A Key Distribution and Management Scheme for Clustered Ad Hoc Sensor Networks

by

Fei Liu

A Thesis Submitted to
the Faculty of Graduate Studies and Research
in Partial Fullfillment of
the Requirements for the Degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School Of Computer Science

Carleton University

Ottawa, Ontario

September 9, 2013

Abstract

Presently, the most practical approach for bootstrapping initial secret keys in sensor networks is to load keys into sensor nodes before they are deployed [54, 57]. These initial keys are used to establish pairwise keys for node-to-node communications. After that, the pairwise keys are used to distribute cluster keys for broadcasting messages. However, some of them are vulnerable to impersonation attacks. We describe a novel key distribution and management scheme for clustered ad hoc sensor networks. The scheme uses the Boneh-Franklin's ID-based encryption (IBE) scheme and Yi's ID-based signature scheme to achieve mutual authentication between nodes [50]. The signature scheme is used to distribute a cluster key which can be updated. We also derive a master key from the signature which can also be updated when needed. Our contribution is that we resolved the impersonation problems that exist in current key distribution schemes for ad hoc sensor networks. A timestamp is incorporated in the signing procedure, avoiding message replay attacks. Finally, This scheme can be extended to hierarchical ad hoc sensor networks.

Contents

List of Figures

List of Acronyms

IBE	ID-based Encryption
KGC	Key Generation Center
CA	Certificate Authority
CH	Cluster Head
ECC	Elliptical Curve Cryptography
WSN	Wireless Sensor etwork
MANET	Mobile Ad Hoc Network
OLSR	Optimized Link State Routing
DCA	Distributed Clustering Algorithm
DMCA	Distributed Mobility-adaptive Clustering
NRT	Neighbourhood Record Table
PKG	Private Key Generator
PKI	Public Key Infrastructure
CBC	Certificate-based Cryptography
IBC	ID-based Cryptography
MN	Member Node
CM	Cluster Member
DLP	Discrete Logarithm Problem

LID	Lowest-identifier
CDS	Connected-dominated-set
MCDS	Maximum Connected-dominated-set
RCA	Ring Clustering Algorithm
BFS	Breadth First Search
MIS	Maximal Independent Set

Chapter 1

Introduction

1.1 Context and Problem Statement

Key management in ad hoc networks is challenging. There are symmetric, asymmetric and hybrid key distribution schemes. There are two kinds of methodologies for asymmetric key distribution: ID-based cryptography with distributed key generation centers (KGCs) [36], and certificate-based cryptography with distributed certificate authorities (CAs) [1, 27, 40, 55]. However, the way KGCs or CAs collaborate can generate considerable overhead is energy consuming and does not scale well because the use of threshold cryptography [45] requires all cluster heads (CHs) to participate in key generation. Also, they promote group key schemes for local broadcast messages. Since all nodes participate in the generation of a group key, a lot of traffic occur. Generally speaking, the asymmetric key schemes are infeasible for nodes that have limited energy and networks with limited bandwidth.

On the other hand, symmetric key schemes are promising as they are not only encryption efficient but also energy saving. Currently, the most practical

approach for bootstrapping secret initial keys in sensor networks is to use pre-deployed keying. Keys are loaded into sensor nodes before they are deployed [57]. The existing symmetric key schemes [54, 57] to secure ad hoc networks use pairwise keys for node-to-node communications and cluster keys for group communications

However, these approaches [54, 57] are vulnerable to impersonation attacks. as they use node (x, y) coordinates, [54] and node IDs [57].

1.2 Objective and Solution

The objective of the thesis is to propose an impersonation free key distribution and management scheme for clustered ad hoc sensor networks. Our contribution is that we resolved the impersonation problem that existed in these key distribution schemes for ad hoc sensor networks. To resolve the problem, a hybrid scheme combining ID-based cryptography, by Boneh-Franklin [5], with a symmetric cluster key scheme is proposed. A base station acts like an offline KGC and preloads each node with an ID-based public and private key pair. Boneh-Franklin's ID-based cryptography is efficient in encryption and decryption. It uses elliptical curve cryptography (ECC), rather than RSA. Moreover, this cryptography system does not need to issue certificates.

An ID-based signature scheme by Yi is also used [50]. The signature on a message, plus appended timestamp, is used for mutual authentication. For each update of a cluster key, the signature is also updated. If authenticated successfully, then the cluster head (CH) uses each member node's signature to derive each node's master key. It uses these master keys to distribute the cluster key. The nice properties of the signature are that no one can forge it, no one can gain additional

information about the private key from the signature, and the signature can be updated when needed to avoid key divulgation.

The cluster key distribution scheme only needs one encryption to distribute the cluster key to all its member nodes. A cluster key is needed for message broadcasting. The reason is that in-network processing operations, such as data aggregation and passive participation, are prohibited by using only individual keys for node-to-node communications. ID-based cryptography keys are used for node-to-node communications.

The use of ID-based cryptography keys for node-to-node communications sacrifices some of the encryption and decryption efficiency in comparison to using symmetric pairwise keys. Nonetheless, to the best of our knowledge this is the most efficient asymmetric key scheme we can find. The impersonation problem is resolved. The security is improved because one node's capture does not release the secret of the other nodes. The current schemes are susceptible to node capture. Moreover, there is no need to distribute certificates. This is different from the current scheme that is susceptible to node capture.

1.3 Organization of the Thesis

The outline for the rest of the thesis is as follows. In Chapter 2, we talk about the key management schemes in ad hoc sensor networks and various techniques that are used in this area. Especially, we emphasize the key escrow problem in identtiy-based encryption (IBE) and the contermeasures for IBE scheme. In Chapter 3, we look at recent solutions for doing key management in ad hoc sensor networks through a literature review. In Chapter 4, we introduce the techniques

we use for our key distribution and management scheme. In Chapter 5, we present our key distribution and management scheme for ad hoc sensor networks and the analysis behind our scheme choice. In Chapter 6, we extend our scheme to a hierarchical model. We also analyze our extended scheme. Finally, in Chapter 7, we draw conclusions based on our research and discuss about the future work of this management scheme.

Chapter 2

Background

2.1 Introduction

This chapter covers the background that is needed for our key distribution and management scheme in clustered ad hoc networks. Key distribution and management scheme is a comprehensive solution which involves many aspects and techniques. First, we talk about the model and characteristics of ad hoc sensor networks. Then we discuss about clustering algorithms that organize sensors in this type of network. By reviewing the various IBE schemes, it helps explain why we chose the Boneh-Franklin's [4] IBE scheme. However, the IBE scheme has an inherent key escrow problem. We investigate the countermeasures to mitigate the problem and also review various ID-based signature schemes to explain why we choose Yi's [50] ID-based signature for our key management scheme.

2.2 Ad Hoc Sensor Networks

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors that monitor physical or environmental conditions such as temperature, sound and pressure, as they cooperatively pass their data through the network to a main location. Typically, in a wireless sensor network, there are base stations that the sensors try to connect and send their sensed data to. The data is sometimes aggregated to a node called a gateway sensor node which is directly connected to the base stations. The sensors have properties typically such as limited battery, limited memory, mobility, and the whole network usually has limited bandwidth.

The more modern networks are bi-directional, it enables control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. Today, such networks are used in many industrial and consumer applications such as industrial process monitoring and control, machine health monitoring.

2.3 Modelling Ad Hoc Networks

An ad hoc network is usually modelled as an undirected graph $G = (V, E)$ where V is the set of the nodes in the ad hoc network and $|V| = n$, is the the size of the network. If node u and v can mutually connect to each other (within each other's transmission range), then there is an edge $u, v \in E$. This implies that all the links between the nodes are bi-directional, and u and v are neighbours. We denote the set of neighbors of a node to be $v \in V$ as $\tau(v)$

We can see from Figure 2.1 that nodes in this network can be any entity

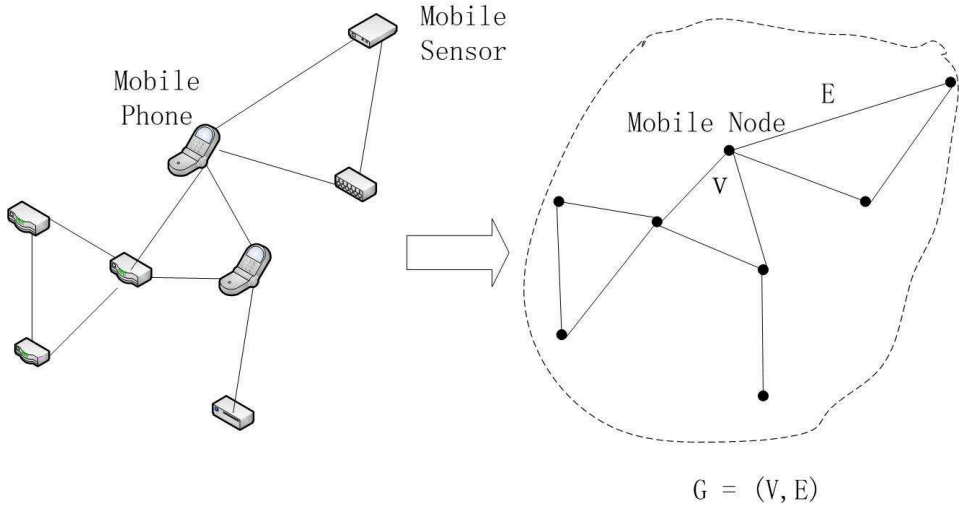

Figure 2.1: Modelling ad hoc networks

that can communicate with other entities such as mobile phones, sensors or even vehicles that are mobile. The transmission range for different entities may vary, but to form an edge, the link between the two entities should be bi-directional. Different nodes in this networks may have different capacities such as different battery life, different memory or velocity.

2.4 Hierarchical Ad Hoc Networks

Clustering is a method by which nodes are organized by some parameter metrics such as a node's mobility, energy or trust levels. The reason for clustering is the scalability of routing in ad hoc networks by increasing the robustness of routes. When a node is joining or leaving a network, only the nodes within the cluster need to update their routing tables. In a time division or frequency division scheme, clustering is also crucial for controlling the spatial reuse of the shared channel. This technique is aimed to minimize the amount of data exchanged for routing

information as well as to build and maintain cluster-based virtual network architectures. There are existing solutions for the clustering of ad hoc networks. The clustering is performed in two phases: clustering set up and clustering maintenance. The setup phase is mainly the process to elect the cluster head nodes and partition the networks into groups in such a way that a cluster is associated with its neighbour nodes. The cluster head nodes can be considered a super node that acts as a local coordinator for the operations of its cluster (as in [30, 16, 20]).

In hierarchical ad hoc networks, nodes are organized by clusters and by levels. This is practical because some nodes may have several interfaces. In this kind of network, a node with several interfaces can play the role of CH in several clusters and even several levels. Some of the CH nodes with several interfaces can be promoted to be member nodes in upper levels. Meanwhile, some CH nodes in lower levels can also be promoted to be the CH in upper level. Figure 2.2 is an example model of hierarchical ad hoc networks with three levels.

We can see from Figure 2.2 that nodes are formed by clusters and organized hierarchically. Nodes have multiple interfaces, for example, node A has two interfaces which are A_1 and A_2. It plays the role of CH in cluster C_1 of level 1 and also is the CH of cluster C_2 of level 2. Node C has three interfaces and plays the role of CH at all three levels. However, some nodes play the role of CH at only one level.

There are routing and clustering formation standards for hierarchical ad hoc networks which will be discussed in the next section.

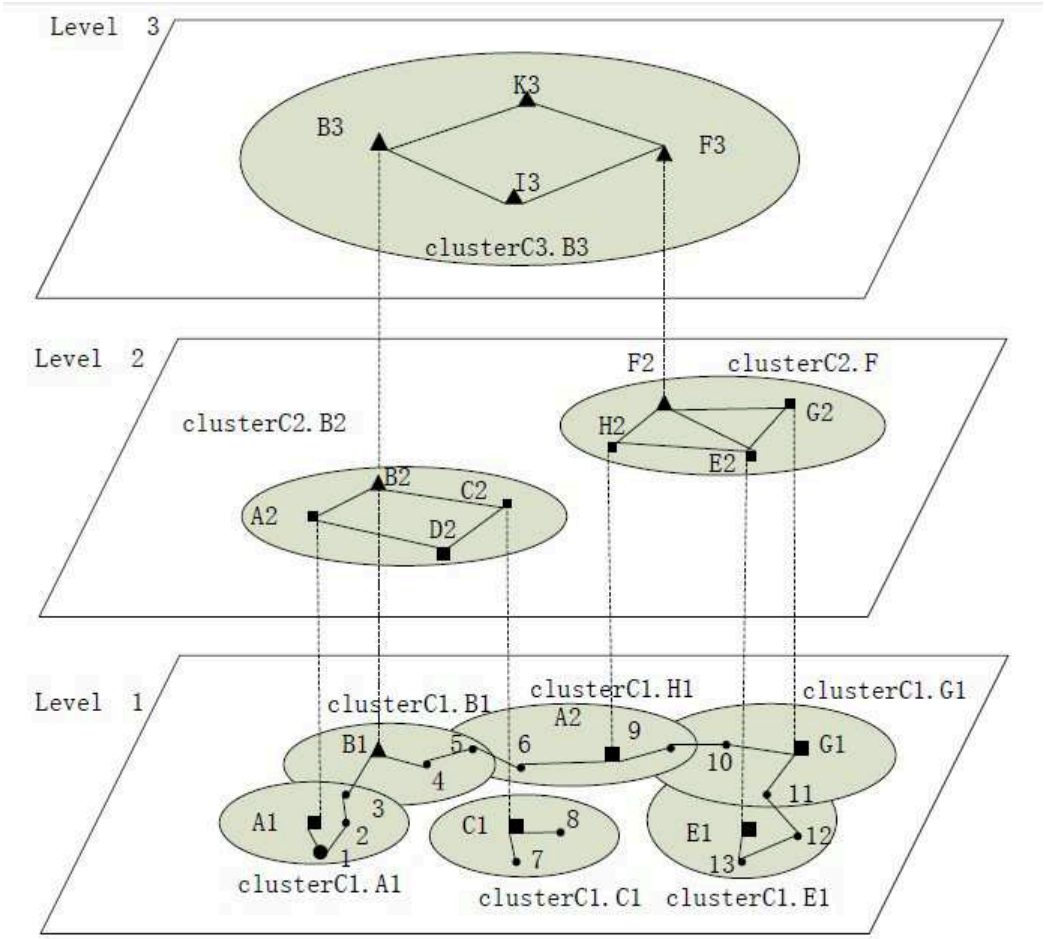

Figure 2.2: Example of a hierarchical architecture with mobile nodes

2.5 Cluster Formation Algorithms

In this section, we exam various clustering algorithms for ad hoc sensor networks. The main focus will be on distributed clustering algorithms. Distributed clustering algorithms is practical in the sense that the nodes are organized distributedly and each super node is responsible for its local cluster. In this topology, there is no central control over the network, and each super node only needs to update its local topology when the network topology changes. This reduces a lot of communication cost. Our key distribution and management scheme is also based on the

network model which is organized by distributed clustering algorithms because of their prevalence for clustering algorithms. We reviewed various distributed clustering algorithms in order to know whether our key management scheme can be mostly adopted on the topology that is organized by these dstributed clustering algorithms. After reviewing these distributed clustering algorithms we know that there are some common properties among all distributed clustering algorithms. For example, they all have super nodes which are known as CH and they all have CM nodes that surround their CH even though they differs in the metric to elect the cluster head.

The existing clustering algorithms differ on the criterion for the selection of cluster heads. For early clustering algorithms [3, 37], the choice of cluster heads is based on the unique identifier (ID) associated to each node: the node with the lowest ID is selected as the cluster head, then the cluster is formed by that node and all its neighbors. The same procedure is repeated among the remaining nodes, until each node is assigned to a cluster. This kind of algorithm is known as the lowest-identifier (LID) based clustering algorithm [37], whych selects the node with the lowest ID as a cluster-head. The system performance is better than the Highest-Degree in terms of throughput. Major drawback of this algorithm is that it is biased towards nodes with smaller IDs which may lead to battery drainage of certain nodes. It also does not attempt to balance the load uniformly across all nodes.

The election of a cluster head can also be based on the maximum degree. However, recent clustering algorithms [25, 34, 29] take into consideration many more factors such has ecurity and energy consumption. The metric to elect the cluster head is not a single factor such as those in early algorithms. The election

criteria can be a metric of many parameters which may carry different weights depending on the algorithms requirements. We will talk about these algorithms below.

