmit a key with L-node $L_y$ encrypted with key $K_{H_a,L_x}$.

$H_a$ generates a key $k_i$ and unicasts the message 2 to $L_x$ and message 3 to $L_y$ shown in Figure 4. When $L_x$ (or $L_y$) receives its message from $H_a$, it decrypts the message using key $K_{H_a,L_x}$ to get key $k_i$. Similarly, $L_y$ uses key $K_{H_a,L_y}$ for decrypting the message. Now, $L_x$ and $L_y$ generate the shared pairwise by applying one-way hash function on $\text{id}_{L_x}$ and $\text{id}_{L_y}$ by using $k_i$, as shown in message 4.

1: $L_x \rightarrow H_a : \text{id}_{L_1}, \text{id}_{L_2}, \text{nonce}, \text{MAC}_{K_{H_a,L_x}}(\text{id}_{L_1} \mid \text{id}_{L_2} \mid \text{nonce})$
2: $H_a \rightarrow L_x : \text{id}_{L_1}, \text{id}_{L_2}, \text{nonce}, < k_i >_{K_{H_a,L_x}}$
3: $H_a \rightarrow L_y : \text{id}_{L_1}, \text{id}_{L_2}, \text{nonce}, < k_i >_{K_{H_a,L_y}}$
4: $K_{L_x,L_y} = H(k_i, \text{id}_{L_x} \mid \text{id}_{L_y})$

Fig. 4. Indirect key discovery phase.

4.4 Key Ring Update

After indirect key-discovery phase, all L-nodes and H-nodes destroy their initial key rings. Because these key rings have globally applicable secrets which can be used by adversary to launch a collusion attack, we delete these initial key rings.

First, before a node (say $L_x$) destroys its initial key ring, it generates a new key ring as shown in Figure 5. For every key $k_i \in R_{L_x}$, it generates a new key $k_i'$ by applying one-way hash function on its id ($\text{id}_{L_x}$) and $k_i$. In this way, it generates a set of new keys from keys in its initial key ring. Further, in order to keep record of the keys in its initial key ring, these newly generated keys are assigned the same ids as those of the original keys. Then, $L_x$ deletes $k_i$ from its key ring $R_{L_x}$. 
procedure keyRingUpdate()
1: for ∀kᵢ ∈ Rₜₓ do
2: kᵢ' = H(kᵢ, idₜₓ)
3: idₜᵢ' = idₜᵢ
4: delete(kᵢ)
5: end for

Fig. 5. Key Ring Update

Further, the above procedure is also applied for H-nodes to update their key rings.

5 Other Security Issues in HSN

In this Section, we discuss other security issues in HSN, including setting up keys for newly deployed sensor nodes, node revocation, and periodic re-keying.

5.1 Addition of a New Node

The common key pre-distribution schemes are unable to add new nodes in the network if the initial key rings are deleted from node’s memory. As a result, we develop a new solution capable of handling addition of new legitimate L-nodes beyond the initial deployment, even after the deletion of initial key rings from node’s memory.

Suppose new L-node Lₓ wants to join a network, it broadcasts a join request consisting of its id (idₜₓ) and a random number nonce, as shown in message 1 of Figure 6. Then, it waits for reply from nearby CHs. Let Lₓ receives a reply message from CH (say Hₕ). For every key kᵢ ∈ Rₜₓ, Lₓ computes \( z = fₜₙₜ \text{id}_ₜₙ \). If for any kᵢ, \( z \equiv 0 \mod p \), it means that kᵢ ∈ Rₜₙ, but it is no longer available now because Rₜₙ has been deleted. So, Lₓ computes the corresponding key i.e. kᵢ' of Hₕ’s new key ring Rₜₙ', by applying one-way hash function on idₜₙ and kᵢ i.e. kᵢ' = H(kᵢ, idₜₙ). Then, Lₓ sends a message to Hₕ consisting of its id idₜₓ, id of kᵢ (idₜᵢ = idₜᵢ'), nonce and MAC is calculated on all these values using kᵢ' as shown in message 3 of Figure 6. Now, Lₓ and Hₕ generate the shared pairwise key by applying one-way hash function on idₜₙ, idₜₓ and 0 by using kᵢ', as shown in message 4.
Both $L_x$ and $H_a$ generate MAC key by applying one-way hash function on $id_{H_a}$, $id_{L_x}$ and 1 by using $k_j'$, as shown in message 5.

