2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks

A Key Management Scheme for Heterogeneous Sensor Networks Using Keyed-Hash Chain

Biming Tian
DEBI Institute
Curtin University of Technology
Perth, Australia
Email: Biming.Tian@postgrad.curtin.edu.au

Song Han, Tharam Dillon
DEBI Institute
Curtin University of Technology
Perth, Australia
Email: {Song.Han, Tharam.Dillon}@cbs.curtin.edu.au

***Abstract*—We present a suite of key management scheme for heterogeneous sensor networks. In view of different types of communications, a single key can not satisfy various communication requirements. It is necessary to study the establishment and renewal of different types of keys in heterogeneous sensor networks. In this paper, we propose a new key management scheme which can support five types of communications. Our basic scheme is based on a keyed-hash chain approach. A new cluster mechanism is used to improve the probability of key sharing between sensors and their cluster heads. Different from existing schemes where a node capture attack might lead to the disclosure of several key chains, our method can avoid this drawback through not storing network-wide generating keys in low-cost sensors. Only pairwise keys involving the compromised node should be deleted in our scheme. It is motivated by the observation that all the information stored on a sensor may be disclosed once the sensor gets compromised. Through the analysis of both security and performance, we show the scheme meets the security requirements.**

***Keywords*-key management, key predistribution, keyed-hash function; heterogeneous sensor network.**

I. INTRODUCTION

Currently Wireless Sensor Networks (WSNs) are being deployed for wide applications ranging from civilian to military use. A typical WSN is composed of a great number of sensor nodes. These sensor nodes have limited battery power, weak data processing capability and short radio range. Most important, sensor nodes are often randomly spread out over specific regions and work in unattended environment. They are prone to all kinds of attacks thus security becomes the first concern. In order to keep communication secure, sensitive data should be encrypted and authenticated. Therefore, key management, which is a prerequisite of encryption and authentication, should be addressed carefully.

Many previous researches on sensor networks considered a homogeneous topology. In a homogeneous network, all the nodes are identical and organized in a flat model. Such a network is simple and efficient for small network scales. However, it lacks of scalability. In such a network, the sensor node which performs data aggregation and the forwarding function will run out of power in advance because it receives higher traffic volumes. It in turn leads to the collapse of the whole network. An alternative way to extend the network life is to randomly and periodically rotate the responsibility of the data aggregation function over all nodes as the scheme proposed in [1]. However, this rotation raises in the requirement for stronger hardware capabilities. Both [2] and [3] demonstrated the performance bottleneck of homogeneous networks.

Lu et al. in [4] consider a multi-hierarchy Heterogeneous Sensor Network (HSN). The network consist of $I(I \geq 3)$ types of nodes. In fact, the common HSNs include 3 types of entities. They are low-end nodes, high-end nodes and the Base Station (BS). Low-end nodes have tiny memory and very limited data processing capability. While high-end nodes have more storage and stronger data processing capability. BS is most powerful. These three types of entities are organized in a hierarchical structure. High-end nodes act as cluster heads and sensor nodes as cluster members. High-end nodes aggregate data and forward the aggregated data to BS and help BS in managing the network. Node addition and revocation operations can be performed within a cluster. In this way, the influence of a compromised node can be localized within a cluster rather than the whole network. Therefore, HSNs provide scalability and security benefits.

Heterogeneous Sensor Networks are liable to be attacked by outside adversaries or inside compromised sensor nodes [5]. Security is a challenging problem in HSNs due to the constraints on memory and energy at low-end nodes. Asymmetric cryptographic algorithms, such as RSA public key techniques, while applicable for wired networks even general ad hoc networks, may not suitable for sensor networks. It is well known that Elliptic Curve Cryptography (ECC) can obtain the same security level as RSA with a shorter key length. A 160-bit ECC key has the same security level as a 1024-bit RSA key [6]. However, the research in [7] demonstrated that the Diffie-Hellman key agreement process using Elliptic Curve asymmetric key algorithm in an ad hoc network is between one to two orders of magnitude larger than the key exchange process based on the AES symmetric key algorithm in a regular non ad hoc network. Therefore, we approach the problem using symmetric key cryptography.

978-0-7695-3935-5/09 $26.00 © 2009 IEEE
DOI 10.1109/MSN.2009.26

In this paper, we focus on the design of a secure and efficient key management scheme for HSNs. Our scheme supports the establishment and renewal of five types of keys in the network because single key cannot satisfy different communication requirements. All keys involved in this paper are symmetric keys. The scheme uses keyed-hash chain to reduce storage overhead. Different from the existing keyed-hash chain based schemes [13] [14], sensor nodes only store commitments for the corresponding key chains. This scheme is resilient to node capture attack in this way. Even if one or more nodes are compromised, no secret information is disclosed. In addition, we define a new cluster mechanism which improves the key sharing probability between the sensors and their cluster heads.

The rest of the paper is organized as follows: In Section II, we introduce the related research work. In Section III, we describe the HSN model that is assumed throughout the paper. In Section IV, we give the full details of the scheme. After that, we study the establishment of other types of keys in Section V. We analyze the security and performance of the proposed scheme in Section VI and VII respectively. Finally, we summarize the work in Section VIII.