2.5.1 Distributed Clustering for Ad Hoc Networks

Basagni proposed a distributed clustering algorithm (DCA) and a distributed mobility-adaptive clustering (DMAC) algorithm [3] for ad hoc networks in 1999. Both the Distributed Clustering Algorithm and Distributed Mobility-Adaptive Clustering algorithm can deal with a node's mobility. The difference is that the former one is based on a weak assumption that the node does not move during cluster formation. The latter releases the weak assumption and deals directly with the node's mobility during cluster formation. So the latter one is more robust. In this algorithm, the cluster head is elected based on a generic weight associated to each node: The bigger the weight of a node, the greater the chances that the node will play the role of cluster head. This algorithm has two properties: 1. Every neighbour has at least one cluster head. 2. Two neighbour nodes can't be the cluster head at the same time. 3. The node joins the neighbour node's cluster who has the maximum weight. The first property ensures that every ordinary node has direct access to at least one cluster head. The second property is trying to form clusters in a distributed and balanced manner. The third property ensures that the ordinary node always joins the cluster with the biggest weight nodes. In addition, nodes only require its one hop neighbour node's information such as ID and weight.

Figure 2.3 is an example of a network that applies Basagni's distributed clus-

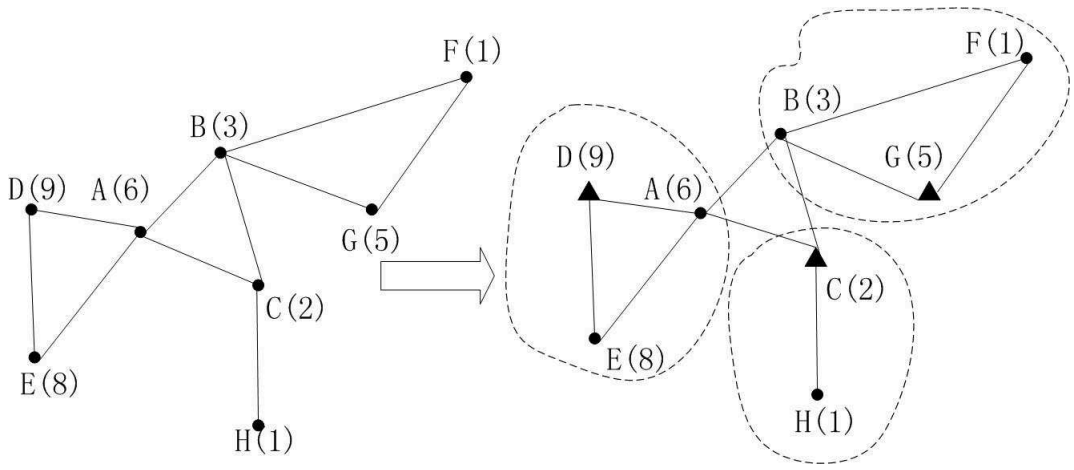

Figure 2.3: An ad hoc network and its clustered topology

tering algorithm. This network consists of eight nodes A, B, C, D, E, F, G, H with the different weights associated to them. The ordinary nodes are denoted as circles and the cluster head nodes are denoted as triangles. From the initial state, we know that node D is the node that has the maximum weight among all its neighbour nodes, so it is elected as the cluster head. Node E becomes the member node of D's cluster. Node A also joins D's cluster because among all its neighbour nodes, D is the one that has the maximum weight. Node G is also the node among all its neighbour nodes, so G becomes the cluster head and node B, F joins the cluster. There are nodes C and H left. Though neighbour nodes A and B are within the C's transmission range, none of them are CH. so C can't join their clusters as a ordinary node. The only option for C is to be the cluster head of the ordinary node H which has a smaller weight than H. In this approach, each node is assigned weights (a real number greater than zero) of being a CH. A node is chosen to be a CH if its weight is higher than any of its neighbor's weight; otherwise, it joins a neighboring CH. Since node weights vary, computing the CH becomes very expensive and there are no optimizations on the system parameters

such as throughput and power control.

How to form the network into clusters without generating two much overhead to the network is challenging. Many researchers are currently focusing on efficient clustering algorithms. They came up with many approaches many approaches to provide an efficient and effective method to update a cluster structure in response to underlying network topology changes. The main objective of clustering in mobile ad-hoc network environments is how can the optimal cluster head be elected and how can the optimal number of clusters be achieved through partition without degrading the whole network's performance too much. Hussein proposed an efficient weighted distributed clustering algorithm for mobile ad hoc networks [29] in 2010 known as CBMD. The name comes from the cluster head election parameters: connectivity (C), residual battery power (B), average mobility (M), and distance (D) of the nodes to choose locally optimal clusterheads. This algorithm tries to maintain a stable clustering structure and minimizes the total number of clusters in the network. It also attempts to minimize the overhead during cluster formation as well as the maintenance to maximize the lifetime of mobile nodes in the system by reducing the battery cost for clustering. The way they evaluate the parameters such as connectivity (C), residual battery power (B), average mobility (M), and distance (D) of the node is interesting. The first parameter is connectivity. If the node with the highest connectivity is elected as cluster head, the link will be congested and the battery power will be consumed rapidly. If the node with the lowest connectivity is elected as cluster head, then the networkwill have a low cluster size which is similar to one's without clusters. The advantage of clustering in this case would be lost. Nodes with the highest connectivity or node's with the lowest connectivity should not be elected as cluster heads. The

connectivity is measured by the number of nodes that a cluster head can support at the same time. The second parameter is the residual battery power. The battery power of a node is related to the lifespan of a node which is also related to the lifetime of the whole network. The cluster head acts as a supernode that takes on more responsibility than other ordinary nodes, such as some coordinating nodes or aggregating traffic. Therefore, it is vital to balance the energy consumption among nodes to avoid node failures. The third parameter is the average mobility of the node. In most cases, when considering the mobility of a node, it requires at GPS to locate each mobile nodes precise position and velocity. In this scheme, each node measures its own average mobility M_i by its neighbour node's hello message. Each node maintains a short Neighbourhood Record Table (NRT). The Neighbourhood Record Table consists of the IDs of neighbouring nodes and is constructed by the hello message from neighbouring nodes. The value is calculated as $M_{i,n} = \frac{1}{n-1} \sum_{i=1}^{n} |NRT_{t(i+1)} \setminus NRT_{t(i)}|$. $M_{i,n}$ which is the mobility value calculated by node i using the latest n hello messages it receives during the current time t. The mobility here is a reliable indicator of the node's future mobility trend. Generally, the mobility trend is measured by the deviation of the changes of its neighbour nodes using the neighbour node's ID. The smaller the number, the node's mobility trend becomes more reliable.

Let's show how the above works. For example, there are four hello messages from its neighbour nodes during time t, after four successive hello messages, the nodes 4 and 6 have four Neighbourhood Record Table entries respectively which are depicted in Table 2.1 and Table 2.2:

The hello messages from its neighbour nodes during time t is broadcasted periodically. We can see from Table 2.1 and Table 2.2, after four successive

Table 2.1: Neighbour Node Table of Node 4

Hello Message	*Neighboring Nodes*	NRT_i
0	1 21 5 10 7 - -	0
1	1 21 5 - - 11 -	3
2	1 - 5 - - 11 -	1
3	1 21 5 - - - 9	3

Table 2.2: Neighbour Node Table of Node 6

Hello Message	*Neighboring Nodes*	NRT_i
0	2 11 - - -	0
1	2 - 7 8 -	3
2	2 - 7 - 9	2
3	2 - 7 - -	1

hello messages, the nodes has four Neighbourhood Record Table entries: $NRT_1 = 1, 21, 5, 10, 7$, $NRT_2 = 1, 21, 5, 11$, $NRT_3 = 1, 5, 11$, $NRT_4 = 1, 21, 5, 9$. So, the average mobility of node 4 during time t is calculated below:

$$M_{4,4} = \frac{1}{4-1}\sum_{i=1}^{4}(i-1)|NRT_{t(i+1)} - NRT_{t(i)}|]$$

$$M_{4,4} = \frac{1}{3}[3 * |NRT_4 - NRT_3| + 2 * |NRT_3 - NRT_2| + 1 * |NRT_2 - NRT_1| + 0]$$

$$= \frac{1}{3}[(3 * 3) + (2 * 1) + (1 * 3)] = 4.67$$

For mobile node with $ID = 6$, after four successive hello messages, the nodes has four Neighbourhood Record Table entries $NRT_1 = 2, 11$, $NRT_2 = 2, 7, 8$, $NRT_3 = 2, 7, 9$, $NRT_4 = 2, 7$. Thus, the average mobility of node 6 during time t is calculated below:

$$M_{4,4} = \frac{1}{4-1}[\sum_{i=1}^{4}(i-1)|NRT_{t(i+1)} - NRT_{t(i)}|]$$

$$M_{4,4} = \frac{1}{3}[3 \times |NRT_4 - NRT_3| + 2 \times |NRT_3 - NRT_2| + 1 \times |NRT_2 - NRT_1| + 0]$$

$$= \frac{1}{3}[(3 \times 1) + (2 \times 2) + (1 \times 3)] = 3.33$$

From the above calculation, node 4's mobility indicator is more than the one of node 6, so we know that node 6 is more stable than 4 which means node 6 has a better chance to be a cluster head than node 4.

The fourth parameter is the distance between a node and others within transmission range. It is better to elect a cluster head with the nearest members. This can potentially enhance cluster stability. For node i, D is computed as the cumulative mean square distance to neighbours divided by the total number of neighbours as shown in the formula below:

$$D = \frac{1}{d_i} \sum_{j \in N(i)} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

where (x_i, y_i) and (x_j, y_j) are the coordinates of the node i and j respectively and d_i is the degree of node i which is the number of neighbours of each node i.

2.6 (t, n) Threshold Cryptography

The idea of the threshold cryptography is proposed by Shamir [45]. The goal of threshold cryptography is to share a secret safely. For example a (t, n) threshold scheme means that if someone wants to share a secret among n shareholders, in order to decrypt the secret, at least t of the shareholders must agree to participate in the decryption process. This is under the assumption that compromising $t-1$ of the shareholders will not leak the secret.

The implementation of threshold cryptography is based on polynomial interpolation (there are other ways to implement this scheme). We know that given k points in the 2-dimensional plane $(x_1, y_1), ..., (x_k, y_k)$, with distinct x_is. There is one and only one polynomial $q(x)$ of degree $k-1$ such that $q(x_i) = y_i$ for all $i = 1, ..., k$. We denote the polynomial $q(x)$ of degree $k-1$ to be $a_0+a_1x+...+a_{k-1}x^{k-1}$. We set the secret to be a_0 which is the actual data that we want to send. In order to calculate a_0, we must know at least k points (which is named shadow) on this polynomial curve. Once we know the k points of polynomial $q(x)$, we can use the Lagrange Polynomial interpolation technique to calculate all the coefficients of $q(x)$. Then we can get a_0 by simply substituting $x = 0$, because $q(0) = a_0$. By doing that, we recover the secret. Compromising only $k-1$ of these points, the opponent still can't recover the secret a_0.

Lagrange polynomial: Lagrange polynomials are used in the Newton–Cotes method of numerical integration and in Shamir's secret sharing scheme in cryptography. Suppose we have a polynomial $q(x) = a_0 + a_1x + ... + a_{k-1}x^{k-1}$ of degree $k-1$. If we know k points on this polynomial curve, then we can calculate all the coefficients of the Lagrange basis polynomials:

$$q_j(x) = \prod_{0 \leq m \leq k \& m \neq j} \frac{x - x_m}{x_j - x_m} = \frac{x - x_0}{x_j - x_0} ..... \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} ..... \frac{x - x_k}{x_j - x_k}$$

where $q_j(x)$ is the polynomial that is calculated using all the points $(x_1, y_1), ..., (x_k, y_k)$, but not the point (x_j, y_j).

2.7 ID-based Cryptography with Private Key Generation (PKG)

In 1984 Shamir [41] proposed the idea of a public key encryption scheme in which the public key can be an arbitrary string. In the scheme there are four algorithms: (1) setup : generate global system parameters and a master-key, (2) extract : use the master-key to generate a private key corresponding to an arbitrary public key string $ID \in \{0,1\}^*$, (3) encrypt : encrypt messages using the public key ID, and (4) decrypt : decrypt messages using the corresponding private key.

Shamir's original motivation for identity-based encryption was to eliminate certificate management in e-mail systems. For example, when Alice sends mail to Bob at bob@company.com she simply encrypts her message using the public key string "bob@company.com". There is no need for Alice to obtain Bob's public key certificate. When Bob receives the encrypted mail he contacts a third party, which we call the Private Key Generator (PKG). Bob authenticates himself to the PKG in the same way he would authenticate himself to a CA in order to obtain his private key from the PKG. Bob can then read his e-mail. Note that unlike the existing secure e-mail infrastructure, Alice can send encrypted mail to Bob

even if Bob has not setup his public key certificate. The key escrow is inherent in the identity-based e-mail systems: the PKG knows Bob's private key. We discuss key revocation, as well as several new applications for Identity-based Encryption (IBE) schemes in the next section. Since the problem was posed in 1984, there have been several proposals for IBE schemes [15, 28, 38, 47, 46]. Figure 2.4 is a picture illustrating the working flow of IBE schemes with a private key generation center.

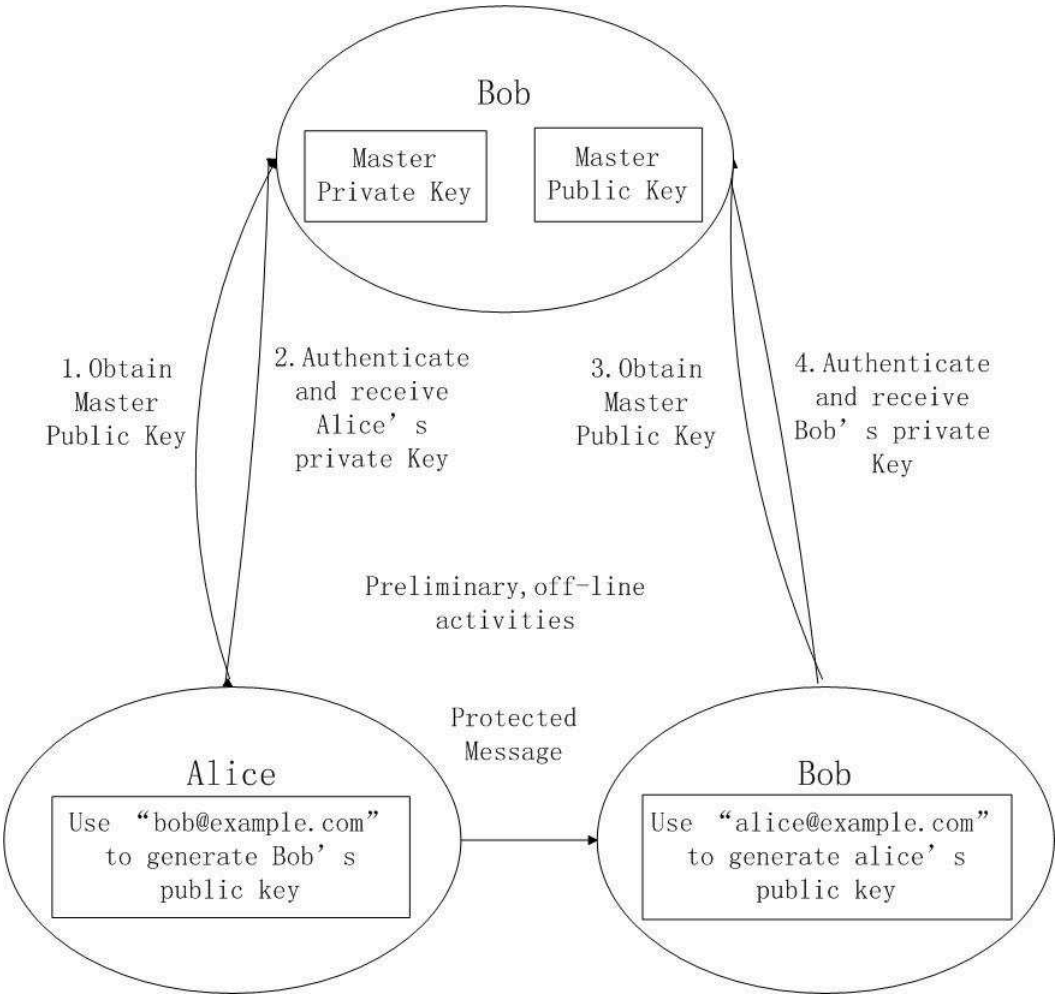

Figure 2.4: ID-based Cryptography with PKG

We can see from Figure 2.4 that there are a PKG and two users Bob and Alice in this IBE system example. The PKG has a master public key and a master private key pair which is used for message encryption and private key generation for each user, such as Bob and Alice in this example. The way Bob obtains his public and private key pair is that he authenticates with the PKG, just like the way a user authenticates with a CA. If Bob passes the authentication, he submits his ID and gets the corresponding ID-based public key and private key pair. He can also get public parameters such as the master public key which is used to encrypt messages along with his private key. In this ID-based encryption scheme, there is no need for any user to use a certificate. Any user can use anyone's ID, which is public, to encrypt messages.