1: $L_x \rightarrow \ast : id_{L_x}, \text{nonce}$
2: $H_a \rightarrow L_x : id_{H_a}, \text{nonce}$
3: $L_x \rightarrow H_a : id_{L_x}, MAC_{k_j'}(id_{L_x} | id_{L_x} | \text{nonce})$
4: $K_{L_x,H_a} = H(k_j', id_{H_a} | id_{L_x} | 0)$
5: $K_{L_x,H_a}' = H(k_j', id_{H_a} | id_{L_x} | 1)$

Fig. 6. New node addition

Then, $L_x$ discovers the shared key with its neighboring L-nodes by using either direct or indirect key discovery phase, as given above.

5.2 Node Revocation

In the proposed scheme there is no need to revoke the key rings of compromised nodes because the initial key rings of all the nodes in the network has been updated and no two nodes in the network has any key common in their key rings after initial deployments. If a node is compromised only the links those are directly associated with that node will be compromise. Therefore, our scheme does not need the revocation of key rings of compromised nodes.

5.3 Fault Tolerance

Our approach should support the ability to allow L-nodes to change the cluster even after the initial key rings has been updated. As the above described scheme will not allow the nodes to change cluster once their initial key ring has been updated. As a result, it is imperative that we develop a new solution capable of handling the change in network topology beyond the initial deployment. Suppose an L-node $L_x$ moves from cluster $a$ to cluster $b$. So it needs to find a cluster key $CK_{b_a}$ shared pairwise key with CH $H_b$ and also the shared pairwise keys with its new neighboring L-nodes. As $L_x$ has updated its initial key ring, it will not have any common key with $H_b$ or any of neighboring L-nodes.
There are three problems that we need to solve: (1) How can $H_b$ who no longer has the initial key rings authenticate $L_x$? (2) How can $H_b$ and $L_x$ setup a pairwise key between each other? (3) How can $L_x$ setup a pairwise key with its neighboring $L$-nodes? In that case $L_x$ contacts the BS for joining the $H_b$ as CH, as described above in section 4.3.1 $L_x$ will setup the pairwise key with neighboring $L$-nodes by using the indirect key discovery phase described in section 4.3.3.

### 5.4 Periodic Re-keying

Periodic re-keying has to be performed if any node finishes $2^{2k/3}$ number of encryptions using the same key, where $k$ is the number of bits in the key. The cluster key re-keying is initiated by CH by generating the new cluster key, encrypting it with the old cluster key and distributing to the cluster members.

Re-keying of cluster key is also necessary, when a cluster member leaves the cluster because of either its battery power gets exhausted or when it is being compromised by an adversary. In that case CH needs to distribute the new cluster key by unicast it to $L$-nodes encrypting with the shared pair-wise keys so the nodes that have leave cluster do not receive new cluster key.

### 6 Performance Analysis

This section analyzes the proposed scheme and explains its features that make this scheme feasible to implement and a better alternative option as compared to the other key pre-distribution schemes.

For any pair of nodes to find a secret key between them, the key sharing graph $G(V, E)$ needs to be connected. Given the size and the density of a network, the objective is to determine the key pool size $S$, the number of keys assigned to $L$-nodes $\gamma$, and the number of keys assigned to $H$-nodes $\rho$ such that, the graph $G$ is connected with high probability. Chan et al. (Chan et al. 2003a) propose the q-composite keys scheme that allows two sensors to setup a pairwise key only when they share at least $q$ common keys. The q-composite keys scheme provides better security for sensor networks. The key management schemes proposed in this paper can be easily extended to require at least $q$ shared keys. We want to find the largest key pool size such that the probability of an $L$-node and an $H$-node sharing at least $q$ keys is no less than a threshold $p$.