II. Related Work

As applications gain more ground, security issues in sensor networks have created more concern. There have been several attempts in securing WSNs, while all of them have their pros and cons and they do not give a satisfactory performance. For reasons of space, it is not possible to list all of the existing work. Instead, we only review several representative key predistribution techniques. After that, we will present the outlines of our new scheme which would be more effective.

Briefly, the previous schemes can be classified into three categories: random key predistribution schemes [8], polynomial-key predistribution schemes [10], and location-based key predistribution schemes [11].

Eschenauer et al. proposed the pioneering work [8]. It is a random key predistribution scheme. In this solution, key distribution is divided into three stages: key predistribution, shared-key discovery, and path-key establishment. Initially a large key pool of P symmetric keys and the keys' identities are generated. k keys are randomly drawn and loaded into each sensor node. In the second stage, two neighboring nodes can set up a pairwise key if they have at least a key match. It may be the case that some of the neighboring nodes may not be able to find a key in common. The key-path establishment is designed for this case. The nodes still can establish a secure channel under the help of one or more intermediate nodes. This scheme is flexible and fairly easy to employ. However, a large fraction of communications may be compromised when just a few nodes are compromised.

The basic scheme in [8] has further been improved by the schemes in [9] [10] [11] from different aspects. Chan et al. in [9] presented two variations of the scheme in [8]: $q(q \geq 2)$-composite scheme and multi-path key reinforcement scheme. The major difference of [8] and q-composite scheme is that q common keys instead of a single key are needed to establish a pairwise key. By increasing the amount of key overlap required for pairwise key establishment, the scheme makes it difficult for an adversary to compromise a node. However, it is pity that q-composite scheme does not satisfy scalability requirements. Multi-path key reinforcement scheme deals with the problem of key updating. A node will generate j random key update shares and send them through j disjoint secure paths. The receiving nodes can then generate a reinforced link key. This approach increases resilience but requires increased CPU use and power consumption.

Liu et al. [10] adopted the same idea as that in [8] but extended the key pool to a t-degree polynomial pool. In this scheme, no key in the network is used twice. By using the polynomial, this scheme can be resilient up to t colluding compromised nodes. Even if some sensors get compromised, there will still be a high probability of establishing pairwise keys for uncompromised sensors. Du et al. [11] described a random key predistribution scheme which combines deployment knowledge to the key predistribution scheme in [10]. The authors argue that only neighboring nodes need secure channels to communicate with each other. The scheme takes advantage of the location information to avoid unnecessary key establishment. Also, by choosing keys shared with nodes likely to be in close proximity, the scheme increases the probability of key sharing. It seems that the scheme has more attractive properties than the schemes in [8] and [10], however, it is not easy to acquire the knowledge of nodes' expected location.

The research on key management for HSNs also emerged. The scheme in [12] is a asymmetric predistribution (AP) scheme for heterogeneous sensor networks. High-end nodes which pick a large number of keys in this paper work as cluster heads. Low-end nodes only pick a small number of keys. The scheme reduced the storage overhead at low-end nodes by shifting the storage overhead to high-end nodes.

Recently, two keyed-hash chain based key predistribution schemes [13] [14] were proposed. Instead of generating a large key pool of random keys, a key pool is represented by a small number of generation keys. A seed S and a keyed hash function H are publicly known by the cluster heads and sensors. Each sensor node is preloaded with several generation keys. Each key chain is generated independently via a unique generation key and the seed S by applying the hash function H repeatedly. These two scheme reduce storage overhead greatly. The main problem is that all the key chains corresponding to generation keys stored on the node will be exposed to adversaries as long as the node is compromised. Although the scheme in [13] is a cluster-based scheme, the generated keys are preloaded to sensors randomly from the whole key pool and a compromised sensor will affect the

whole network. The scheme in [14] uses the multi-path key reinforcement method to update the number of iterations to strengthen the security. Even though stronger security is achieved too much communication overhead is needed at the same time.

The existing key predistribution schemes only address the establishment of pairwise keys between low-end nodes. There are some other solutions (e.g.[15] and [16]) address the establishment of network-wide session key in a sensor network. Tian et al. in [15] adopt vector space secret sharing technique to realize general monotone decreasing structures for the family of subsets of users that can be revoked. Furthermore, random numbers were used in the renewal of session keys. The scheme not only keeps the forward and backward secrecy but also resists collusion between the new joined users and the revoked users. The scheme in [16] enables a large and dynamic group of users to establish and renew session keys for secure communications over an unreliable wireless network. The scheme also enables a user to recover, from a single broadcast message, T keys associated with the sessions in which it belongs to the communication group. The proposed scheme has been comprehensively analyzed in an appropriate security model to prove that it is secure.

However, neither pairwise keys nor session keys themselves can satisfy different kinds of communication requirements in heterogeneous sensor networks. Therefore, in this paper, we discuses the establishment of five types of keys. The basic scheme is a improved key predistribution scheme. Other kinds of keys are established based on the pairwise keys which are established in key predistribution stage.

III. Network Model and Design goal

A. Network Model

WSNs are application-specific networks. No scheme can be a one-fits-all security scheme. Therefore, it is important to define a network model according to a concrete application. In this paper, we focus on a military heterogeneous sensor network scenario. The entities involve a BS, a small number of cluster head (CH) nodes and a large number of ordinary sensor nodes. All of these entities form a heterogeneous network structure. The root level is the BS, the second level are powerful cluster heads, and the bottom lever includes a large number of low-cost sensor nodes. An example of such a network model is shown in Figure 1.