However, none of the above IBE schemes had been fully satisfactory until 2001. Some solutions require that users not collude. Other solutions require the PKG to spend a long time for each private key generation request. Some solutions require tamper resistant hardware. In 2001, Boneh and Franklin proposed a fully functional IBE [5]. It is based on bilinear maps between groups. It uses a technique called Weil pairing, which has chosen ciphertext security in the random oracle model. Our key management and distribution scheme also uses Boneh and Franklin's IBE scheme.

2.8 Key Escrow Problem in ID-based Cryptography and Countermeasures

A serious disadvantage with the IBE scheme is that the PKG knows the private keys of all users. For example, PKG knows Bob's private keys. It might abuse this power. This is called key escrow, which is inherent in ID-based systems. Clearly, key escrow is a serious problem for some applications. This is unlike a traditional public key infrastructure (PKI), where the CA only issues certificates on user generated public keys, but does not know the corresponding secret keys. While most people find it uncomfortable that a mala fide Key Generation Center(KGC) can sign any message on their behalf, one should be aware that the same type of fraud is possible in the public-key setting as well. Since the certificate is usually sent along with the signature, a cheating CA can also generate a fake certificate for a public key of which it knows the corresponding secret key, thereby creating a valid signature. The victim could try to prove he's innocent by showing his real certificate to a judge, but nothing prevents the CA from claiming that the user registered two different public keys. The escrow property is therefore not so much an issue for signatures as it is for encryption, where a mala fide KGC can actually decrypt ciphertexts intended for any of its users.

2.8.1 Distributed Solutions

Several researchers have provided solutions to mitigate the key escrow problem in IBE. There are distributed key generation center (KGC) approaches which use a (t, n) threshold cryptography to limit a single KGC power. We know that in

a traditional model, the system just has one off-line KGC, to run IBE schemes. The network has a public/private key pair, called master key $< PK, SK >$, which is used to provide key generation service to all the nodes in the network. In the distributed model, there are multiple off-line KGCs. The master public key is generated by the key generation center in such a manner that the master PK is well known to all the nodes in the network. The master private key SK is shared by all the KGCs in a (t, n) threshold fashion. To be specific, each of the KGCs only has a secret share of the master private key SK, and no one is able to reconstruct the master private key based on its own information. To recover a master private key, a KGC needs at least any other $t-1$ KGCs agree to collaboratively send their valid partial share. Any KGC with less than $t-1$ collaborators will not be able to construct the master private key, thus not be able to answer the private key request for each node. From this perspective, we know that a single KGC is not able to abuse his power. Furthermore, the extraction of each node's private key is also in a (t, n) threshold fashion. Each of the k PKG service nodes generates a secret share of a new private key SK and sends it to the each requesting node. To be precise, the process of generating a share of the new secret key SK can be represented by function $SK_i = f_{extract}(S_i, ID) = S_i Q_{ID}$ where $S_i (i = 1, ..., k)$ is the share of the master private key of the KGC, ID is the identity of the requesting node, Q_{ID} is its public key, and SK_i denotes the generated private key share for the requesting node. By collecting the k shares of its new private key, the requesting node would compute its own private key $SK = \sum_{i=1}^{k} S_i Q_{ID}$. The KGC does not know the node's private key either.

Deng's paper "Threshold and Identity-based Key Management and Authentication for Wireless Ad Hoc Networks" [14] published in 2004, extends the above

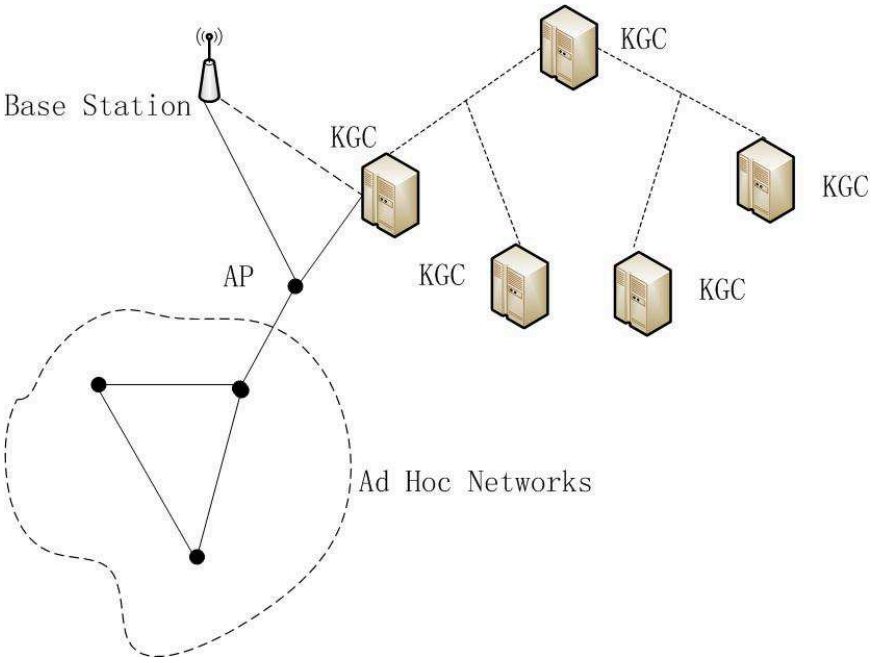

Figure 2.5: Sensor Networks with Distributed Key Generation Centers

idea to a fully distributed manner where every node in the network acts as the distributed KGCs and can provide the key services to other nodes. This means that each node knows the master public key and owns a partial share of the master secret key. Any t of the node can reconstruct the master secret key jointly which will be used to extract the secret key for each node. Any node's private key extraction can be make from at least $t-1$ nodes in the network. This scheme is an extreme example of how the key escrow problem of the IBE scheme can be solved. This solution is not feasible in our ad hoc sensor networks model for two reasons. First, for sensor networks to work, there are fixed points such as base stations that aggregates all the sensed data from the sensors, and can act as KGCs. A fully distributed protocol is not necessary and it does fit in this model. Second, the use of the (t, n) threshold scheme to construct a master private key and also each node's private key incurs lots of message traffic. This is infeasible

in sensor networks where networks have limited bandwidth and nodes that have limited battery life.

2.8.2 Partial Identity Solutions

There are other options aside from the distributed KGCs to mitigate the key escrow problem. A secure key issuing protocol is proposed in 2004 in which a private key is issued by the key generation center (KGC) and then its privacy is protected by multiple key privacy authorities (KPAs) [35]. In this protocol, a secure channel is provided by using a simple blinding technique in pairing-based cryptography. Only a legitimate user who has the secure blinding parameter can retrieve its private key from the protocol. In this proposal [35], the author introduces a single KGC and multiple KPAs. Before issuing the public key and partial private key to the user, the KGC checks the identification of the user. The public key is computed by hashing the user's ID with some information from the KGC and all the KPAs. The user node incorporates some secret information which is a binding factor, and sends the binding factor to KGC when requesting the partial key from it, and the partial key is then sent in a secure way that the partial key is timed with the blinding factor. Multiple KPAs sequentially provide key privacy services to user's private key by issuing their signature in a blinded manner. Finally, the user tries to get a private key of the ID-based cryptography through an interactive protocol with the KGC and n KPAs.

We can see from Figure 2.6 that there are several KPAs and one single KGC. The user node first authenticates with the KGC just like the user authenticates with a certificate authority. The user node incorporates private information x and

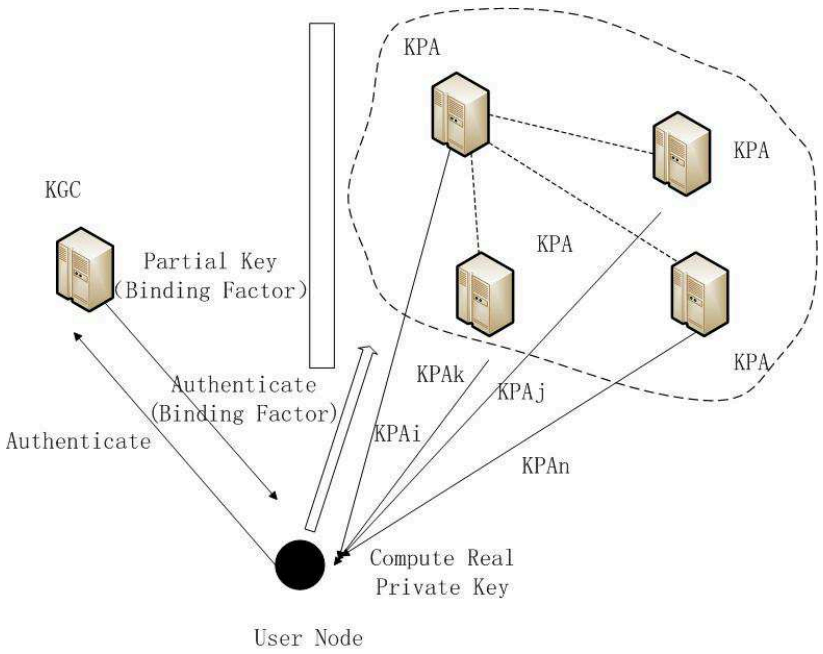

Figure 2.6: A single KGC with multiple KPAs to achieve identity anonymity

the KGC replies with a partial key which is secured by a binding factor, which only the user node has private information x can unbind it. Upon obtaining the partial key (note : from the partial key, no one can gain additional information about one's real private key), the user node uses such partial key to authenticate with multiple KPAs, and KPAs reply with partial information which is secured by the binding factor. The user node which has the private information x unbinds all the partial information from the KPA's and calculates the real private key from the partial information gathered from the KPA. The ID is calculated by incorporating all the partial information from KPAs which is unknown to KGC. So the user node remains anonymous to KGC and the real private key is also unknown to any entity. The protocol provides a way for the KGC and KPAs to authenticate the user node without knowing the user node's identity.

The key issuing process consists of the following three stages.

1 In the key issuing stage, a user sends his identity and blinding factor to the KGC and requests it to issue a partial private key. After checking the identity of the user, the KGC issues a partial private key to the user in a blinded manner.

2 In the key securing stage, the user requests multiple KPAs in a sequential manner to provide key privacy service, then the KPAs return the real private key in a blinded manner.

3 Finally, in the key retrieving stage, the user unblinds it to retrieve the real private key.

To be precise, the proposed secure key issuing protocol includes the following 5 stages; system setup, system public key setup, key issuing, key securing, and key retrieving stages.

Stage 1. System setup (by KGC)

As shown in the preliminaries section, the Bilinear Pairing subsection, the KGC specifies two groups G_1 and G_2 and the bilinear map $e : G_1 \times G_1 \rightarrow G_2$ between them, and three hash functions H_1, H_2, H_3. KGC also picks its master key $s_0 \in Z_q^*$ at random and computes its public key $P_0 = s_0 P$ where P is the generator of G_1. KGC publishes a description of the groups G_1, G_2, the bilinear map e, hash functions H_1, H_2, H_3, and the public key P_0.

Stage 2. System public key setup (by KPAs) The n KPAs [35] establish their key pairs. For all $i = 1, ..., n, KPA_i$ chooses his master key s_i and computes his public key $P_i = s_i P$. Then KPAs cooperate sequentially to compute the system public key,

$$Y = s_0 s_1 ... s_n P$$

which will be used as a system parameter in the group of users. More specifically,

$$KPA_1 \qquad computes \qquad Y_1' = s_1 P_0,$$

$$KPA_2 \qquad computes \qquad Y_2' = s_2 Y_1',$$

.....

$$KPA_n \qquad computes \qquad Y_n' = s_n Y_{n-1}'.$$

Then Y is published as the system public key. It will be used for encryption and verification by the users in the group. Note that the correctness of this sequential processes can be verified.

Stage 3. Key issuing (by KGC and user) A user with identity ID chooses a random secret x and computes a blinding factor $X = xP$. The user also requests the KGC to issue a partial private key by sending X and ID. Then the KGC issues a blinded partial private key as follows:

- Checks the identification of the user
- Computes the public key of the user as

$$Q_(ID) = H_1(ID, KGC, KPA_1, ..., KPA_n).$$

It is the result of hashing the user's ID, public information of KGC's and KPAs

- Computes a blinded partial private key as

$$Q'_0 = H_3(e(s_0X, P_0))s_0Q_{ID}.$$

  It is the result of hashing the user's ID, blinding factor and KGC's public key.

- KGC times its master key with the above Q'_0, and gets $s_0Q'_0$, which can be viewed as KGC's signature. Here $H_3(e(s_0X, P_0))$ is a blinding factor; a secure channel between the user and the KGC. The user can unblind it using his knowledge of x.

- sends Q'_0 and $s_0Q'_0$ to the user

Stage 4. Key securing (by user and KPAs) The user requests $KPA_i(i = 1, ..., n)$ sequentially to provide key privacy service by sending ID, X, Q'_{i-1}, and $s_0Q'_{i-1}$, then KPA_i.

- Verifies the $s_0Q'_{i-1}$ using the Bilinear Pairing property.

- Sends Q'_i and $s_0Q'_i$ to the user.

He repeats this process from KPA_1 to KPA_n and receives Q'_n. This means that the user really is the one who has the secret x, is authenticated and gets the partial key from the KGC. Finally, it can get the Q'_n which is the joint result from KPAs.

Stage 5. Key retrieving (by user) The user retrieves his private key D_{ID} by unblinding Q'_n as follows.

$$D_{ID} = \frac{Q'_{ID}}{H_3(e(P_0, P_0)^x)...H_3(e(P_n, P_n)^x)}$$

$$= s_0 s_1 ... s_n Q_{ID}$$

This means that the user who gets the joint result from KPAs can get the real private key. The private key D_{ID} is unknown to KGC because KGC does not generate this. This is also unknown to the KPA's because the KPA does not have the users secret x and without that, the KPA cannot derive the real secret key corresponding to the public key Q_{ID}. So the key escrow problem is solved.

In another dimension, which is different from distributed KGCs, people try to resolve this issue by hiding the node's ID from the KGC. A recent interesting paper [13] by Sherman proposes a key issuing protocol in which the KGC does not know the intended recipient of the ciphertext. They propose a new system architecture with an anonymous private key generation protocol such that the KGC can issue a private key to an authenticated user without knowing the list of users identities. The architecture is as follows:

From Figure 2.7, we can see that there are two components in the key issuing process: the Identity Certifying Authority and the Key Generation Center. From a high level, the user authenticates with the ICA. After authentication, the ICA is responsible for issuing each user a certificate based on its identity. This certificate is generated using the master certifying key sk_{cert}. The ICA is not responsible for generating the private key for the user. After the user gets the certificate from the ICA, the KGC issues private keys. The user requests the private key from the KGC by presenting the certificate obtained from the ICA. The certificate has no information about the user's identity, so the KGC never gets to know the identity involved in the certificate. The user private key is still generated with the help of the master secret key, that is owned by the KGC and kept secret from the ICA.

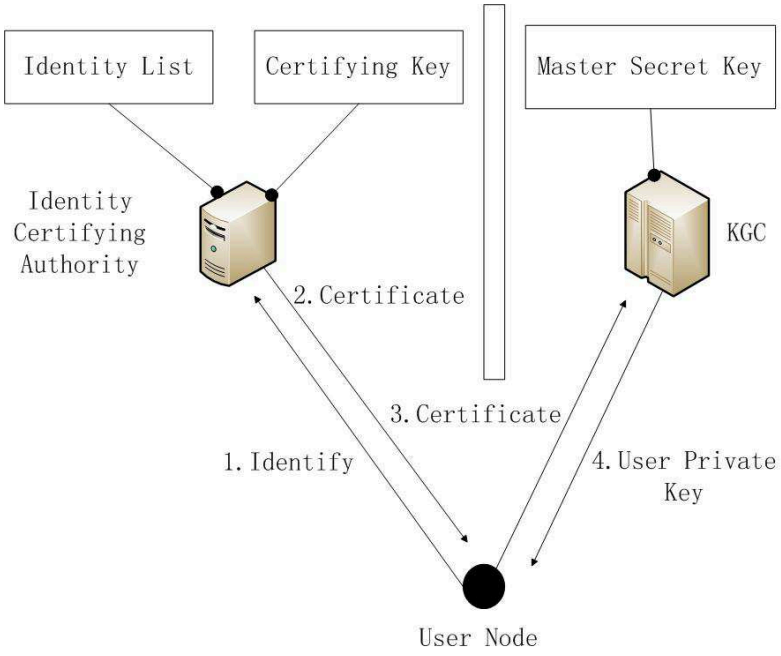

Figure 2.7: Anonymous Private Key Issuing Protocol Architecture

Figure 2.7 depicts the certification and the key issuing process. Since the ICA keeps the identity list of the system's users, it is believed that the ICA does not collude with the KGC (or the KGC can get the identity list easily). As in PKI, we also assume that the ICA would not impersonate any user. The solution requires a user to contact two parties before getting a key. Nevertheless, it may be cost prohibitive to have a globally available KGC to authenticate users and issue keys to users via secure channels in a typical ID-based cryptosystem.