Let $p(j)$ be the probability that an $L$-node and an $H$-node have exactly $j$ keys in common. Recall that an $L$-node and an $H$-node are pre-loaded with $\gamma$ and $\rho$ keys,
respectively.

An L-node has \( \binom{S}{\gamma} \) different ways of picking \( \gamma \) keys from a key pool with the size \( S \), and an H-node has \( \binom{S}{\rho} \) different ways of picking \( \rho \) keys from the key pool. Thus, the total number of ways for an L-node and an H-node to pick \( \gamma \) and \( \rho \) keys, respectively, is \( \binom{S}{\gamma} \binom{S}{\rho} \). Suppose that the two nodes have \( j \) keys in common. There are \( \binom{S}{j} \) ways to pick \( j \) common keys.

After the \( j \) common keys are picked, there remain \( \rho + \gamma - 2j \), distinct keys in the two key rings which are to be picked from the remaining pool of \( S-j \) keys. The number of ways to do so is \( \binom{S-j}{\rho + \gamma - 2j} \). The \( \rho + \gamma - 2j \) distinct keys must then be partitioned between the L-node and the H-node. The number of such partitions is \( \binom{\rho + \gamma - 2j}{\gamma - j} \).

Hence the total number of ways to choose two key rings with \( j \) keys in common is the product of the three terms, i.e., \( \binom{S}{j} \binom{S-j}{\rho + \gamma - 2j} \binom{\rho + \gamma - 2j}{\gamma - j} \). Thus the probability of sharing at least \( j \) keys in common is given in Eq. 1

\[
p(j) = \frac{\binom{S}{j} \binom{S-j}{\rho + \gamma - 2j} \binom{\rho + \gamma - 2j}{\gamma - j}}{\binom{S}{\gamma} \binom{S}{\rho}}
\]

Let \( p_c \) be the probability that an L-node and an H-node share sufficient keys to form a secure connection. If \( q \) shared-keys are required, we have: \( p_c = 1 - (p(0) + p(1) + \cdots + p(q-1)) \). For given key ring size \( \gamma \) and \( \rho \), key overlap \( q \), and minimum connection probability \( p \), the largest key pool size \( S \) can be computed such that \( p_c \geq p \).

The probability of an L-node and H-node with key rings sizes \( \gamma \) and \( \rho \) sharing at least one key with each other is given in Eq. 2:
Similarly the probability of sharing at least one key between two L-node is given in Eq. 3

\[
p_{sk} = 1 - \frac{\left(\begin{array}{c} S \\ \gamma \end{array}\right) \left(\begin{array}{c} S - \gamma \\ \rho \end{array}\right)}{S^2} = 1 - \frac{(S-\gamma)!(S-\rho)!}{S!(S-\gamma-\rho)!}
\]  
(2)

Similarly the probability of sharing at least one key between two L-node is given in Eq. 3

\[
p_{sk} = 1 - \frac{\left(\begin{array}{c} S \\ \gamma \end{array}\right) \left(\begin{array}{c} S - \gamma \\ \rho \end{array}\right)}{S^2} = 1 - \frac{(S-\gamma)!^2}{S!(S-2\gamma)!}
\]  
(3)

Figure 7 shows the probability of key sharing among H-node and L-node with respect to key pool size. Further, a fixed number of pre-loaded keys are used in H-nodes, \( \rho = 500 \); whereas pre-loaded keys for L-nodes vary as \( \gamma = 10, 20, 30 \). The graphs show that the pre-loaded keys in L-nodes can be significantly reduced with acceptable probability of key sharing.

![Fig. 7. The Key Sharing Probability](image-url)
In our scheme, only a fraction of CHs is probabilistically accessible by an ordinary node. Probability of key sharing between H-node and L-node and the number of CHs $\beta$ in the network can also determine the expected number of unsupervised nodes, i.e. the probability that an ordinary node will be unsupervised. Given $p$ and $\beta$, the probability of the number of unsupervised nodes is given in Eq. 4:

$$p_{\text{us}} = \left(1 - \left(1 - \left(\frac{(S-\gamma)!(S-\rho)!}{S!(S-\gamma-\rho)!}\right)^{\beta}\right)\right)$$

(4)

In a network with $N$ number L-nodes, it is then expected that $N \times p_{\text{us}}$ nodes will be unsupervised. Figure 8 shows fraction of unsupervised nodes as function of $\beta$ under different value of $p$. As $\beta$ increases, the number of unsupervised nodes decreases rapidly. Further, as $p$ increases, the number of unsupervised nodes also increases.