1) BS is an infrastructure and assumed to be secure. Because it is usually located far from the wireless network, in our model, it takes charge of the network by manipulating cluster heads. Interaction between BS and low-end sensor nodes is not involved.
2) CHs are equipped with high power batteries, large memory storage, strong data processing capabilities and wide radio communication range. More important, they are equipped with tamper-resistant hardware. Cluster heads can communicate with sensors within the cluster, the BS, and peer cluster heads.
3) Sensor nodes have limited battery power, small memory space, weak data processing capability, and short radio range. Sensor nodes are mobile but with limited mobility. To reduce the energy consumption, a node only communicates with its CH or peer neighboring nodes within the cluster it belongs to.

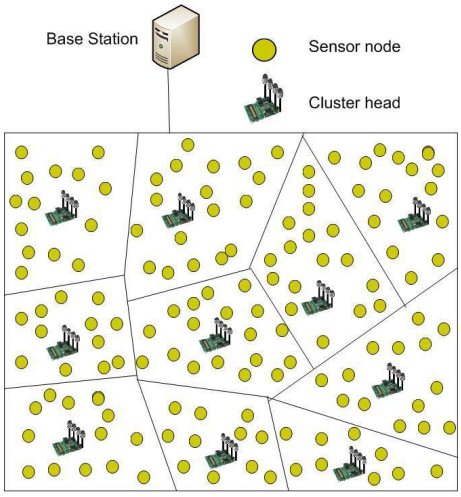

Figure 1. The network model

We assume that the base station will not be compromised. A cluster head might be compromised but the adversaries cannot get the secret data stored on it. However, we assume that if a node is compromised, all the information it holds will be known to the attacker. Further, for simplicity, we assume cluster heads and sensor nodes are uniformly and randomly distributed. For scalability, both cluster head and sensor nodes can be added as needed. For security, the compromised cluster heads and nodes can be revoked.

B. Design goal

1) In view of the fact that no single keying mechanism is appropriate for all secure communications that are needed in sensor networks, we devote to establish five types of keys. These include (1) a master key shared between CHs and BS for sending aggregated data, (2) authentication keys shared by a CH and its cluster nodes for verifying messages shared by them, (3) pairwise keys shared by neighboring nodes within a cluster for intra-cluster node-to-node communication, (4) a cluster key for each cluster for broadcasting within a cluster, and (5) pairwise keys shared by cluster heads for inter-cluster communications.
2) Authentication is required for all types of packets, whereas confidentiality may only be required for some sensitive data. Also, authentication on new nodes is

also necessary as a new node might be an adversary one.

3) The scheme should be resilient against node capture attacks. Once a node is compromised, it only discloses the pairwise keys that is related to it. It will not affect the whole network operation.

IV. The Proposed Scheme

In this Section, we present a key management scheme specifically for HSNs. The essence of the suite of key management is a key predistribution scheme. Just like the existing key management schemes in [13] and [14], the scheme includes three stages: key pool generation, key ring assignment and common key discovery. We will first introduce two definitions which will facilitate the understanding of the scheme.

Definition 1. Keyed hash function is the foundation of keyed hash chain. A keyed hash function H takes a key K and a binary string M of arbitrary length as input, and outputs a binary string of fixed length, which is called hash value h: $h = H(K, M)$. A keyed hash function H satisfies the following two properties:

1) *Given a key K and a string M, it is easy to compute h such that $h = H(K, M)$;*
2) *Given a hash value h and a string M, it is computationally infeasible to find a key K' such that $H(K', M) = h$;*

Definition 2. Keyed hash chain in this paper can be derived as follows:

1) *generating a random key seed X and a generation key K;*
2) *repeatedly applying the same keyed hash function to produce the hash chain. Suppose the expected length of the key chain C is L, the j-th key of the key chain C is generated as follows:*

$$k_{C_j} = \begin{cases} H(K, X), & \text{if } j = 0, \\ H(K, k_{C_{j-1}}), & \text{if } j = 1, \ldots, L-1. \end{cases} \quad (1)$$

A. Key Pool Generation

The size of a key pool should be determined before its generation. The optimum size of the key pool is constrained by the expected probability of key sharing, the expected resilience capability, and the number of keys on each node. The number of keys on each node fully depends on the node's memory. We need to point out the size of a key pool should maintain an acceptable key sharing probability and a reasonable resilience capability for the network.

Different from the large key pool that is generated in the scheme [8], a small key pool which only stores generation keys is adopted in our scheme. Suppose the expected number of keys is P and the length of each key chain is L. X is a publicly known seed. $gk_i (1 \leq i \leq M)$ are generation keys. H is a keyed hash function. Each generation key gk_i has an unique ID IDG_i. Input the seed X and a generation key gk_i to the hash function H, the j-th key of the key chain C_i is generated as follows:

$$k_{C_{i,j}} = \begin{cases} H(gk_i, X), & \text{if } j = 0, \\ H(gk_i, k_{C_{i,j-1}}), & \text{if } j = 1, \ldots, L. \end{cases} \quad (2)$$

The last factor $k_{C_{i,L}}$ is called the commitment of the key chain C_i. We call it $Commit_i$ for simplicity. It is used for authentication and it does not belong to the key pool. Each commitment $Commit_i$ has an unique ID IDC_i. There is a corresponding relation between IDG_i and IDC_i. That is, we refer to the same key chain C_i when we talk about the key chain that is corresponding to either IDG_i or IDC_i.