2.8.3 Private Key Anonymity Solutions

Hiding the private key from the KGC is another option to mitigate the key escrow problem in IBE. KGC is still the entity that generates the private key for users. It appears to be contradictory that the KGC does not know the private key that he

generates. There are multiple private keys for a corresponding public key in this system. The number of private keys is so large that the probability for a KGC to pick up the same private key as the user's private key is negligible.

Goyal [22] introduced the notion of Accountable Authority Identity Based Encryption (A-IBE) as a approach to mitigate the (inherent) key escrow problem in identity based encryption schemes. The idea is to restrict the PKG's misbehaviour in such a way that if the KGC maliciously generates and distributes a decryption key for an Identity, it is highly likely that this misbehave action will be caught. In contract to the distributed KGCs, this approach does not require multiple key generation authorities. The new approach can be described as follows:

- In the IBE scheme, there will be an exponential (or super-polynomial) number of possible decryption keys corresponding to every identity ID.
- Given one decryption key for an identity, it is intractable to find any other.
- Users get the decryption key corresponding to his identity from the PKG using a secure key generation protocol. The protocol allows the user to obtain a single decryption key d_{ID} for his identity without letting the PKG know which key it obtained.
- Now if the PKG generates a decryption key d'_{ID} for the identity for malicious usage, with all but negligible probability, it will be different from the key d_{ID} which the user obtained. Hence the key pair (d_{ID}, d'_{ID}) is a cryptographic proof of malicious behavior of the PKG (since in normal circumstances, only one key per identity should be in circulation).

From this protocol, we know that the KGC does not know which private key

is selected from the private key pool and which user uses what private key. From one private key, others cannot gain additional information about other private keys which are in the same decryption key family. So if a KGC wants to abuse its power and decrypt the ciphertext of some users, it chooses a decryption key d'_{ID} from decryption key family of exponential size. The number is so large that the probability of selecting the same decryption key is negligible. So $d_{ID} \neq d'_{ID}$. The knowledge of the key d'_{ID} enables an entity E together with the honest user whose private key is d_{ID} with identity ID to sue PKG by presenting the pair $(d_{ID} \neq d'_{ID})$ as proof of fraud. The KGC could be shut down. This means that if the PKG ever uses a decryption key for an identity to decrypt some user's ciphertext, it runs the risk of being caught and shut down. This idea is borrowed from a regular PKI. In a regular PKI,a user will go to a CA and get a certificate binding his identity with his public key. The CA could generate one more certificate binding a malicious public key to his identity. However, two certificates corresponding to the same identity constitutes a cryptographic proof of fraud. PKG is free to generate one more decryption key for his identity. However, two decryption keys corresponding to the same identity constitutes a proof of fraud. PKG is free to generate one more decryption key for his identity. Two encryption keys corresponding to the same identity constitute a proof of fraud. This scheme leads to a construction of a black box A-IBE in a weak model, and the construction of full black-box A-IBE scheme is still not yet achieved.

Black-box Accountable Authority Identity based Encryption. The concept of accountable authority identity based encryption originates from the black-box traitor tracing in broadcast encryption. It is based on the idea of accountable-IBE, and takes the accountable-IBE one step further. In order to

understand the accountable authority identity based encryption, we should first have a look at what the traitor tracing in broadcasting encryption is. In broadcast encryption, there may be multiple decryption keys corresponding to one encryption key. In other words, there are multiple authorized entities able to decrypt a ciphertext. An example of such scheme is the Pay-per-view or subscription based television broadcasts. Usually, Pay-per-view or subscription based television broadcast schemes take on the following general form: a key supplier generates a set of meta-keys and assigns a subset of these meta-keys to each authorized user, for example m keys person user. These m keys jointly form the user's personal key which can be used to decrypt the encrypted message. Different users may have a non-empty meta-key intersection. The form of the message in a traitor tracing scheme is $(enablingblock, cipherblock)$. The cipher block is the actual date encrypted under symmetric key. The enabling block is the meta-key values encrypted under some keys. Every authorized user will be able to decrypt the values and use these values to compute its personal key. Given the personal key, it can decrypt the actual data in the enabling block. A high level logic view of such a scheme is depicted in Figure 2.8:

By taking advantage of multiple decryption keys, a traitor may conspire and give an unauthorized user (or users) a subset of their keys. Given these keys, the unauthorized user is able to decrypt the ciphertext of other users. In a traitor tracing scheme, when a pirate decoder is captured, it should be possible for the judge to trace the source of the malicious action, and detect at least one traitor (The scheme assumes that k traitors can collude to release private keys).

As explained, the scheme considers the scenario when a PKG generates and tries to distribute a decryption key corresponding to an identity. A-IBE specifically

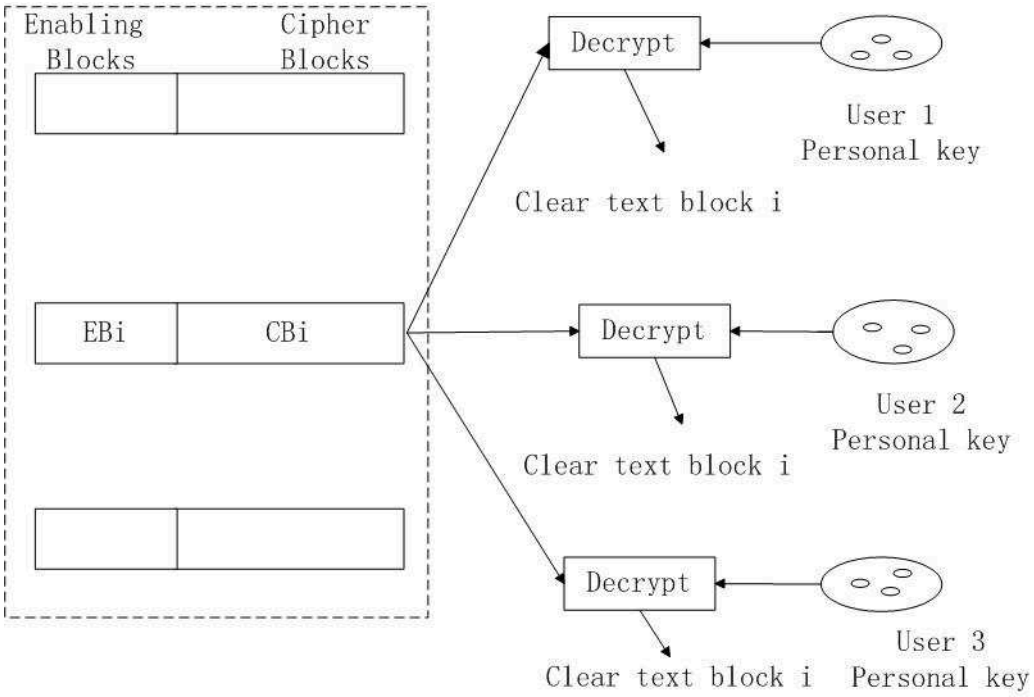

Figure 2.8: A high-level view of the traitor tracing scheme

assumes that the key is a well-formed decryption key. However, one can imagine a scenario where the PKG constructs a malformed decryption key which, when used in conjunction with some other decryption process, is still able to decrypt the ciphertexts. In extreme cases, there could be a black box (using an unknown key and algorithm) which is able to decrypt the ciphertexts. Given such a box, a third party (such as the court of law), possibly with the cooperation of the PKG and the user, can trace the box back to its source. It should be able to determine whether it was the PKG or the user who was involved in the creation of the black box. Such a system is called a black-box accountable authority identity based encryption system. This is a natural extension of the A-IBE concept and is related to the concept of black-box traitor tracing in broadcast encryption. The black-box A-IBE is an ideal requirement for countable Authority Identity based

Encryption.

The paper only constructs an accountable authority IBE in a weak model with the assumption that the PKG cannot maliciously distribute a well formed decryption key in which a "white box" guarantees and are completely insufficient in practice.

Goyal proposed a full black-box accountable authority identity based encryption [23] in 2008 and resolves the main open question left in his previous work. The main technical difficulty is resolving the tension between the information being leaked as part of the decryption queries and the success of the exoneration procedure. On one hand we require that during regular operation, the outcome of the decryption of a ciphertext should not leak information about which decryption key the user selects. On the other hand, during exoneration, a judge should be able to extract enough information about the user key selection from the black box in order to determine that the user could not have generated the box (and therefore the PKG must be at fault).

The key idea in this construction is to first design a scheme having imperfect completeness. That is, for every possible decryption key, there exist a negligible fraction of (valid) ciphertexts which cannot be decrypted by this key. On one hand, this property is helpful in tracing: a judge (given the decryption box and the decryption key of the user) can probe the box exactly on those ciphertexts which the user key should not be able to decrypt. On the other hand, this does not seem to create a problem for decryption queries since the chance that a malicious PKG will hit such a ciphertext (with a polynomial number of queries) is negligible.

The scheme uses ideas from key-policy attribute-based encryption (KP-ABE) [24, 44]. Very roughly, they label each ciphertext as well as a decryption key with

a list of dummy attributes. Then a policy is used to decide whether or not a ciphertext will be decrypted by a particular private key. To achieve statistical completeness, for every decryption key, all but a negligible fraction of ciphertexts will satisfy this policy.

While the approach of constructing such an AIBE scheme has imperfect completeness, a "complementary" system in parallel is run with such a scheme so that the resulting system also achieves the property of perfect completeness (while also maintaining the functionality of the tracing procedure).

2.9 ID-based Signature Schemes from Bilinear Pairings

In traditional Public-key infrastructure(PKI), the signer of the public key is essentially a random bit string picked from a given set. How to associate the public key with the signer becomes a problem. The digital certificate is then used to bind the public key and identity of the signer in PKI. However, this digital certificate may not be efficient since a fixed authority certificate authority is needed and should always be online to issue certificates and maintain an up-to-date depository for all the valid public key and identity pairs, or the whole network is broken. As noticed by Shamir, it would be more efficient if there is no such association needed. To be precise, if the network does not need a certificate authority to issue the digital certificate, then the users identity would be their public key. Actually, in terms of implementation, the public key would be directly derived from the public key using some one way hash function. After that, many researchers has worked on

ID-based encryption and its corresponding signature schemes.

Generally speaking, identity based signature schemes are always based on a hard problem, and there are usually two well-known hard problems that are commonly used in the world of cryptography. The first one is the large prime factorization problem and the other is the discrete logarithm problem. At first, the identity based signature schemes are usually RSA-based and rely on the difficulty of factorization integers. There are not any ID-based signature schemes based on discrete logarithm problem as efficient as RSA-based signature schemes until 2003 when Boneh and Franklin [5] proposed their ID-based signature schemes which are based on the difficulty of the discrete logarithm problem. They implement their ID-based signature schemes using the bilinear pairings technique, which actually is a discrete logarithm problem based signature scheme.

Bilinear Pairings was thought of a bad thing in cryptography at first, because the discrete logarithm problem in supersingular elliptic curves was reducible to that in a finite field using the bilinear pairings which causes the supersingular elliptic curves to be excluded for use from the world of cryptography. Until 2000, Joux [33] proposed a tripartite Diffie-Hellman based using the Bilinear Pairings in supersingular elliptic curves. It turned the bilinear pairings into a useful technique for cryptography use. After that the a number of ID-based signature schemes using bilinear pairings are proposed which will be talked about below and become the mainstream in the ID-based signature scheme world. In the next section, we will discuss about various kinds of ID-based signature schemes from bilinear pairings and Yi's [50] ID-based signature scheme that is used in our key distribution scheme is also one of them.

2.9.1 Convert ID-based Encryption Schemes to Signature Schemes

The Id-based encryption schemes can be directly converted to signature schemes. In 2003, Boneh and Franklin gave a generic method to convert any Identity-Based Encryption scheme into a signature scheme. The public key of the signature scheme is the global parameters of the IBE scheme. The signature scheme that is converted from the Identity-Based Encryption scheme using Boneh's method is not ID-based signature scheme. The public key is not a user's ID, but some public information and sometimes even composite data. So the signature schemes we talked about in this section are not ID-based signature schemes, though they are converted from ID-based encryption schemes. In other words, the public key in those ID-based encryption schemes is their ID, however, if you want use the signature schemes that are converted from these ID-based encryption schemes, the public key becomes some other public information and the private key is also different. We still want to talk about schemes because we want people to notice this fact so that the misconception is excluded.

Boneh proposed a Secure Signature Scheme from Bilinear Pairings [6]. This signature scheme is based on the difficulty of discrete logarithm problem and its efficiency is close to the RSA-based signature schemes. To be specific, the scheme is based on the Computational Diffie-Hellman problem. It is proved secure against existential forgery under adaptively chosen message. Actually, all the ID-based Signature from Bilinear Pairings are based on the bilinear pairings property. The way they prove the security of these Signature from Bilinear Pairings is to prove that the Bilinear Pairings problem is isomorphic to the Computational Diffie-

Hellman problem. The methods they use to prove the bilinear pairings problem is isomorphic to the Computational Diffie-Hellman problem. First, is the assumption that CDH can be solved in time t with probability at least ε , this assumption leads to the corollary that the Bilinear Pairings is not (t, ε)-secure. Second, if the assumption that the Bilinear Pairings problem can be solved in time t with probability at least ε, this assumption leads to the corollary that the CDH problem can be solved. By the above steps, we can get the conclusion that the Bilinear Pairings problem is isomorphic to the Computational Diffie-Hellman problem. So the Bilinear Pairings problem is as hard as the Computational Diffie-Hellman problem.

Boneh's signature scheme is based on signature authentication tree with a large branching factor. From a high level perspective we can see from Figure 2.9, in the tree structure, there are nodes, parents and leaf. The nodes x_i are some value calculated by picking some random value and hashing it using the bilinear pairings. f_i is the authentication value of x_i. Each nodes in the tree is authenticated with respect its parent. The signing procedure of the message is that the signer generates a path from the root to the leaf of the authentication tree, and the path is the list of operations to do on a message. The leaves are selected in sequential order and each message has a different leaf. Finally the signature is a sequence of values of f_i. The signature is accepted if and only if $\widehat{x_0}$ matches the root of the tree where $\widehat{x_0}$ is the last node in the node list of a bottom-up order.