Fig. 8. Unsupervised Nodes

6.1 Security Analysis

We evaluate our key pre-distribution scheme in terms of its resilience against node capture and collusion attack. We would like to investigate when $\alpha$ number of nodes
are captured, what fraction of the additional communication (i.e. communication among uncaptured nodes) would be compromised?

To compute this fraction, we first compute the probabilities that any one of the additional communication links is compromised after \( \alpha \) node are captured. In our analysis, we are considering the links which are secured using a pairwise key computed from the common key shared by the two nodes of this link. We should also notice that during shared key discovery process, two neighboring nodes find the common key in their key rings and use this key to agree upon another random key to secure their communication. Because this new key is generated by applying one-way hash function on common shared key and node ids, the security of this new random key does not directly depend on whether the key rings are broken. Further, the nodes’ initial key rings are also deleted from their memory, after setting up shared pairwise keys with neighbors. As a result, the fraction of communications compromised when \( \alpha \) number of nodes being compromised can be given as

\[
\frac{\text{number of links in } \alpha \text{ compromised nodes}}{\text{Total number of links in the network}}
\]

which means that only those links will be affected which are directly connected with \( \alpha \) compromised nodes, while the other links in the network will remain safe. Figure 9 shows the graphs of number of compromised communication links with respect to the number of compromised nodes. We compare our proposed scheme (PS) with basic scheme (EG) (Eschenauer & Gligor 2002) and q-composite scheme (Chan et al. 2003a). The graphs show that as the number of compromised nodes increases, the traditional schemes are severely affected as compared to PS.

Further, in collusion attacks, the adversary takes advantage of the pairwise secret keys stored by each sensor node as these keys are globally applicable secrets and can be used throughout the network, yet ordinary sensors can only communicate with the small fraction of nodes within radio range. So, the adversary can launch a collusion attack by exploiting this lack of communication between nodes and can now share its pairwise keys between compromised nodes, enabling each node to present multiple ‘authenticated’ identities to neighboring nodes, while escaping detection. In proposed scheme, we delete the initial key rings from nodes memory after setting up
shared pairwise keys with neighbors. However, nodes generate new key rings from initial key rings by applying one-way hash function on node ids and keys in their initial key rings.

Consider two arbitrary L-nodes, $L_a$ and $L_b$, where $R_{L_a} = \{k_1, k_2, ..., k_l\}$, $R_{L_b} = \{k_1, k_2, ..., k_l\}$, and $R_{L_a} \cap R_{L_b} = k_i$. As $L_a$ and $L_b$ are not within the communication range of each other, they do not use $k_i$. After setting up shared pairwise keys with neighbors, both $L_a$ and $L_b$ delete the initial key rings ($R_{L_a}$ and $R_{L_b}$) and generate the new key rings (say $R'_{L_a}$ and $R'_{L_b}$) by applying one-way hash function on all the keys in their initial key rings and node ids. As a result, $R'_{L_a} \cap R'_{L_b} = \emptyset$. Similarly, in $\alpha$ number of compromised nodes, there will be no common key in their new key rings i.e $R'_{L_1} \cap R'_{L_2} \cap ...... \cap R'_{L_\alpha} = \emptyset$. As no more globally applicable secrets remain in the node’s memory, it is not possible by an adversary to launch a collusion attack.