By iterating the above hash algorithm, we generate $M = P/L$ key chains as well as the commitment for each key chain. It is assumed that no common keys exist between any two key chains. That is $C_i \cap C_j = \phi$ for $1 \leq i, j \leq M$ and $i \neq j$. In addition, BS also generates a master key K_M.

B. Key Ring Assignment

1) Before deployment, each sensor S_i is preloaded with r randomly selected commitments $(< i_1, Commit_{i_1} > , \ldots, < i_r, Commit_{i_r} >)$ where $i_1, \ldots, i_r$ are IDs of commitments. We denote the set of IDs of $< i_1, \ldots, i_r >$ as $[IDs]_{S_i}$. In addition, each node S_i is preloaded with an authentication key $K_{M,S_i} = H(K_M, ID_{S_i})$.
2) Each CH_a is preloaded with $T (T >> r)$ randomly selected generation keys together with the corresponding commitments $(< a_1, gk_{a_1}, Commit_{a_1} >, \ldots, < a_T, gk_{a_T}, Commit_{a_T} >)$ where $a_1, \ldots, a_T$ are IDs of commitments. We denote the set of IDs $a_1, \ldots, a_T$ as $[IDs]_{CH_a}$. Note that we refer to the same key chain C_i when we talk about the key chain that is corresponding to either IDG_i or IDC_i, CH_a only stores IDs of commitments rather than IDs of both commitments and generation keys. In addition, each CH_a is preloaded with the master key K_M.

After the king ring assignment stage, both cluster heads and sensor nodes are uniformly and randomly deployed in the assigned area.

C. Common Key Discovery

1) Cluster Formation. Each cluster head CH_a broadcasts a Hello message to its neighboring nodes with a random delay in order to avoid the collision of the Hello message from neighboring CHs. The Hello message is $< ID_{CH_a}, [IDs]_{CH_a} >$. A node within the network can receive several Hello messages because of large communication range of CHs. Suppose a node S_i receives three Hello messages from CH_a, CH_b, and CH_c respectively. S_i makes a comparison between the intersections $[IDs]_{CH_a} \cap [IDs]_{S_i}$, $[IDs]_{CH_b} \cap [IDs]_{S_i}$, and $[IDs]_{CH_c} \cap [IDs]_{S_i}$. If

$|[IDs]_{CH_b} \cap [IDs]_{S_i}| \geq |[IDs]_{CH_c} \cap [IDs]_{S_i}| > |[IDs]_{CH_a} \cap [IDs]_{S_i}|$, S_i will choose the CH_b as its cluster head. S_i also record CH_c as the backup cluster head. Other nodes also perform as the node S_i. In our scheme, to reduce the energy consumption and the redundant traffics in a network, sensor nodes only communicate with their cluster head and neighboring nodes. We assume two nodes have no interaction if they are located in two clusters.

2) Neighborhood Discovery. Each sensor node S_i broadcasts a Hello message. Suppose CH_a is the cluster head of the node S_i, the Hello message is $< ID_{S_i}, ID_{CH_a}, [IDs]_{S_i} >$ where $[IDs]_{S_i}$ is the set of IDs of commitments. Suppose one of its neighboring node S_j receives the message, S_j first check whether S_i belongs to the same cluster with it or not. If they do not belong to the same cluster, S_j will discard the message. Otherwise, S_j further compares $[IDs]_{S_i}$ and $[IDs]_{S_j}$. If $[IDs]_{S_i} \cap [IDs]_{S_j} = \phi$, S_j just acknowledges with ID_{S_j}. Otherwise, S_j will reply with $< ID_{S_j}, [IDs]_{S_i} \cap [IDs]_{S_j}$. Finally, $S_j >$ adds S_i together with $[IDs]_{S_i} \cap [IDs]_{S_j}$ to its *neighboring nodes list*. After node S_i receives a reply message from neighboring node S_j, S_i adds S_j together with $[IDs]_{S_i} \cap [IDs]_{S_j}$ to its *neighboring nodes list* too. Here we suppose the *neighboring nodes list* is a list of two-tuples. Suppose S_i's neighboring nodes are S_a, S_b, S_c and $[IDs]_{S_i} \cap [IDs]_{S_b} = \phi$. S_i stores $< S_a, [IDs]_{S_i} \cap [IDs]_{S_a} > \| < S_b, \phi > \| < S_c, [IDs]_{S_i} \cap [IDs]_{S_c} >$. These procedures proceed until all the sensor nodes have obtained the neighborhood information.

The procedures are as follows:

a) $S_i \Rightarrow * : ID_{S_i}, ID_{CH_a}, [IDs]_{S_i}$;
b) if $CH_a \neq CH_b$, S_j discards the message;
if $CH_a = CH_b$ & $[IDs]_{S_i} \cap [IDs]_{S_j} = \phi$,
$S_j \rightarrow S_i$: ID_{S_j};
if $CH_a = CH_b$ & $[IDs]_{S_i} \cap [IDs]_{S_j} \neq \phi$,
$S_j \rightarrow S_i$: ID_{S_j}, $[IDs]_{S_i} \cap [IDs]_{S_j}$;
S_j adds ID_{S_i} and $[IDs]_{S_i} \cap [IDs]_{S_j}$ to its *neighboring nodes list*;
c) S_i adds ID_{S_j} and $[IDs]_{S_i} \cap [IDs]_{S_j}$ to its *neighboring nodes list*;

Here, $\Rightarrow$ denotes broadcast and $\rightarrow$ denotes unicast. No authentication mechanism is involved in this stage. It is a general assumption that adversaries do not launch active and explicit pinpoint attack on the nodes during deployment and initialization which usually does not last too long.