In 2005, Waters [49] proposed a signature scheme which also use the bilinear pairings. The scheme assumes a standard oracle model which is more secure than the one assumed in a random oracle model. arguments involved in this signature is similar to the one mentioned in Yi's. Actually, most of the signature schemes

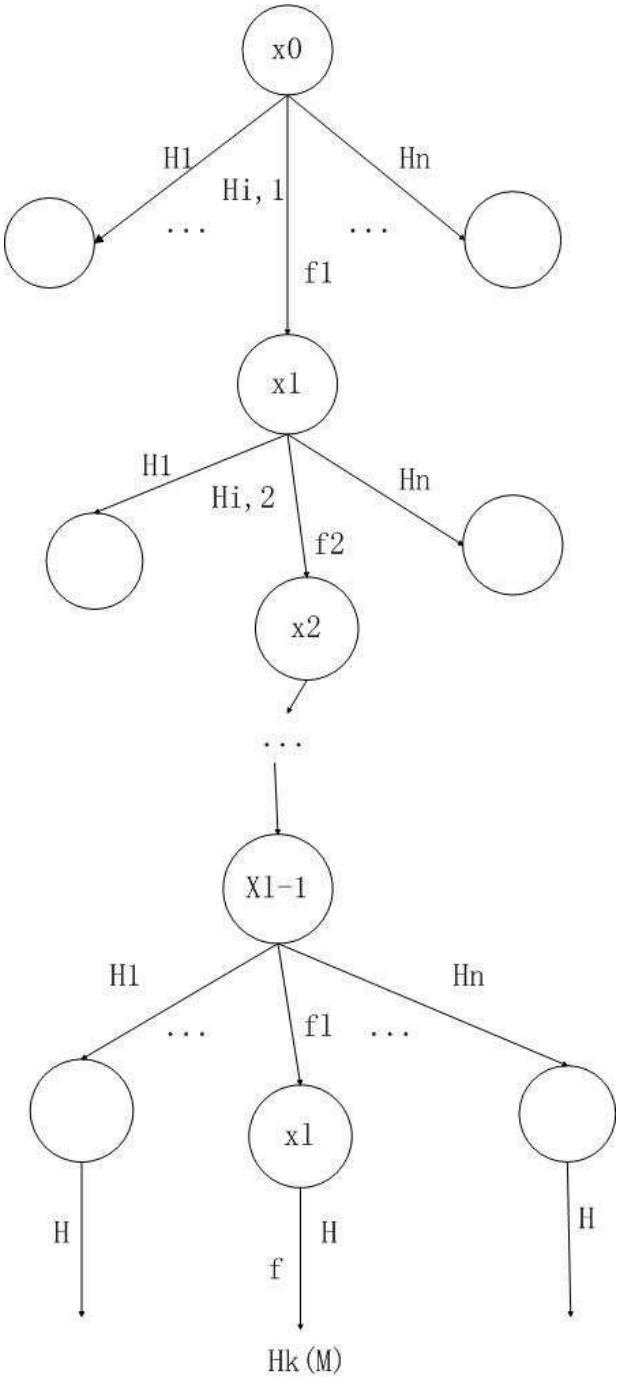

Figure 2.9: Authentication tree

converted from ID-based encryption schemes using bilinear pairings have similar arguments. But they are different in their public key, private key derivation and signing procedure. The public key is not actually a user's ID, but some public information and sometimes even composite data. Waters' signature scheme is an example. The public key in [49] is (g_1, g_2, u', U) where $g_1 = g_a$ and a is some random prime number. $g_2 \in G$ where G is a cyclic group. u' is also random selected from G. U is a list of random vectors that are selected from G. The secret key is g_2^a. After the signing procedure, the signature in this scheme is a list of vectors. So these signature schemes that are converted from ID-based encryption schemes using Boneh's generic transformation method can't be used in our key distribution scheme.

2.9.2 ID-based Signature Scheme

Boneh's transformation produces ordinary signature schemes from ID-based encryption schemes. In 2010, Zhang [52] proposed a ID-based signature with strong unforgeability. The work is based on Waters signature scheme [49]. This signature scheme also uses the Bilinear Pairings property and is secure against Existential Unforgeability under adaptive Chosen Message Attack. Just as ordinary signature schemes, the signing procedures consists of four steps: Setup, Key Extract, Signing, Verification. The public key is user's $ID \in \{ 0, 1 \}^*$ and the private key is a pair (d_{ID}, Y_{ID}) where $d_{ID} = (d_1, d_2) \in G_1^2$. d_1 is derived by a list of vectors of a fixed length that belongs to the cyclic group G_1 and $d_2 = g^r$, where r is some random prime number and g is the generator of cyclic group G_1. So we can see the private key here is a composite data. The signature is the message M that is

signed by the private key d_{ID}. After the signing procedure, signature on message M is $\sigma = (S_1, S_2, S_3, S_4)$ Where S_1, S_2, S_3, S_4 are four different numbers generated by incorporating message M, private key d_{ID}, a random prime number k and the generator g. However, this ID-based scheme is not compatible with the ID-based encryption scheme by Boneh that is used in our key distribution scheme since the private key here is a composite data that belongs to the set G_1^2 which is different from the private key,a point in the cyclic group G_1, in Boneh's IBE scheme. Yi's ID-based signature scheme we use in our key distribution scheme is compatible to the Boneh's IBE scheme. To be precise, the public key and private key is the same as Boneh's IBE scheme. Yi's ID-based signature scheme's logic is similar to that of Zhang's [52]. For a detailed description of Yi's ID-based signature scheme, please refer to our preliminary section.

2.9.3 ID-based Group Signature Scheme

The group signature scheme was first proposed by Chaum and van Heijst [9] in 1991. The motivation is that by using the group signature, any member from a group can sign the signature on behalf of the group. Anyone can verify the signature with the group public key, but the verifier does not know the identity of the members in the group, he can only know that the signer is coming from the group. The identity of the signer of a signature can be known if necessary, for example, if the signer has some malicious action, an entity called group manager can tell the identity of the malicious signer, and take countermeasures towards the malicious action. A use case of group signature is that suppose you are in an organization and everyone belongs in a team that has access to the printers which

are connected to the network within their department. Only the members on the team can use the printers, but for some privacy concerns, the user's identity needs to be kept a secret. However, if some users use the printer too much and print pages exceeding the limit, the company should know the identity of the user and charge the individual extra fees.

In 1997, Park [42] proposed the first ID-based group signature scheme. After that a number of ID-based group signature schemes [7, 10] came out. A recent ID-based Group Signature Schemes [11] is proposed by Chen. It uses the Bilinear Pairing property. The idea is that there is a probabilistic algorithm that generates a group public key and a secret key of the group manager. The group manager is responsible for the group member joining. After becoming a group member, the user has a membership certificate and a membership secret. The signature is generated by the probabilistic algorithm which takes the input group public key, a membership certificate, a membership secret and a message M. Different from the ordinary signing procedures which consist of four steps, the ID-based group signature scheme has one more step. Open is used to trace the identity of a malicious signer if necessary. In the ID-based encryption scheme model, the PKG is the group manager and is the only one that can trace the identity of a malicious signer.

2.9.4 Hierarchical ID-based Signature Scheme

The first Hierarchical ID-Based signature scheme was proposed by Gentry [19] in 2002. In this model, PKGs are organized by levels and the users are also by levels. In other words, the users and PKGs are organized in a hierarchical manner.

The motivation of hierarchical ID-based signature is that instead of using a single PKG in IBE, there is root PGK and low level PKGs. The root PGK can delegate the private key generation task to low level PKGs so that the burden of a single PKG is released. The scheme has a property that disclosing the secret of low level PKGs does not affect the high level PKGs and the secret of the high level PKGs is still kept. The hierarchical ID-based signature scheme also comes from the bilinear pairings property. The idea is that the root PKG only generates the private keys for low level PKGs. The low level PKGs are the entities that generate the real private keys of the user. The low level PKGs has the responsibility to authenticate the identity of the users before issuing the private keys to them. Thus, the workload is distributed to the low level PKGs instead of a single PKGs.

Unlike the ordinary signature signing procedure which has four steps, the Hierarchical ID-based Signature Scheme has five steps: Root Setup, Lower-level Setup, Extraction,Signing, Verification. The Setup step in ordinary signature signing procedure actually splits into Root Setup and Lower-level Setup. The identity of the user is also different from the ordinary user's ID which is a single value. The user ID in hierarchical ID-based signature scheme is its position in the hierarchy which is denoted its tuple of IDs: $(ID_1, ID_2, ..., ID_t)$, its signature is a list of secret points which belong to the cyclic group G_1. The signature size is dependent on the depth of the user. The user's ancestors in the hierarchy tree are the root PKG and the users/lower-level PKGs. Its security is proved under the random oracle, which is a weaker model. After that a number of hierarchical ID-based signature scheme are proposed. In 2009, Zhang [53] proposed a short Hierarchical Identity-based signature in the standard oracle model. They reduce the private key size and the signature size is constant, consisting of only three group elements, which

is independent of the depth of the user.

2.9.5 ID-based Threshold Signature Scheme

Baek [2] proposed the first ID-based threshold signature scheme in 2004. We know the ID-based signature scheme eliminates the issues of digital certificates among users. So the certificate authority is not needed and the users can just use another public ID as the public key to encrypt messages. Besides this reason, the ID-based Threshold Signature Scheme can decentralize the power to sign a message. A use case of such a scheme would be an executive that wants to delegate his power to several department managers, such as accessing the company's key server room which contains sensitive data. If the executive is off-site, by using the threshold signature scheme, a number of managers can jointly generate the signature and pass the authentication. Baek's ID-based threshold signature scheme also comes from bilinear pairings, and is secure against chosen message attacks in the random oracle.

Quite similar to the logic of the threshold cryptography, in order to generate a valid signature and pass the authentication, a user needs at least a number of valid partial signatures from other users. In this model, there are signature generation servers. The private key associated with an identity of a user is shared among those signature generation servers. This is achieved by the using the "Secret-Sharing over g" method, where g is the generator of the cyclic group G. To be precise, we know that the private key of an IBE is a point in G. So Baek uses a technique to share a point in G which is the lagrange interpolation technique that is used in Shamir's secret sharing scheme. This scheme distributes the role of the

signature signing, however, it also incurs a lot of message traffic because a number of signature generation servers are needed to jointly generate a valid signature.

2.9.6 ID-based Mediated Signature Scheme

How to revoke a user's identity in identity-based cryptosystems is an important topic. Boneh introduced a method to revoke a user's public key based on RAS hard problem. The scheme is named mediated RSA [4]. In order to revoke the identity of a user, a special online entity, named a security mediator is proposed. The working logic is as follows: For example, if Alice wants to send a message to Bob, Alice must first connect to the security mediator to obtain a token. Only by using this token, can Alice sign a signature or decrypt a message that is encrypted using the public key. If Alice is detected to be some malicious user, the revocation procedure of Alice's identity, also its public key, is that an administrator contact to the security mediator to stop issuing the token to Alice, thus Alice can't sign or decrypt any messages. In other words, Alice's ability to sign signatures or decrypt messages is revoked. The early way to implement that idea is this a private usually has a valid period, if the private key expires and the PKG stops to issue a new private key to the user, then the user's public key is revoked. However, this idea has drawbacks because the user's identity can't be revoked immediately. And the PKG should always be on-line to periodically re-issue all private keys in the system which incurs a lot of traffic.

In 2006, Cheng [12] proposed an ID-based mediated signature scheme that can revoke the user's public key immediately. The main idea behind is to introduce a trusted on-line entity security mediator (SEM). To achieve immediate revocation,

the private key is split in two parts by the PKG. The PKG gives one part to the user, and the other part is kept in the SEM. If the PKG wants to immediately revoke the user's ability to sign signatures or decrypt messages, he can just instruct the SEM to stop issuing the part of the private key to the user. This scheme also comes from bilinear pairings. It is proved secure against existential forgery under adaptively chosen message and ID attacks in the random oracle model. In 2011, Cheng proposed another ID-based mediated signature scheme without trusted PKG [48], he improved his previous ID-based mediated signature scheme. This scheme assumes that the PKG might be dishonest and impersonate users to forge signatures because the PKG knows every node's private key. With additional modifications to the security mediator, the PKG is able to generate any valid signature in the improved ID-based Mediated Signature Scheme. For detailed explanations, please refer to his paper [48].

2.10 Main Methods for Key Management Schemes

For cryptosystems on ad hoc network, there are symmetric keys protocols and asymmetric keys protocols. Most schemes that have been proposed depend on certificate-based cryptography (CBC) and ID-based cryptography (IBC).

The key issue for cryptosystems in ad hoc networks is the distribution of the keys. Generally speaking, there are three types of ways to do key distribution.

1) Centralized Approaches: In centralized approaches, a designated entity (e.g., the group leader or a key server) is responsible for the calculation and distribution of the group key to all participants.

2) Distributed Key-Agreement Approaches: With distributed or contributory key-agreement protocols the group members cooperate to establish a group key. This improves the reliability of the overall system and reduces the bottlenecks in the network in comparison to the centralized approach.

3) Decentralized Approaches: The decentralized approach divides the multicast group into subgroups or clusters, each sub-group is managed by a LC (Local Controller) responsible for security management of members and its subgroup. Two kinds of decentralized protocols are distinguished as static clustering and dynamic clustering.

2.11 Security Requirements for Cryptography Key Schemes

The goals of these cryptography secure methods or Security requirements are as follows:

1) Forward secrecy: In this case, users left the group should not have access to any future key. This ensures that a member cannot decrypt data after it leaves the group.

2) Backward secrecy: A new user who joins the session should not have access to any old key. This ensures that a member cannot decrypt data sent before it joins the group.

3) Non-group confidentiality: Users are never part of the group should not have access to any key that can decrypt any multicast data sent to the group.

4) Collusion freedom : Any set of fraudulent users should not be able to deduce the currently used key.

5) Trust relationship: In mobile ad hoc groups, there is no trusted central authority that is actively involved in the computation of a group key that all participants have equal rights during the computation process.

An ideal key distribution and management scheme in ad hoc networks should meet all the above security requirements. In practice, different schemes have their advantages and disadvantages, and seldom do schemes meet all the above requirements.

To meet the above requirements, no matter what specific encrypt or decrypt methods the secure protocols use, they all need groups keys which are for the communication between CHs and pairwise keys which are for the communication between CH and MN, and when the node leaves or joins or there is a malicious node detected, the protocols will need key joining, key eviction or key update operations.

Chapter 3

Related Work

3.1 Introduction

This chapter reviews the related work about key management schemes for ad hoc networks. There are two main methods to do key management. One way is to preload nodes with symmetric keys. The other way is to preload nodes with asymmetric keys. As sensor networks have limited resources, preloading nodes with symmetric keys works best. We also talk about the impersonation problems in some key management schemes for ad hoc sensor networks.

3.2 Key Distribution and Management Scheme in Ad Hoc Networks

In 2010, Li proposed a key distribution and management scheme [36] which secures the cluster-based ad hoc network with distributed authorities. In this cluster-based ad hoc networks, they assume there is a off-line key generation centers (KGC)

which issues the ID-based cryptography key pairs, the cluster heads (CHs) use (t, n) threshold cryptography to establish a group key which is shared by all the CH nodes. And the (t, n) threshold cryptography technique is also used to revoke a malicious node if at least t of the CH nodes agree to send a revocation accusation message towards that node. It consist of five phases: network initialization, key revocation, multiple secrets key update, member joining and eviction, and group key generation.

Min-Ho[41] proposed a key management for multiple multicast groups in wireless networks. The usual way to protect group communication is to use a group key or cluster key that is shared by cluster members. However, the updating of a group key may cause overhead. This happens when there are nodes leaving or joining. To reduce the overhead caused by the rekeying, group key management schemes are proposed. Most of them are aiming to secure communication within a single group, and are not suitable in multiple multicast group environments. Min-Ho's scheme is designed for multiple multicast group environments. It uses asymmetric keys such as master keys and slave keys. The master keys is used to distribute the cluster keys. The nice thing of this scheme is that only one slave key needs to be updated when there are nodes joining or leaving. The rest of the slave keys can still remain the same by only changing the master keys. The overhead caused by the group key rekeying is reduced compared to the schemes designed for single group environment.

K. Gomathi proposed an efficient cluster based key management scheme [21] for MANET with authentication in 2010. The network topology is a multicast group clustering. This paper proposes a new cluster based tree (CBT) algorithm for secure multicast key distribution, in which the source node uses Destination

Sequenced Distance Vector(DSDV) routing protocol to collects its 1 hop neighbors to form clusters. Clustering is dividing the multicast group into several sub-groups. Local controller (LC) manages each subgroup, which is responsible for local key management within the cluster. Thus, after the Join or Leave procedures, only members within the concerned cluster are affected by the re-keying process. The local dynamics of a cluster does not affect the other clusters of the group and hence it overcomes 1-affects-n phenomenon which means a change in cluster can cause changes in other clusters.

A verifiable dynamic threshold key management scheme based on bilinear pairing without a trusted party in mobile ad hoc network [39] is proposed in 2012 by Meng and Li. In this scheme, all participants must provide confidential and verifiable information to each other in order to verify. In the scheme, the main difference is that the master key of MANET is generated by all participants collectively which is different from current schemes generated by the Key Generation Center (KGC). Every participant can calculate the system's public key, but participants cannot recover the system's secret key without any other $t-1$ participants' help. The (t, n) threshold polynomial in this scheme is based on the elliptic curve cryptography (ECC) while the current mainstream threshold polynomial is based on prime-field.

Yu [51] in 2010 proposed a hierarchical identity based key management scheme in tactical mobile ad hoc networks. This scheme assumes the network has a hierarchical nature and is designed for military applications. The scheme is ID-based and it selects the nodes to be the PKG. The current mainstream key management scheme is focused on key distribution and considers the networks topology when distributing keys. So they have key update and revocation procedures. However, they do not take the node's state or attributes into account. This scheme considers

the node's attributes such as security conditions and energy states when selecting their PKGs. The way they update their keys is also different. Nodes can get their keys updated from their parent nodes or from their sibling nodes in a jointly manner which is implemented using the (t, n) threshold scheme technique. Similar schemes are [43, 18] where in [18] the nodes gets their keys updated from a subset of keys from their parents.