7 Conclusion

Key establishment is a fundamental prerequisite for secure communication in wireless sensor networks. A key pre-distribution scheme is one of the common solutions for establishing secure communication in sensor networks. Random key pre-distribution schemes are vulnerable to collusion attacks because preloading global secrets onto exposed devices can be used in these attacks. This work presents a
new efficient key distribution scheme for heterogeneous sensor networks which is secure against collusion attack. The analysis shows that the proposed scheme provide more resiliency against node capture and collusion attack by deleting the initial key rings from their memory after generating the shared pairwise key with neighbors. It also allow new nodes to join the system once initialization is completed and initial key ring has been destroyed from node’s memory.

References


L. Eschenauer and V. D. Gligor (2002). ‘A Key Management Scheme For Distributed Sensor Networks’. In ACM CCS.


F. Kausar, et al. (2008). ‘Scalable and efficient key management for heterogeneous


**Firdous Kausar** is a PhD candidate in Department of Information Security, College of Signals, National University of Science and Technology, Pakistan. She has authored more than 10 papers including a book chapter and journal papers. Her research interests are in network and data security, key management and authentication protocols, sensor networks, mobile and ad hoc networks. She is a member of IEEE and IACR (International Association of Cryptologic Research).

**Dr. Sajid Hussain** is an Assistant Professor in the Jodrey School of Computer Science, Acadia University, Canada. He received Ph.D. in Electrical Engineering from the University of Manitoba, Canada. Dr. Hussain is investigating sensor networks, communication protocols, security, database management, intelligent techniques and architectures for ubiquitous and pervasive applications such as smart homes and petroleum offshore facilities. He has published more than 35 refereed journal, conference and workshop papers. The research is financially supported by several grants and contracts, such as Natural Sciences and Engineering Research Council (NSERC) Canada, National Research Council (NRC) Canada, Atlantic Innovation Fund (AIF), and Nova Scotia Health Research Foundation (NSHRF). Dr. Hussain has co-organized several International workshops and conferences, served on many technical program committees, and reviewed papers for several journals, conferences and workshops. Further, he has reviewed grant proposals for NSERCs Discovery Grants, Strategic Project Grants (SPG), and Research Tools and Instrument (RTI) Grants. He is a member of IEEE and ACM societies.

**Dr. Jong Hyuk Park** received his Ph.D. degree in Graduate School of Information Security from Korea University, Korea. Before August, 2007, Dr. Park had been a research scientist of R and D Institute, Hanwha Sand C Co., Ltd., Korea. He is now a professor at the Department of Computer Science and Engineering, Kyungnam University, Korea. Dr. Park has published many research papers in international journals and conferences. Dr. Park has been served as Chairs, program committee or organizing committee chair for many international conferences and workshops; Chair of SH06/07/08, MUE07/08, IPC07/07, FGCN07/08, TRUST’07/08, SMPE07/08, UASS’07/08, SSDU’07/08, UIC07/08, Mobility08, SUTC08, WPS08, SecUbiq’07, ISM’07, and PC member of PerCom’08, AINA’07/08, MOBIQUITOUS07/08, ATC08, EUC07 and so on. Dr. Park is the founder of International Conference on International Conference on Multimedia and Ubiquitous
Dr. Park is editor-in-chief of the International Journal of Multimedia and Ubiquitous Engineering (IJMUE), the managing editor of the International Journal of Smart Home (IJSH), and Associate Editor of Security and Communication Networks (SCN). In addition, he has been served as a Guest Editor for international journals by some publishers: Oxford, Emerald, Hindawi, Springer, Elsevier, Inder science, SERSC. Moreover, he is a member of the Task Force in the IEEE IUC. Dr. Park’s research interests include Digital Forensics, Security, Ubiquitous and Pervasive Computing, Context Awareness, Multimedia Service, etc. He is a member of the KICS, KIISC, KMS, and IEICE.

Dr. Ashraf Masood is a Head of Research and Development Establishment, College of Signals, National University of Science and Technology, Pakistan. He received Ph.D. in Electrical Engineering from the Michigan State University, USA. Dr. Masood has published many research papers in international journals and conferences. He has been served as Chairs, program committee or organizing committee chair for many international conferences and workshops. His research interests are in cryptography, network security, sensor networks, cryptanalysis, stream ciphers, algebraic attack. He is a member of the IEEE.