3) Shared Pairwise Key Establishment. The node S_i sends messages to its cluster head CH_a. The message contain node's ID ID_{S_i}, nonce, *neighboring nodes list*, and Message Authentication Code(MAC) on all these values. Suppose S_j is a neighboring node of S_i, the cluster head CH_a determines the pairwise key for S_i and S_j by generating a random number k, where $0 \leq k \leq L-1$. k is used as an index in the key chain for selecting the pairwise key. After that CH_a disseminates the shared key information to S_i and S_j. The shared-key information consists of the following: (1) IDs of neighboring nodes (S_i and S_j), (2) the result of $\oplus$ on all the shared generation keys and commitments. If $[IDs]_{S_i} \cap [IDs]_{S_j} = \phi$, CH_a will generate a generation key $gk_{i,j}$ for them. If $[IDs]_{S_i} \cap [IDs]_{S_j} = l$, CH_a will send $gk_l \oplus Commit_l$ to S_i and S_j. If $[IDs]_{S_i} \cap [IDs]_{S_j} = \{m, n\} (1 \leq m, n \leq M$ and $m \neq n)$, CH_a will send $gk_m \oplus Commit_m \oplus gk_n \oplus Commit_n$ to S_i and S_j. For the conditions that two neighboring nodes share more than two commitments, CH_a does the same operation. (3) k, (4) nonce, and (5) MAC which is calculated on all these values using corresponding authentication keys K_{M,S_i} and K_{M,S_j}.

The procedures are as follows:

a) $S_i \Rightarrow CH_a$: ID_{S_i}, nonce, *neighboring nodes list*, $MAC_{K_{M,S_i}}$;
b) If $[IDs]_{S_i} \cap [IDs]_{S_j} = \phi$,
$CH_a \rightarrow S_i$: ID_{S_i}, ID_{S_j}, $E_{K_{M,S_i}}(gk_{i,j})$, k, nonce, $MAC_{K_{M,S_i}}$;
$CH_a \rightarrow S_j$: ID_{S_i}, ID_{S_j}, $E_{K_{M,S_j}}(gk_{i,j})$, k, nonce, $MAC_{K_{M,S_j}}$;
If $[IDs]_{S_i} \cap [IDs]_{S_j} = l$,
$CH_a \rightarrow S_i$: ID_{S_i}, ID_{S_j}, $gk_l \oplus Commit_l$, k, nonce, $MAC_{K_{M,S_i}}$;
$CH_a \rightarrow S_j$: ID_{S_i}, ID_{S_j}, $gk_l \oplus Commit_l$, k, nonce,$MAC_{K_{M,S_j}}$;
If $[IDs]_{S_i} \cap [IDs]_{S_j} = \{m, n\}$,
$CH_a \rightarrow S_i$: ID_{S_i}, ID_{S_j}, $gk_m \oplus Commit_m \oplus gk_n \oplus Commit_n$, k, nonce, $MAC_{K_{M,S_i}}$;
$CH_a \rightarrow S_j$: ID_{S_i}, ID_{S_j}, $gk_m \oplus Commit_m \oplus gk_n \oplus Commit_n$, k, nonce, $MAC_{K_{M,S_j}}$;

CH_a iterates the above procedures to determine the shared pairwise keys for S_i with all the nodes in S_i's *neighboring nodes list*.

After S_i receives the message from CH_a, S_i will proceed as follows:

a) S_i authenticates the message by verifying the message with its authentication key K_{M,S_i}. If the verification fails, S_i will discard the message. Otherwise, go to b).
b) If $[IDs]_{S_i} \cap [IDs]_{S_j} = \phi$, S_i computes the key $H^k(gk_{i,j}, X)$ shared by it and S_j.
If $[IDs]_{S_i} \cap [IDs]_{S_j} = l$, S_i first computes the generation key $gk_l = Commit_l \oplus gk_l \oplus Commit_l$. Then S_i computes $H^k(gk_l, X)$ shared

by it and S_j.
If $[IDs]_{S_i} \cap [IDs]_{S_j} = \{m, n\}$, S_i first computes the generation key $gk_m \oplus gk_n = Commit_m \oplus Commit_n \oplus gk_m \oplus Commit_m \oplus gk_n \oplus Commit_n$. Then S_i computes $H^k(gk_m \oplus gk_n, X)$ shared by it and S_j. For the condition that S_i and S_j share more than two commitments, the key recovery operations are the same.

4) Special Condition. There is a special condition that two nodes share at least one commitment, while their cluster head does not have any of the corresponding generation key. For this condition, the cluster head will turn for help to its neighboring cluster heads. The neighboring cluster heads which has the corresponding generation key will encrypt and send it to the cluster head.