Saju [32] proposed a distributed hierarchical key management scheme for mobile ad hoc networks in 2010. Similarly to the above scheme, the author provides a clustering formation algorithm as well as a key distribution management scheme. This scheme uses an adaptive weight cluster (AWC) technique to choose the cluster head. The selection parameters are power level, connectivity and stability. The power level is measured using a centralized algorithm. Connectivity is counted by link disjoint and joint values. Stability values of the node are calculated based on the entropy of node distances. These combined factors help in the selection of a stable and powerful cluster head. The key management scheme is based on Wen's [26] self-contained public key-management scheme. The node in this scheme uses more than one public and private key pairs to encrypt and decrypt messages. The idea is that to save communication overhead for authentication, nodes are preloaded with a small number of public and private key pairs during deployment. For example, if node A wants to send a message to node B, A first requests B's ID to get a set of private keys of B. Then node A uses the public key set that corresponds to the private keys of node B to encrypt the message. The scheme has a key allocation algorithm to allocate these set of public and private keys. The goals of these key allocation algorithms are to minimize the memory need for storing these cryptographic keys while still maintaining a high level of security

resilience.

3.3 Key Distribution and Management Scheme in Ad Hoc Sensor Networks

People try to derive the different pairwise keys from the same initial keys which is preloaded by the KGC, and when people try to establish the pairwise keys from the preloaded initial keys, many of them hash the initial keys together with information that is public or ID that others can impersonate.

In 2004, a localized encryption and authentication protocol (LEAP) [57] was proposed by Sencun for large-scale distributed sensor networks. They use multiple symmetric keys to secure different types of communication messages. In this scheme, there is a group key which is a globally shared key used by the base station for encrypting messages that are broadcasted to the whole network. There is a cluster key which is shared by a node and its neighbors, and it is mainly used for securing locally broadcasted messages,e.g., routing control information, or securing sensor messages which can benefit from passive participation. A locally shared key is needed for in-network processing techniques such as data aggregation and passive participation which is important for saving energy in sensor networks. For example, for passive participation, if a node can overhear a neighbouring sensor node's message being transmitted and chooses not to transmit the same message, energy is saved. As for data aggregation operations, a node can also suppress its own reading if the reading is larger than an overheard one to save energy. All of the above cannot happened without a locally shared key: cluster key. There is also

a pairwise key shared by two nodes, and the way the pairwise key is established is to hash the initial key with each node's ID. The impersonation problem occurs because in the establishment of pairwise keys, any inside node which is preloaded with the initial key can declare any node's ID and can establishe pairwise keys of any node pair. No method can be used to stop nodes from establishing another nodes pair of pairwise keys.

[57] tries to prevent an attacker from deriving the master keys and pairwise keys of the nodes that were deployed earlier or will be deployed late. They preload different initial keys for different time intervals. To be precise, the time is divided in to m intervals $T_1, T_2, ..., T_m$, and the network controller randomly generates m keys: $K_1, K_2, ..., K_m$. So a node u deployed in time interval T_i is loaded with the initial key K_i, from which it derives its master key $K_i(u) = f_{K_i}(u)$ for T_i and this master key is then used to establish pairwise keys with other nodes. By doing that, an attacker compromising a new node u deployed in T_i can't know any pairwise keys that the other nodes established in the current interval time. or later time intervals because node u does not know any K_j that is not in the present time intervals. However, that still does not solve the impersonation problems because an inside node can still declare any ID and derive any other's master key, thus establishing any node pair's pairwise key for the present time intervals.

Huawei in 2009 published a paper entitled "Hiberarchy Cluster Model and Key Management in Wireless Sensor Network" [54]. The network model is different from the hierarchical ad hoc sensor network model in which clusters are formed by two levels: The first level is near the base station while the second level is far from it, and only sensors in the first level can be elected as cluster heads. Sensors except the head node in the cluster only communicate with their vicinal nodes and

the head node. The base station is in charge of the election of a head node, and broadcasts the election result in a cluster, which can avoid energy consumption of sensors while negotiating head node. The model is pictured as Figure 3.1:

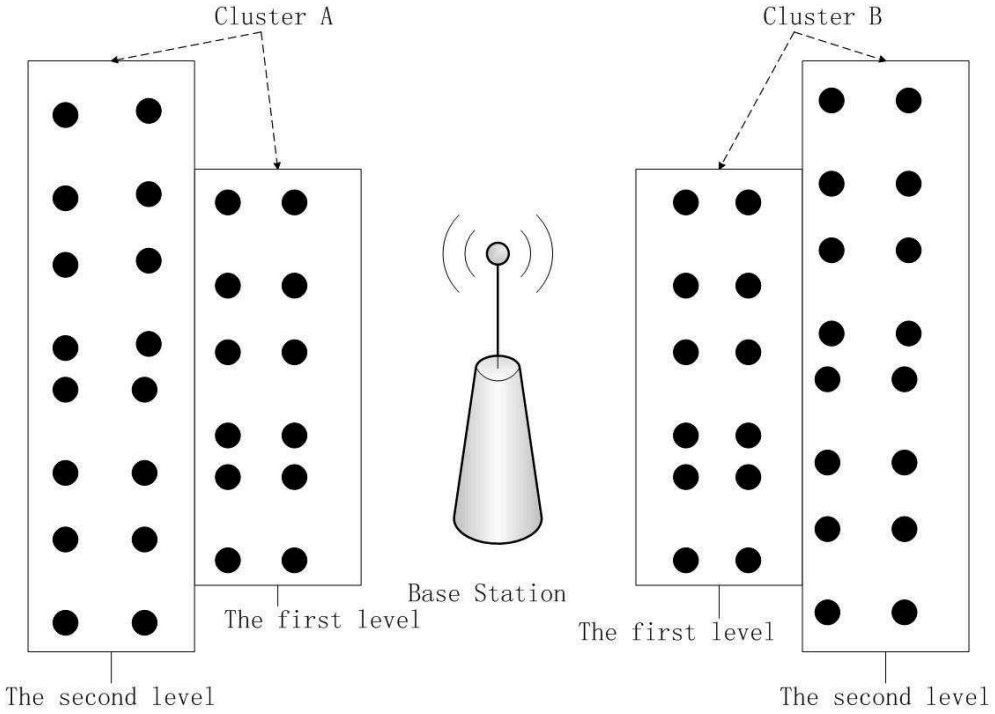

Figure 3.1: The hiberarchy cluster model

The way the pairwise key is established is that the KGC uses two hash functions $F_1(x)$ and $F_2(x)$ to compute two key chains: $\{k_1^1 : kid(1,1), k_2^1 : kid(1,2), ..., k_n^1 : kid(1,n)\}$ and $\{k_1^2 : kid(2,1), k_2^2 : kid(2,2)..., k_n^2 : kid(2,n)\}$ where kid is the ID of a key and n is the number of sensors in a sensor network. They choose a unique positive value $t \in [1,n]$ for each sensor, and pick two keys $k_t^1 : kid(1,t), k_t^2 : kid(2,t)$ from two keys chains to install each sensor as its preinstalled keys. When two nodes want to establish a pairwise key between them, they randomly choose one of the two keys and derive the pairwise key using two chosen keys. When a sensor establishes all pairwise keys with its vicinal nodes, it deletes their preinstalled keys

and the secret information seed e which is also installed during node deployment. " They use master key $K_{master} = F^e(x||y)$ to distribute cluster key where F is a one way hash function, e is a secret seed, (x, y) is the coordinate of a node. Every node generates his own master key by hashing the seed e and coordinate (x, y) and sends it to their cluster head. The cluster head generates a polynomial using these master keys, appends the cluster key with the polynomial and broadcasts the polynomial to its cluster member nodes. The polynomial is computed as follows : $g_u(x) = \prod_{i=1}^{n}(x - K^i_{master})$ and $m_u(x) = g_u(x) + K^u_{cluster}$, which is a product of x subtracting different master keys. Any node that has the master key can derive the cluster key. The nice property of this polynomial is that the cluster head encrypts and broadcasts the cluster key distribution message once while the above needs to encrypt and broadcast the cluster key distribution message several times. However, the drawback is the impersonation problem. We can see from the master key $K_{master} = F^e(x||y)$ that every node can declare any other node's position or even any position he wants and there is no way to authenticate this. Even worse, every inside node can derive any other's master key and get the cluster key of another cluster. This makes the cluster key a global key.

Chapter 4

Preliminaries

4.1 Introduction

This chapter presents the details of the two techniques we use for our key distribution and management scheme. Boneh-Franklin's [4] IBE scheme and Yi's [50] ID-based signature. Yi's [50] ID-based signature is a compatible signature scheme that works well with Boneh-Franklin's [4] IBE scheme. We identify the four steps that are involved in distributing the ID-based public and private key pairs. We also show that the four steps are also involved in signing and verifying signatures.

The Boneh and Franklin ID-based encryption scheme is used by the KGC to preload initial keys which are used for node-to-node communications later on. Yi's ID-based signature scheme uses the public and private keys that are generated by the Boneh and Franklin ID-based encryption scheme to sign messages. It is used for authentication between nodes. The signature generated by Yi's ID-based signature scheme is hashed into a master key and used to distribute a cluster key for broadcasting messages.

4.2 Bilinear Pairing and the DL Problem

Let G_1 be a cyclic additive group generated by p, whose order is a prime q, and let G_2 be a cyclic multiplicative group of the same order, q. We assume that the discrete logarithm problem (DLP) in both G_1 and G_2 is difficult to solve. Let $e : G_1 \times G_1 \to G_2$ be a pairing that satisfies the following conditions:

1) Bilinear: $e(P_1+P_2, Q) = e(P_1, Q)e(P_2, Q)$ and $e(P, Q_1+Q_2) = e(P, Q_1)e(P, Q_2)$ or $e(aP, bP) = e(P, Q)^{ab}$, where $P, P_1, P_2, Q, Q_1, Q_2 \in G_1$ and $a, b \in Z$.

2) Non-degenerate: There exists $P, Q \in G_1$ such that $e(P, Q) \neq 1$.

3) Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Discrete Logarithm Problem (DLP) : Given two group elements P and Q, find an integer n, such that $Q = nP$ whenever such an integer exists. We assume throughout this paper that the DLP is intractable, which means no polynomial time algorithm exists that solves the DLP with non-negligible probability. In practice, G_1 is a point group on an elliptic curve or a Jacobian group on a hyperelliptic curve over a finite field. G_2 denotes a subgroup of the multiplicative group of a finite field.

4.3 Boneh-Franklin Identity Based Encryption

Boneh and Franklin introduced the ID-based encryption (IBE) scheme in 2001. Its implementation relies on a bilinear group and corresponding bilinear pairing e to perform the necessary computations.

The system parameters are: $G_1, G_2, q, P, e, H_1, H_2$ in which G_1, G_2 are two cyclic groups with prime order q, where q is a prime. The generator of G_1 is P. $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing. $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : G_2 \rightarrow \{0,1\}^n$, for some $n \in Z^*$, are two cryptographic hash functions.

The ID-based cryptography system consists of four algorithms: system setup algorithm (Setup), encryption algorithm (Encrypt), key generation algorithm (KeyGen) and decryption algorithm (Decrypt). They are discussed below:

4.3.1 Setup $\rightarrow PP$

We denote PP as a public parameter.

1. Randomly choose $s \in Z_p$ to be the master secret key (MSK), where Z_p is a set that contains all the integers between 1 and p. The MSK is used to generate a private key for each user.

2. Output $PP = \{q, G_1, G_2, e, n, P, P_{pub}, H_1, H_2\}$, where G_1 and G_2 are prime $q - ordered$ groups, P is a generator for G_1 and P_{pub} is the KGC's master public key and $n \in Z^*$ is the length of the public key.

4.3.2 KeyGen$(ID, s) \rightarrow d_{ID}$

Output $d_{ID} = sQ_{ID}$ where $Q_{ID} = H_1(ID) \in G_1$, noting that s is the MSK which is used to generate the private key d_{ID} for the public key Q_{ID}.

4.3.3 Encrypt$(M, ID) \rightarrow CT$

CT denotes the cipher text, M is the message and ID is the node's ID.

1. Randomly choose $r \in Z_p$

2. Compute $g_{ID} = e(H_1(ID), P_{pub})$

3. Output $CT = \langle\ rP, M \times H_2(g_{ID}^r)\rangle$

4.3.4 Decrypt$(CT, d_{ID}) \rightarrow M$

$CT = \langle\ rP, M \times H_2(g_{ID}^r)\rangle$ is the cipher text, d_{ID} is private key of the user.

1. Compute $H_2 \langle\ e(sH_1(ID), rP)\rangle$

   $= H_2 \langle\ e(sH_1(ID), P)^{rs}\rangle = H_2 \langle\ e(sH_1(ID), sP)^r\rangle$

   $= H_2 \langle\ e(sH_1(ID), P_{pub})^r\rangle$

   $= H_2(g_{ID}^r)$

   which is the blinding factor for the message M in CT.

2. Output M $= CT/H_2 \langle\ e(d_{ID}, C_1)\rangle$, where $C_1 = rP$.

4.4 Yi's ID-based Signature Scheme

The signature scheme by Yi [54] is secure against existential forgery under adaptively chosen message and ID attack in the random oracle model.

Existential Forgery: Existential forgery is a weak message related forgery against a cryptographic digital signature scheme. Given a victim's verifying key, an existential forgery is achieved, if the attacker finds a signature σ for at least one new message m, such that the signature σ is valid for m with respect to the victim's verifying key.

Yi's ID-based signature consists of four algorithms: system setup algorithm (Setup), private key extraction algorithm (Extract), signature generation algorithm (Sign) and signature verification algorithm (Verify). They are described below:

The algorithms use two cryptographic hash functions: $H_1 : \{0,1\}^* \rightarrow G_1$, $H_3 : \{0,1\}^* \times G_1 \rightarrow \{0,1\}^n$, for some $n \in Z^*$.

1) Setup: Given a security parameter k, private key generation center (KGC) runs the Gap Diffie-Hellman Groups (GDH) Parameters Generator to obtain $params = \{G_1, G_2, e, P, H_1, H_3\}$. GDH Parameters Generator is a probabilistic algorithm that takes in a given positive integer, which is a security parameter, outputs a cyclic group and sloves Decisional Diffie–Hellman problem in that cyclic group. Then, pick a random number $s \in Z_q$ as a master key and compute the system public key $P_{pub} = sP$. P_{pub} is published, but s is kept secretly.

2) Extract: Given a user's identity ID. The KGC computes $Q_{ID} = H_1(ID)$, $d_{ID} = sQ_{ID}$ and preloads d_{ID} to the node during deployment. The user's private key is d_{ID}.

3) Sign: Given message M, the signer randomly picks a number $r \in Z_q$ and computes $R = rP$, $h = H_3(M, R)$, $Q_{ID} = H_1(ID)$ and $S = rP_{pub} + h \times d_{ID}$. The signature of message M is set to be $\sigma = (R, S)$.

4) Verify: Given a signature $\sigma = (R, S)$, on message M under ID, the verifier computes $h = H_3(M, R)$, $Q_{ID} = H_1(ID)$ and $L = R + h \times d_{ID}$. The verifier accepts the signature if $e(P, S) = e(P_{pub}, L)$.

Correctness of the signature: If $\sigma = (R, S)$ is a valid signature on M under

ID, then

$$e(P, S) = e(P, rP_{pub} + h \times d_{ID}) = e(P, rsP + hsQ_{ID})$$

$$= e(P, s(rP + hQ_{ID})) = e(P, s(R + hQ_{ID}))$$

$$= e(P, sL) = e(sP, L)$$

$$= e(P_{pub}, L).$$

Chapter 5

Key Distribution and Management Scheme

5.1 Introduction

This chapter presents our key distribution and management scheme. We first talk about the adversary model. We show the details of how we distribute keys and how nodes authenticate with each other. The scheme also handles key updates when nodes are joining or leaving. Mutual Authentication Procedure deals with the mutual authentication between CH and CM. Cluster Key Distribution Procedure handles the distribution of cluster keys. The three procedures: CM Node Leaving Procedure, CM Node Joining Procedure and CH Node Leaving Procedure handles the mobility of nodes. Finally, we analyze the security and performance of our scheme and compare our scheme with various other schemes for ad hoc sensor networks.

5.2 Adversary Model

We assume that the adversary (or a malicious node) can eavesdrop, intercept, manipulate and replay any message passing through it. The adversary can also impersonate others by declaring their public information such as ID or location. Every node can also generate its own ID-based public private key pair and signature. But the malicious node is not able to know the others' private information such as the other node's private key or the master secret key.

5.3 Notations

The following are the notations used in our key distribution and management scheme for ad hoc sensor networks.