V. Establishment of Other Types of Keys

No single keying mechanism is appropriate for all types of secure communications in sensor networks. Therefore, we explore the way to establish other types of keys. From the aforementioned description we know, cluster heads share the master key K_M with BS. Each node S_i shares a pairwise key with its cluster head $K_{M,S_i} = H(K_M, ID_{S_i})$. Also we have discussed how to establish pairwise keys between neighboring nodes within a cluster. However, in real applications, a cluster key CK shared between a cluster head and all nodes in the cluster is needed in order to broadcast a message within this cluster, and a network key NK shared by all the entities in the network is also needed in order to share some network-wide information. Furthermore, the communication between cluster heads is also necessary. We now discuss the establishment of these keys in turn.

A. Establishment of Cluster Key Within a Cluster

A cluster key for each cluster which can facilitate secure broadcast within the cluster is necessary. In our scheme, the cluster key CK_a is generated by CH_a. After its generation, CH_a will encrypt with the shared key between itself and each node. Finally, cluster head CH_a sends the following message to a cluster member S_i:

$$CH_a \rightarrow S_i : E_{K_{M,S_i}}(CK_a),$$

where K_{M,S_i} is the shared key between CH_a and node S_i.

The cluster key must be updated once a node in the cluster is compromised. The updating of the cluster key is similar to the establishment of a new cluster key. CH_a generates a new cluster key and encrypts it with shared key between itself and legal nodes. Finally, CH_a sends the encrypted cluster key to legal nodes in the cluster.

In real applications, a node may broadcast messages to its neighboring nodes. We do not explore how to share a cluster key between neighboring nodes as this kind of broadcasting happens infrequently. If a node wants to share a message with its neighboring nodes, it encrypts the message with the pairwise keys and then links up the encrypted messages with IDs of the expected nodes. When a node receives the message, it first checks whether its ID is included in the message. If so, it will decrypt the corresponding part. Otherwise, it will discard the message.

We do not explore the establishment of network key NK here. If BS wants to share information with all the nodes, it broadcasts the message to cluster heads. Cluster heads then broadcast the message within their clusters.

B. Establishment of Pairwise keys Between Clusters

In order to keep communication secure between clusters, each cluster head has to agree on pairwise keys with cluster heads in its communication range. The pairwise key shared by two cluster heads CH_a and CH_b is generated as follows:

$$K_{CH_a,CH_b} = H(K_M, ID_{CH_a} \parallel ID_{CH_b}),$$

where H is hash function, K_M is the master key, and $\parallel$ is the string concatenation function. The pairwise keys established in this way guarantee that sensor nodes cannot decrypt massages shared by cluster heads. However, the communication between two cluster heads can be seen by other cluster heads. We don't think this characteristic destroy the security because all the cluster heads have the same security level.

Up to now, we have discussed the establishment of five types of keys. Please see Table I for details.

Table I
Five types of keys

Key	Entities	Function
Master key K_M	BS and CHs	Keeps secure communications between BS and CHs
Authentication key K_{M,S_i}	CH and S_i	Authenticates messages shared by CH and S_i
Pairwise key K_{S_i,S_j}	S_i and S_j	Keeps secure communications between S_i and S_j
Cluster key CK_a	CH_a and cluster nodes	Keeps secure broadcasts within the cluster
Pairwise key K_{CH_a,CH_b}	CH_a and CH_b	Keeps secure communications between CH_a and CH_b

VI. Security Analysis

In this section, we discuss several security issues in HSNs, including node revocation, node addition, and resisting node capture attacks. We show our scheme satisfy these security requirements.

A. Node Revocation

In existing keyed-hash chain based schemes [13] [14], the compromise of a node will result in the disclosure of all the key chains corresponding to generation keys which it stores. While in our scheme, a node only stores commitments. No network-wide secret information is disclosed

even if a large number of nodes are compromised. Therefore, only neighboring nodes delete the pairwise keys they share with the compromised node once a sensor is compromised by an adversary. The cluster head will send a revocation message to neighboring nodes of the compromised node. The message is authenticated by the authentication keys. The cluster head will add the ID of the compromised node to *Revoked Node List*.

B. Node Addition

Addition of new nodes is necessary for key management schemes in order to achieve the scalability property or to substitute compromised nodes. However, it poses a challenge on security schemes because a newly deployed sensor could have been compromised. Therefore, an efficient way to authenticate new nodes is of great importance. Recall that cluster heads have the master key K_M, each new sensor S_{new} is preloaded with a authentication key which is $K_{M,S_{new}} = H(K_M, ID_{S_{new}})$. After the node S_{new} is deployed in the network, S_{new} sends join request to a cluster head CH_a. The request is given as below:

$$S_{new} \rightarrow CH_a : ID_{S_{new}}, nonce, \\ MAC_{K_{M,S_{new}}}(ID_{S_{new}}, nonce).$$

CH_a first checks whether $ID_{S_{new}}$ is included in the *Revoked Node List*. If $ID_{S_{new}}$ is included in the *Revoked Node List*, CH_a will discard the message. Otherwise, CH_a generates $K_{M,S_{new}}$ and then authenticates the node S_{new} by verifying the MAC. After that, S_{new} determines its neighboring nodes as described in the stage of Neighborhood Discovery. We do not discuss the attacks launched by a compromised node which has not been detected by the cluster head.

C. Resisting Node Capture Attack

Because wireless communication is not secure, an adversary can not only eavesdrop on all communications but also inject packets or replay messages. Also, an adversary may inject new nodes into the network. In our scheme, all the sensitive data is authenticated by MAC or encrypted by corresponding keys. When a new node is added into the network, the first step is to check its validity.