Table 5.1: Notations and their meanings

Symbols	*Meaning*
T	timestamp
ID_u	ID of node u
I_v	the set of IDs of the CM nodes of cluster in which one of the CM nodes is v
σ_{M+T}	signature of message M plus timestamp using Yi's ID-based signature scheme (Setp 3 of Section 4.3)
K_{ID_v}	master key of node v
d_{ID_v}	private key of node v
C_u	cluster key which is generated by CH u

5.4 System Setup

1. The Key Generation Center (KGC) preloads each node with an ID during node deployment.

2. The KGC runs the GDH Parameters Generator to obtain params $= \{G_1, G_2, e, P, P_{pub}\}$.

3. The KGC chooses a random number $s \in Z_q$ as the master secret key that is only known to the KGC, and $P_{pub} = sP$ as the master public key.

5.5 Extraction

For every node v, if its identity is $ID_v \in \{0,1\}^*$, the KGC preloads the node with $Q_{ID_v} = H_1(ID_v)$ as its public key and $d_{ID_v} = sQ_{ID_v}$ as its secret key. The KGC also publishes the public parameters $PP = \{q, G_1, G_2, e, n, P, P_{pub}, H_1, H_2, H_3\}$ in which G_1, G_2 are two cyclic groups with prime order q, where q is a prime. The generator of G_1 is P. $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing. $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : G_2 \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^* \times G_1 \rightarrow \{0,1\}^n$, for some $n \in Z^*$, are three cryptographic hash functions.

5.6 Mutual Authentication Procedure

1. Every CH u broadcasts a HELLO beacon $(ID_u, T, \sigma_{ID_u+T})$ to all his MNs.

2. Every CM v verifies the σ_{ID_u+T}. If $e(P, S) = e(P_{pub}, L)$, using Step 4 of Section 4.3, this proves that u's ID and private key are preloaded by the KGC. Validate that the timestamp is within an acceptable number of time units of the current time, this prevents the message replay attacks.

3. Every CM v replies with a HELLO beacon (ID_v, σ_{ID_v+T}) to its CH u.

4. CH u verifies the σ_{ID_u+T}. If $e(P,S) = e(P_{pub}, L)$, using Step 4 of Section 4.3, then this proves that v's ID and private key are preloaded by the KGC. Validate that the timestamp is within an acceptable number time units of the current time, this prevents message replay attacks.

5. CH u keeps the signatures σ_{ID_v+T} for every authenticated MN v.

5.7 Master Key Derivation

The CH u derives a CM v's master key: $K^v_{master} = F(\sigma_{ID_v+T})$, where F is a hash function: $G_1 \times G_1 \rightarrow \{0,1\}^n$ for some $n \in Z^*$, and $\sigma_{ID_v+T} = (R,S)$ is a signature on message $(ID_v + T)$, $R \in G_1$, $S \in G_1$.

5.8 Cluster Key Distribution Procedure

Cluster key distribution uses the polynominal technique in Zhao's paper [54]. For each CH u:

1. Generates a cluster key C_u, calculates the polynomial $g_u(x) = \prod^n_{i \in I_v}(x - K_i)$.

2. Broadcasts $m_u(x) = g_u(x) + C_u$.

3. When a CM v receives the $m_u(x)$, it substitutes x with its master key K_{ID_v}.

4. v obtains the cluster key C_u. Note that $g_u(K_{ID_v}) = 0$, so $m_u(K_{ID_v}) = g_u(K_{ID_v}) + C_u = C_u$.

5.9 CM Node Leaving Procedure

CM leaving can be detected using a neighbor discovery protocol. When a CM v leaves:

1. The CH u generates a new cluster key $C\prime_u$.

2. The CH u calculates a new polynomial $g'_u(x) = \prod_{i \in I_v \& i \neq ID_v}^{n} (x - K_i)$.

3. u broadcasts $m'_u(x) = g'_u(x) + C\prime_u$.

4. A CM w receives the $m'_u(x)$, it substitutes x with its master key K_{ID_w}.

5. w obtains the new cluster key $C\prime_u$.

5.10 CM Node Joining Procedure

Node joining can be detected using a neighbor discovery protocol. If node v joins a cluster:

1. The CH u generates a new cluster key C'_u.

2. The CH u calculates a new polynomial $g'_u(x) = g_u(x)(x - K_{ID_v})$.

3. u broadcasts $m'_u(x) = g'_u(x) + C'_u$.

4. A CM w receives the $m'_u(x)$, it substitutes x with its master key K_{ID_w}.

5. w obtains the new cluster key C'_u.

5.11 CH Node Leaving Procedure

CH leaving can be detected using a neighbor discovery protocol. The newly elected CH just repeats the **Mutual Authentication Procedure** in Section 5.6.

5.12 Scheme Analysis

5.12.1 Correctness of Key Distribution Scheme

The key distribution scheme has the following properties:

Theorem 1. *The difficulty with which an insider node can successfully impersonate another node by forging the node's master key or ID without knowing the node's private key is as hard as attacking the key generation algorithm.*

Proof. We prove the above by contradiction. Suppose an inside malicious node u can forge node v's master key, we know that v's master key is derived from v's signature. In order to forge the master key, node u should be able to forge the signature. However, the signature scheme by Yi [50] is secure against existential forgery under adaptively chosen message and ID attack in the random oracle model. In order to forge the signature, the only way for node u is to know v's private key which is contradictory to the assumption that node u does not know v's private key. Note: refer to Section 4.4 for the existential forgery.

If an inside malicious node u can successfully impersonate v's ID ID_v which is also v's public key, then during the authentication using the signature scheme we know that the node u has to know v's private key in order to pass authentication. This authentication procedure is contradictory to the assumption that node u does not know v's private key.

So an inside node is not able to impersonate another node by forging the node's master key or ID without knowing the node's private key. ■

Theorem 2. *The probability that an inside node can successfully decrypt a broadcast message from other clusters is no better than a random guess of the cluster key of other clusters.*

Proof. From the cluster key distribution procedure, we know that in order to derive the cluster key of other clusters, a node needs to have the master key of one of the nodes from the other clusters. However, from **Theorem 1**, we know that a node will not be able to forge the master key without knowing the node's private key from the other clusters. However, the assumption is that the private key is kept secretly, so the node can only make a random guess of the cluster key of another cluster in order to decrypt the other clusters' broadcast messages. ■

Theorem 3. *A node can generate its own ID-based public and private key pair or even use any node's ID to generate such a key pair, but without knowing the KGC's master secret key s, the node is not able to pass the mutual authentication procedure and is not able to use this key pair to communicate with other nodes.*

Proof. As the KDC's master public key is used as a public parameter for encryption, and the decryption and signing by every node, only the node that has the preloaded public private key pair or the master secret key s from the KGC can pass authentication. Meanwhile, s is kept secret and only known by the KGC.

Also, when node u uses node v's own generated public key to encrypt messages, node v is not able to decrypt the cipher text using its own generated private key because when node u encrypts the message, KGC's master public key is used as a public parameter during the encryption process. ■

5.12.2 Performance and Security Comparison

Computation Cost Comparison: Let us assume the resource cost for encrypting a message is m and the resource cost for decrypting a message is n, the resource cost to broadcast a message is k, and we denote the size of the sensor network to be N. We compare the computation cost to distribute a cluster key for a cluster.

We know that for LEAP in [57], they use pairwise keys to distribute cluster keys. If the CH's degree is d, the cluster head needs to use every member node's pairwise keys to encrypt the cluster key. The total number of symmetric encryptions during this procedure is d, and the total number of symmetric decryptions during this procedure is also d. So the computation cost is $d \times (m + n)$. The computation cost for other schemes that use pairwise keys to distribute cluster keys is also $d \times (m + n)$.

While in our scheme, the cluster head just needs to encrypt the cluster key once which is achieved by appending the cluster key to a polynomial. The encryption and decryption is different from the symmetric encryption and decryption procedure. We denote the resource cost to encrypt and decrypt a cluster key using polynomials as m' and n'. So the communication cost to distribute a cluster key for a cluster is $(m' + n')$. We know that $m' < m$ and $n' < n$.

Our scheme's computation cost is less than the LEAP scheme in [57] and any schemes that use a pairwise key to distribute the cluster key.

Communication Cost Comparison: The analysis of communication cost to distribute a cluster key is similar to the analysis of computation cost. Because the CH uses pairwise keys to distribute a cluster key in LEAP, the CH has to broadcast the encrypted message d times. The number of messages broadcasted

is d and the communication cost is $d \times k$. However, in our scheme, the CH needs to append the cluster key to a polynomial and broadcast the encrypted message once. So the communication cost is k.

Storage Comparison: For storage, our scheme needs to store a private key and a cluster key, so the storage is $O(1)$. According to LEAP, each node has to store $(3d + 2 + L)$ keys. So the storage is $O(3d + L)$ and L is the size of the key chain. For other schemes that use pairwise keys to distribute cluster keys, every node needs to store the all pairwise keys with all its neighbour nodes plus a cluster key. So the storage is $O(d)$. For Eschenauer [17] and Chan [8]' schemes, though they don't need to store the pairwise keys, they need to store a subset of keys from a key pool. c and w are the size of the keys they store in each scheme.

Security comparison in terms of the effect of a node capture: We know that for LEAP in [57] and the scheme in [54], each node is preinstalled a seed e, and this initial key is used to establish pairwise keys between two nodes. If a node is captured, the secret seed e is known and the whole network's pairwise keys are known. Though the seed can be deleted after pairwise keys are established, every node still keeps seed e in its memory. Also, the newly joined node has and needs seed e to establish the pairwise key with the existing nodes in the network. If a node that has seed e in its memory is captured, the whole network is compromised.

In our scheme, if a node is captured, the private key of that node is known. But from the private key of that node, the attacker cannot gain additional information about the system's master secret key nor the other nodes' private keys. The whole network will not be compromised, and only the compromised node is affected.

We investigated the recent key management schemes for ad hoc sensor networks and compared them to different perspectives that are shown in Table 5.2 and

Table 5.3

Table 5.2: Security comparison of a list of key management schemes for ad hoc sensor networks

Protocol	*Key Gen*	*Cluster Key*	*Vulnerability*
Our Scheme	KGC, ID-based scheme	Polynomials	Key Escrow
LEAP [57]	Key server, symmetric key seed (e)	Pairwise keys	Impersonation, node capture
Zhao [54]	Base station, symmetric key chains	Polynomials	Impersonation, node capture
Zhou [56]	Base station, ECC and Trivariate Symmetric Polynomial	Pairwise keys	unauthenticated node can join
Jiang [31]	Base station, bilinear Pairing-based Scheme	Pairwise keys	unauthenticated node can join
Eschenauer [17]	Base station, probabilistic symmetric key	Pairwise keys	node capture
Chan [8]	Base station, q-composite probabilistic symmetric key	Pairwise keys	node capture

Table 5.3: Performance comparison of a list of key management schemes for ad hoc sensor networks

Protocol	*Storage*	*Computation*	*Communication*
Our Scheme	$O(1)$	$m' + d \times n'$	k
LEAP [57]	$O(3d + L)$	$d \times (m + n)$	$d \times k$
Zhao [54]	$O(d)$	$m' + d \times n'$	k
Zhou [56]	$O(d)$	$d \times (m + n)$	$d \times k$
Jiang [31]	$O(d)$	$d \times (m + n)$	$d \times k$
Eschenauer [17]	$O(c)$	$d \times (m + n)$	$d \times k$
Chan [8]	$O(w)$	$d \times (m + n)$	$d \times k$

The Protocol column represents all protocols of the key schemes we compare. The Key Gen column represents what components are used for the key generation and what kind of keys are preloaded. For example, the LEAP protocol uses a key server to preload each node with a symmetric key seed (e). The Cluster Key column represents the way each scheme distributes cluster keys. For example, our scheme uses polynomials to distribute cluster keys while in LEAP, they use pairwise keys to distribute cluster keys. The Energy Cost column represents the energy cost for each scheme to distribute a cluster key. The vectors used are defined

in Section 5.12.2. The Vulnerability column represents the vulnerabilities that each scheme has. For example, Sushmita's protocol has the impersonation vulnerability. Our Scheme has the key escrow problem because we use the IBE scheme (Note: we talked about the existing countermeasures to the key escrow problems for IBE schemes in Section 2.8). This means that if a node is captured, it causes the other nodes secret to be released. The Zhou [56] and Jiang [31]'s schemes do not have impersonation problems. However, each node can't authenticate with each other and any unauthenticated node can join their network as long as it has the capability to generate public and private key pairs.

Chapter 6

Extended Scheme for the Hierarchical Model

6.1 Introduction

This chapter extends our key distribution and management scheme to the hierarchical model. In the hierarchical model, some nodes may have several interfaces and can play the role of CH in several clusters at different levels. In order to be adaptable to a hierarchical topology, we modified the cluster key and master key notations which specify the indices of the clusters for the same CH node. We first talk about the adversary model. We show the details of how we distribute keys and how nodes authenticate with each other. The scheme also handles key updates when there are nodes joining or leaving. Mutual Authentication Procedure deals with the mutual authentication between CH and CM. Cluster Key Distribution Procedure handles the distribution of cluster keys. The three procedures: CM Node Leaving Procedure, CM Node Joining Procedure and CH Node Leav-

ing Procedure handle the mobility of nodes. Finally, we analyze the security and performance of our scheme.

6.2 Notations

The following notations are used in our key distribution and management scheme for hierarchical ad hoc sensor networks

Table 6.1: Notations and their meanings for Hierarchical Model

Symbols	*Meaning*
T	timestamp
ID_u	ID of node u
σ_{M+T}	signature of message M plus timestamp using Yi's ID-based signature scheme
K_{ID_v}	master key of CM node v
d_{ID_v}	private key of node v
$C_{u,k}$	cluster key generated by CH u for its k_{th} cluster (CH u's clusters are indexed from 1,2,...,n where n is the number of CH u's clusters)
I_v	the set of IDs of the CM nodes within the same cluster in which one of the CM nodes is v
S_u	the set of clusters in which the node u plays the role of cluster head
$g_u^k(x)$	the polynomial calculated by using the master keys of CMs in k_{th} cluster of CH u
$m_u^k(x)$	the polynomial calculated by appending $C_{u,k}$ to $g_u^k(x)$

6.3 Extension

In hierarchical ad hoc sensor networks, nodes are formed in clusters at several levels. Some nodes have several interfaces. They may play the role of CH in several clusters. So for cluster key distribution, a CH needs to generate different

cluster keys for its different clusters and calculates different polynomials $m(x)$ to distribute different cluster keys for each of its clusters.

For the node leaving and joining procedures, the CH only needs to update the cluster key and distribute it to the cluster in which the leaving or joining node is. It does not need to broadcast the polynomials $m(x)$ through all its interfaces to all its clusters.

The logic of the mutual authentication, cluster key distribution and updating procedures is similar to the logic in Chapter 5.

6.4 Extraction

For every node v if its identity is $ID_v \in \{0,1\}^*$, KGC preloads the node $Q_{ID_v} = H_1(ID_v)$ as its public key and $d_{ID_v} = sQ_{ID_v}$ as its secret key. KGC also publishes the public parameters $PP = \{q, G_1, G_2, e, n, P, P_{pub}, H_1, H_2, H_3\}$ in which G_1, G_2 are two cyclic groups with prime order q, where q is a prime. The generator of G_1 is P. $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing. $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : G_2 \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^* \times G_1 \rightarrow \{0,1\}^n$, for some $n \in Z^*$, are three cryptographic hash functions.

6.5 Mutual Authentication Procedure

1. For every CH u

2. For every $cluster \in S_u$

3. CH u broadcasts a HELLO beacon (ID_u, σ_{ID_u+T}) to all his MNs

4. Every CM v verifies the σ_{ID_u+T}. If $e(P,S) = e(P_{pub}, L)$, this proves v's ID and private key are preloaded by the KGC, also implies that the timestamp is valid.

5. Every CM v replies a HELLO beacon (ID_v, σ_{ID_v+T}) to its CH u

6. CH u verifies the σ_{ID_u+T}. If $e(P,S) = e(P_{pub}, L)$, this proves v's ID and private key are preloaded by the KGC, also implies that the timestamp is valid.

7. CH u keeps the signatures σ_{ID_v+T} for all the authenticated MNs v.

6.6 Master Key Derivation

CH u derives its k_{th} cluster's MN v's master key : $K_v = F(\sigma_{ID_v+T})$, where F is a hash function: $G_1 \times G_1 \rightarrow \{0,1\}^n$ for some $n \in Z^*$, and $\sigma_{ID_v+T} = (R,S)$ which is a signature on the message $(ID_v + T)$, $R \in G_1$, $S \in G_1$.