It is infeasible to setup tamper-resistance hardware for each sensor node. Once a node capture attack is launched, all the information the compromised node holds will be known to the attacker. In our scheme, a node only stores commitments rather than generation keys. Once a node is compromised, only commitments and pairwise keys it shares with the cluster head and its neighboring nodes are disclosed. According to the one-way property of the hash function, the adversary cannot get any key on the key chain from the commitment. As long as the cluster head updates the session key within the cluster and deletes the pairwise keys which belong to the compromised node, this scheme is secure.

VII. Performance Analysis

The storage overhead of our scheme is the same as that of [13]. The only difference is that nodes store commitments rather than generation keys. The result in [13] shows the scheme can significantly reduce the storage load.

The memory of a sensor is fixed after the sensor is produced. The largest number r of commitments stored on each sensor is constrained by the memory of the sensor. Once r is set, the larger of key pool size P is, the smaller probability that two nodes share a key; the smaller the key pool size P is, the larger the impact on other sensor's communication when a fixed number of sensors are compromised. We want to find the largest key pool size that the probability of a node and its cluster head sharing a key chain is no less than a threshold p.

Suppose the number of key chains is M. A sensor node has $C(M, r)$ different ways of selecting r commitments from a key pool with the size P.

$$C(M, r) = \frac{M!}{r!(M-r)!}. \tag{3}$$

A cluster head has $C(M, T)$ different ways of selecting T generation keys from a key pool with the size P.

$$C(M, T) = \frac{M!}{T!(M-T)!}. \tag{4}$$

Therefore, the total number of ways for a sensor node to pick up r commitments and a cluster head to pick up T generation keys respectively is $C(M, r)C(M, T)$.

$$C(M, r)C(M, T) = \frac{M!}{T!(M-T)!}. \tag{5}$$

In [13], the probability that a sensor node and a cluster head node share a common key can be given as $p_{[13]} = 1 - Pr[\text{a node and a cluster head node do not share any key}]$.

$$p_{[13]} = 1 - \frac{(M-r)!(M-T)!}{M!(M-r-T)!}. \tag{6}$$

While in our scheme, we use a new cluster mechanism which can increase the probability of key sharing between sensor nodes and their clusters. In order to simplify the analysis, we suppose each node can receive broadcast messages from *three* cluster heads. The node chooses a cluster head with whom it shares the largest number of commitments as its cluster head. Therefore, the probability that a sensor node and its cluster head share a common key can be given as $p_{our} = 1 - Pr[\text{a node does not share any key with three cluster heads}]$.

$$p_{our} = 1 - (\frac{C(M, r)C(M-r, T)}{C(M, r)C(M, T)})^3. \tag{7}$$

In order to highlight the advantage of our scheme, we also calculate the probability of sharing at least one key between

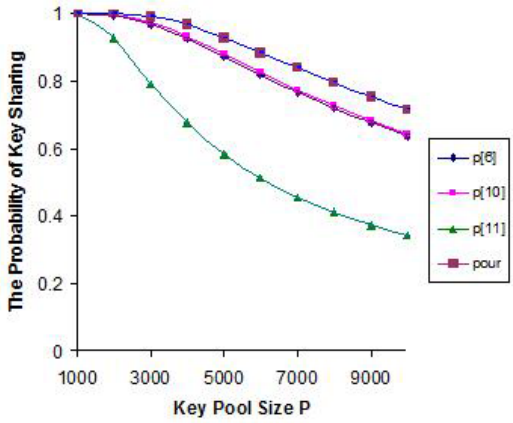

Figure 2. Comparison of key sharing probability for different key pool size in schemes [8], [12], [13] and our proposed scheme

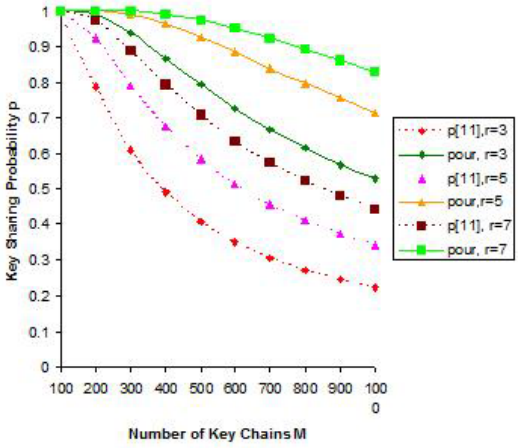

Figure 3. The relation between the key sharing probability and sensors' storage overhead

two nodes in basic schemes [8] and [12].

$$p_{[8]} = 1 - \frac{C(P-m,m)}{C(P,m)}, \tag{8}$$

where m is the number of keys stored by each sensor node.

$$p_{[12]} = 1 - \frac{(P-l)!(P-L)!}{P!(P-l-L)!}, \tag{9}$$

where l is the number of picked by a low-end sensor node and L is number of keys picked by a high-end sensor node.

We derive the probability of sharing at least one key for schemes [8] [12] [13] and our scheme in Figure 2. In Figure 2, we suppose the key pool size in schemes [8] and [12] ranges from 1000 to 10000, with an increment of 1000. Suppose the length of each key chain is 10, the key pool size in scheme [13] and our scheme ranges from 100 to 1000, with an increment of 100. The corresponding parameters $[T, r, L, l, m]$ are $[80, 5, 500, 20, 100]$. The relationship between these parameters are $L \times l = m^2$, $L > T$, and $l > r$.

From Figure 2, we can see that the proposed scheme achieves the higher probability of key sharing while it keeps low storage overhead.

It is easy to come to the conclusion that the probability of key sharing between a node and its cluster head increases with the number of commitments in sensor nodes. In Figure 3, we show our scheme always has higher key sharing probability than scheme [13] for different key pool sizes and different number of keys stored in sensors.

For keyed-hash chain based random key predistribution scheme, one of the main problems is how to determine the length of the key chain. Once the key pool size is fixed, the longer the key chain, the smaller is the number of key chains. A small number of key chains contribute to reducing the storage overhead. However, in terms of security, long key chains may weaken the security of the scheme. How to get the best key chain length which keeps an acceptable security level and storage overhead is an open problem.

VIII. Conclusion

We have presented a key management scheme for Heterogeneous Sensor Networks in this paper. The scheme is based on keyed-hash chain. It has the following properties:

1) The scheme supports the establishment and renewal of five types keys in the network because a single key can not satisfy different communication requirements.
2) Each sensor node only stores several commitments instead of generation keys. On the one hand, it reduces storage overhead and the network is still secure even if a large number sensor nodes are compromised. While in previous schemes, once a node is compromised, all the key chains corresponding to the generation keys stored on the node will be disclosed.
3) A new cluster mechanism is used to improve the probability of key sharing probability between the sensors and their cluster head. This mechanism facilitates the management of clusters.

Through security analysis and performance analysis, we show the scheme meets the security requirements in resource-constrained sensor networks

References

[1] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, *An Application-Specific Protocol Architecture for Wireless Microsensor Networks*, IEEE Transactions on Wireless Communications, Vol.1, No.4, Oct. 2002.

[2] P. Gupta and P. R. Kumar, *The Capacity of Wireless Networks*, IEEE Transactions on Information Theory, Vol,46, No.2, pp.388-404, Mar. 2000.

[3] E. J. Duarte-Melo and M. Liu, *Data-gathering Wireless Sensor Networks:Organization and Capacity*,Vol.43, Issues.4, pp.519-537, Nov. 2003.

[4] K. Lu, Y. Qian and J. Hu, *A framework for distributed key management schemes in heterogeneous wireless sensor networks*, The 25th IEEE International Conference on Performance, Computing, and Communications (IPCCC), pp.513-519, Arizona, USA, April 10-12, 2006.

[5] S. Han, E. Chang, L. Gao, T. S. Dillon, *Taxonomy of Attacks on Wireless Sensor Networks*, Proceedings of the 1st European Conference on Computer Network Defence (EC2ND), Springer Press, Springer LinkDate: December 2007.

[6] N. Gura, A. Patal, A. Wander, H. Eberle, S. C. Shantz, *Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs*, Proceedings of the 6th International Workshop on Cryptographic Hardware and Embeded System, Aug. 2004.

[7] A. Hodjat and I. Verbauwhede, *The Energy cost of Secrets in Ad-Hoc Networks*, Proceedings of IEEE CAS Workshop on Wireless Communication and Networking, Sep. 2002.

[8] L. Eschenauer and V. Gligor, *A Key Management Scheme for Distributed Sensor Networks*, Proceedings of the 9th ACM Conference on Computer and Communications Security, pp.41-47, 2002.

[9] H. Chan, A. Perring, and D. Song, *Random Key Predistribution Scheme for Sensor Networks*, Proceedings of the IEEE Symposium on Security and Privacy, 2003.

[10] D. Liu and P. Ning, *Establishing Pairwise Keys in Distributed Sensor Networks*, Proceedings of the 10th ACM Confence on Computer and Communications Security, pp.52-61, 2003.

[11] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, *A Key Management Scheme for Wireless Sensor Networks using Deployment Knowledge*, Proceedings of IEEE Conference on Computer and Communications (INFOCOM), 2004.

[12] X. Du, Y. Xiao, M. Guizani, and H. H. Chen, *An Effective Key Management Scheme for Heterogeneous Sensor Networks*. Ad Hoc Network, No.5. Vol.1, pp.24-34, 2007.

[13] F. Kausar, S. Hussain, L. T. Yang, and A. Masood, *Scalable and Efficient Key Management for Heterogeneous Sensor Networks*, Special Issues in Journal of Supercomputing, pp.44-65, 2008.

[14] S. Hussain, M. S. Rahman, and L. T. Yang, *Key Predistribution Scheme using Keyed-Hash Chain and Multipath Key Reinforcement for Wireless Sensor Networks*, IEEE International Conference on Pervasive Computing and Communication, pp.1-6, Mar. 2009.

[15] B. Tian, S. Han, T. S. Dillon, S. Das, *A Self-healing Key Distribution Scheme Based on Vector Space Secret Sharing and One Way Hash Chains*, Proceedings of the 9th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Newport Beach, CA, USA, Jun. 23-27, 2008.

[16] S. Han, B. Tian, M. He, E. Chang, *Efficient Threshold Self-healing Key Distribution with Sponsorization for Infrastructureless Wireless Networks*, The IEEE Transactions on Wireless Communications, Vol.8, No.4, pp.1876-1887, Apr. 2009.