6.7 Cluster Key Distribution Procedure

1. For every CH u

2. For every $cluster \in S_u$

3. Generates a cluster key $C_{u,k}$ for its k_{th} cluster, calculates polynomial $g_u^k(x) = \prod_{i \in I_v}(x - K_i)$

4. Broadcasts $m_u^k(x) = g_u^k(x) + C_{u,k}$ to its k_{th} cluster for $k = 1, 2, ..., n$. where n is the number of CH node u's clusters

5. CM v receives the $m_u^k(x)$, substitutes x with its master key K_{ID_v}

6. v obtains the cluster key $C_{u,k}$. Note: $g_u^k(K_{ID_v}) = 0$, so $m_u^k(K_{ID_v}) = g_u^k(K_{ID_v}) + C_{u,k} = C_{u,k}$

6.8 CM Node Leaving Procedure

If node v in cluster head u's k_{th} cluster leaves:

1. The CH u generates a new cluster key $C'_{u,k}$.
2. The CH u calculates a new polynomial $g_u^{'k}(x) = \prod_{i \in I_v \& i \neq ID_v}(x - K_i)$.
3. u broadcasts $m_u^{'k}(x) = g_u^{'k}(x) + C'_{u,k}$ using the interface for its k_{th} cluster.
4. Every CM w receives the $m_u^{'k}(x)$, substitutes x with its master key K_w.
5. Every CM w obtains the new cluster key $C'_{u,k}$.

6.9 CM Node Joining Procedure

If node v joins in cluster head u's k_{th} cluster:

1. The CH u generates a new cluster key $C'_{u,k}$.
2. The CH u calculates a new polynomial $g_u^{'k}(x) = g_u^k(x)(x - K_{ID_v})$.
3. u broadcasts $m_u^{'k}(x) = g_u^{'k}(x) + C'_{u,k}$ using the interface for its k_{th} cluster.
4. Every CM w receives the $m_u^{'k}(x)$, substitutes x with its master key K_w.
5. Every CM w obtains the new cluster key $C'_{u,k}$.

6.10 CH Node Leaving Procedure

The newly elected CH just repeats the **Mutual Authentication Procedure** in Section 6.5.

6.11 Scheme Analysis in Hierarchical Model

The key distribution and management scheme for the hierarchical model is based on the one in Chapter 5. The hierarchical model scheme has the same properties as the one in Chapter 5. We do not want to repeat the validation process of Chapter 5, instead we list the properties as follows:

- The difficulty with which an insider node can successfully impersonate another node by forging the node's master key or ID without knowing the node's private key is as hard as attacking the key generation algorithm.
- The probability that an inside node can successfully decrypt a broadcast message from another cluster is no better than a random guess of the cluster key of another cluster.
- A node can generate its own ID-based public and private key pair or even use any node's ID to generate such a key pair without knowing the KGC's master secret key s. The node is not able to pass the mutual authentication procedure and is not able to use the key pair to communicate with other nodes.

6.11.1 Performance and Security Comparison in Hierarchical Model

Computation Cost Comparison: In the hierarchical model, let us assume that the resource cost for encrypting a message is m and the resource cost for decrypting a message is n. The resource cost for broadcasting a message is k via one interface. We denote the degree of CH in its i-th cluster to be d_i. We compare the computation cost to distribute a cluster key for a CH's i-th cluster.

We know that for LEAP in [57], they use pairwise keys to distribute cluster keys. If the the degree of the CH in its i-th cluster is d_i, then the cluster head needs to use every member node's pairwise key from its i-th cluster to encrypt the cluster key and broadcast different encrypted messages to every member node within the i-th cluster via its corresponding interface. The total number of messages broadcasted during this procedure is d_i. The computation cost to distribute a cluster key for a cluster is $d_i \times (m + n)$.

While in our scheme, the CH just needs to encrypt the cluster key once which is achieved by appending the cluster key to a polynomial and broadcasting the encrypted message once. The encryption and decryption is different from the symmetric encryption and decryption procedure. We denote the resource cost to encrypt and decrypt a cluster key using polynomials are m' and n'. So the communication cost to distribute a cluster key for a cluster is $(m' + n')$. We know that $m' < m$ and $n' < n$.

So our scheme's computation cost is less than the LEAP scheme in [57] and any schemes that use pairwise key to distribute the cluster key.

Communication Cost Comparison: The analysis of communication cost

to distribute a cluster key is similar to the analysis of computation cost. Because the CH uses pairwise keys to distribute a cluster key in LEAP, it has to broadcast d_i times for its i-th cluster. The number of messages broadcasted is d_i and the communication cost is $d_i \times k$. However, in our scheme, the CH just needs to append the cluster key to a polynomial and broadcast the encrypted message once. So the communication cost is k.

Storage Comparison: For storage, our scheme just need to store a private key and a cluster key, so the storage is $O(1)$.

According to LEAP, each node has to store $(3d+2+L)$ keys. So the storage is $O(3d+L)$ and L is the size of the key chain. For other schemes that use pairwise keys to distribute cluster keys, every CH needs to store all the pairwise keys with all its neighbour nodes plus a cluster key. So the storage is $O(d_1 + d_2 + ...d_n)$ (CH's clusters are indexed from 1,2,...,n where n is the number of CH's clusters). For Eschenauer [17] and Chan [8]' schemes, though they don't need to store the pairwise keys, they need to store a subset of keys from a key pool. The c and w are the size of the keys they store in each schemes.

Security comparison in terms of the effect of a node capture: We know that for LEAP in [57] and the scheme in [54], each node is preconfigured with a seed e. This initial key is used to establish pairwise keys between node pairs. If a node is captured, then the secret seed e is revealed as well as the whole network's pairwise keys. Though the seed can be deleted in each node after the pairwise keys are established, until the pairwise keys are established, every node still keeps the seed e in its memory. Also, a new joining node has and needs the seed e to establish the pairwise key with existing nodes in the network. So, if a node that contains the seed e in memory is captured, then the whole network is

compromised.

In our scheme for the hierarchical model, if a node is captured, then its private key is revealed. However, using the private key of that node, the attacker cannot gain additional information about the system's master secret key. Other nodes' private keys are still kept secret. So the whole network is not compromised. Only the compromised node is affected.

Chapter 7

Conclusion

This chapter summarizes the methodology used, the problems and the solution to these problems. Below, we also discuss the possibility of future work.

7.1 Conclusion

We resolved the impersonation problem in the key distribution schemes [57, 54] for ad hoc sensor networks. The methodologies that the key distribution schemes use for ad hoc sensor networks are to preload each sensor with an initial key during deployment. The initial keys may vary in different schemes. Some may use a sigle initial key for the whole network, others use a pool of initial keys. After the initial key predeployment, each node uses its preloaded initial key along with some public information such as its ID, or (x, y) coordinates to derive master keys which are used to generate the pairwise keys for node-to-node communications. Then, the pairwise keys are used to distribute the cluster key for broadcasting messages within a cluster. These schemes can prevent outsider attacks. However,

they have serious impersonation problems. Every insider node can derive any other nodes' master keys thus deriving any other nodes' pairwise keys. There is no authentication method to prevent that.

To fix this impersonation problem, we use Boneh's IBE system along with Yi's ID-based signature scheme to provide key distribution and management for ad hoc sensor networks. Leveraging Yi's ID-based signature scheme, the impersonation problem is fixed. The key distribution and management scheme also improves the security of sensor networks as explained in Section 6.11.1. If a node is captured, the node's secret is released, but the other node's secret is kept secretly in our key scheme. However, [57, 54] if the one node is captured, other nodes' secret is released. We also extend our key distribution and management scheme to the hierarchical network model.

7.2 Future Work

We proposed a key distribution and management scheme for clustered ad hoc sensor networks. We combined both symmetric and asymmetric keys to make this scheme free of impersonation problems. Although our scheme improves the security for clustered ad hoc sensor networks, it sacrifices some efficiency due to asymmetric encryption costs and the additional resources needed than the symmetric one.

Our key distribution and management scheme uses both symmetric and asymmetric keys to encrypt messages. So the underlying graph is directed. This means that the communication paths are directed. So the communication path might be longer. Our future work will be focused on investigating how many extra communication paths that occurs due to the incorporation of asymmetric key methods

and find ways to balance between security and performance requirements.

Also, we have extended our scheme to the hierarchical model. Moving forward, the techniques that are used in a one level clustered model such as the encryption schemes and signature schemes, group key management scheme needs to be refined for hierarchical ad hoc networks. Most of the existing group key management schemes aim to secure communication within a single group. As they are designed for one-level clustered topology, it is not suitable for multiple group multicast environments such as the hierarchical model where CH has multiple clusters. Recently, a group key management scheme by Min-Ho [41] shows promising work that our scheme can leverage for our cluster key distribution for the hierarchical model. This scheme also uses asymmetric keys to distribute the cluster keys. The overhead generated by rekeying the cluster key can be reduced by leveraging this scheme.

Bibliography

[1] Aziz Baayer, Nourdine Enneya, and Mohamed EI Koutbi. A recent survey on key management schemes in manet. *3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6, 2008.

[2] Joonsang Baek and Yuliang Zheng. Identity-based threshold signature scheme from the bilinear pairings. *International Conference on Information Technology: Coding and Computing. Proceedings*, pages 124–128, 2004.

[3] Stefano Basagni. Distributed clustering algorithm for ad-hoc networks. *Parallel Architectures, Algorithms, and Networks. Fourth InternationalSymposium*, pages 310–315, 1999.

[4] Dan Boneh, Xuhua Ding, Gene Tsudik, and Chi Ming Wong. A method for fast revocation of public key certificates and security capabilities. *Proceedings of the 10th conference on USENIX Security Symposium*, pages 22–22, 2001.

[5] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, 2001.

[6] Dan Boneh, Ilya Mironov, and Victor Shoup. A secure signature scheme from bilinear maps. *In: Joye, M. (ed.) CT-RSA. LNCS*, pages 98–110, 2003.

[7] Jan Camenisch and Markus Stadler. Efficient group signatures schemes for large groups. *Advances in Cryptology-Crypto, LNCS 1294*, 1294:410–424, 1997.

[8] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. *Security and Privacy*, pages 197–213, 2003.

[9] David Chaum and Eugene van Heijst. Group signatures. *Advances in Cryptology-Eurocrypt, LNCS 547*, pages 257–265, 1991.

[10] Lei Chen and Tod Pedersen. On the efficiency of group signatures providing information-theoretic anonymity. *Advances in Cryptology-Eurocrypt, LNCS 1233*, pages 465–479, 1997.

[11] Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. A new id-based group signature scheme from bilinear pairings. *Journal of Electronics (China)*, 2908:892–900, 1997.

[12] Xiangguo Cheng, Lifeng Guo, and Xinmei Wang. An identity-based mediated signature scheme from bilinear pairing. *International Journal of Network Security*, 2(1):29–33, 2006.

[13] Sherman S. Chow. Removing escrow from identity-based encryption. *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, pages 104–115, March 18-20, 2009.

[14] Hongmei Deng, A. Mukherjee, and D.P. Agrawal. Threshold and identity-based key management and authentication for wireless ad hoc networks. *International Conference on Information Technology: Coding and Computing. Proceedings*, pages 107–111, 2004.

[15] Yvo Desmedt and Jean-Jacques Quisquater. Public-key systems based on the difficulty of tampering. *Proceedings on Advances in cryptology—CRYPTO '86*, pages 111–117, 1986.

[16] Anthony Ephremides, J.E. Wieselthier, and D.J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 75(1):56–73, 1987.

[17] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. *Proc of the 9th ACM conference on computer and communication security*, pages 41–47, 2002.

[18] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, Tal Rabin, Steffen Reidt, and Stephen D. Wolthusen. Strongly-resilient and non-interactive hierarchical keyagreement in manets. *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, 5283:49–65, 2008.

[19] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pages 548–566, 2002.

[20] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, mobile, multimedia radio network. *Wirel. Netw*, 1(3):255–265, 1995.

[21] K. Gomathi and B. Parvathavarthini. An efficient cluster based key management scheme for manet with authentication. *Trendz in Information Sciences Computing (TISC)*, pages 202–205, 2010.

[22] Vipul Goyal. Reducing trust in the pkg in identity based cryptosystems. *Proceedings of the 27th annual international cryptology conference on Advances in cryptology*, pages 430–447, 2007.

[23] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. *Proceedings of the 15th ACM conference on Computer and communications security*, pages 427–436, 2008.

[24] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.

[25] D.S.M. Hassan, H.M.A. Fahmy, and A.M. Bahaa. Ring clustering algorithm for wireless ad hoc networks. *15th IEEE Mediterranean Electrotechnical Conference*, pages 458–465, 2010.

[26] Wenbo He, Ying Huang, R. Sathyam, K. Nahrstedt, and W.C. Lee. Smock: A scalable method of cryptographic key management for mission-critical wireless ad-hoc networks. *Information Forensics and Security, IEEE Transactions on*, 4(1):140–150, 2009.

[27] Anne Marle Hegland, Eli Winjum, Sting F. Mjolsnes, Chunming Rong, Qivind Kure, and Pal Spilling. A survey of key management in ad hoc networks. *Communications Surveys Tutorials, IEEE*, 8(3):48–66, 2006.

[28] Detlef Hühnlein, Michael J. Jacobson, Jr., and Damian Weber. Towards practical non-interactive public key cryptosystems using non-maximal imaginary quadratic orders. *Des. Codes Cryptography*, 30(3):281–299, 2003.

[29] AbdelRahman Hussein, Sufian Yousef, Samir Al-Khayatt, and Omar S. Arabeyyat. An efficient weighted distributed clustering algorithm for mobile ad hoc networks. *International Conference on Computer Engineering and Systems (ICCES)*, pages 221–228, 2010.

[30] Dennis J.Baker, Anthony Ephremides, and Julia A Flynn. The design and simulation of a mobile radio network with distributed control. *IEEE Journal on Selected Areas in Communications SAC-2, 1*, 2(1):226–237, 1984.

[31] Rongrong Jiang and Tieming Chen. A bilinear pairing-based key management scheme for wireless sensor networks. *Seventh International Conference on Computational Intelligence and Security (CIS)*, pages 670–673, 2011.

[32] Saju P Joh and Philip Samuel. A distributed hierarchical key management scheme for mobile ad hoc networks. *International Conference on Information Networking and Automation (ICINA)*, pages 308–314, 2010.

[33] Antoine Joux. A one round protocol for tripartite diffie-hellman. *In Algorithmic Number Theory Symposium, ANTS-IV, Springer-Verlag LNCS 1838*, 17(4):263–276, 2000.

[34] Andreas Larsson and Philippas Tsigas. A self-stabilizing (k,r)-clustering algorithm with multiple paths for wireless ad-hoc networks. *Proceedings of the 14th international conference on Principles of distributed systems*, pages 79–82, 2010.

[35] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Secure key issuing in id-based cryptography. *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, 32(6):69–74, 2004.

[36] Lung-Chung Li and Ru-Sheng Liu. Securing cluster-based ad hoc networks with distributed authorities. *Wireless Communications, IEEE Transactions*, 9(10):3072–3081, 2010.

[37] Chunhung Richard Lin and Mario Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, 1997.

[38] Ueli M. Maurer and Yacov Yacobi. Non-interactive public-key cryptography. *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, pages 498–507, 1991.

[39] Xianyong Meng and Yangmin Li. A verifiable dynamic threshold key management scheme based on bilinear pairing without a trusted party in mobile ad hoc network. *IEEE International Conference on Automation and Logistics (ICAL)*, pages 315–320, 2012.

[40] Johann Van Der Merwe, Dawoud Dawoud, and Stephen McDonald. A survey on peer-to-peer key management for mobile ad hoc networks. *ACM Comput. Surveys*, 39(1):3–4, 2007.

[41] Min-Ho Park, Young-Hoon Park, Han-You Jeong, and Seung-Woo Seo. Key management for multiple multicast groups in wireless networks. *Mobile Computing, IEEE Transactions*, 12(9):1712–1723, 2013.

[42] Sangjoon Park, Seungjoo Kim, and Dongho Won. Id-based group signature. *Electronics Letters*, 33(19):1616–1617, 1997.

[43] Mahalingam Ramkumar, Nasir Memon, and Rahul Simha. A hierarchical key predistribution scheme. *IEEE International Conference on Electro Information Technology*, pages 6–6, 2005.

[44] Amit Sahai and Brent R. Waters. Fuzzy identity based encryption. *In Advances in Cryptology - Eurocrypt*, 3494(11):45–47, 2004.

[45] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[46] Hatsukazu Tanaka. A realization scheme for the identity-based cryptosystem. *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, pages 340–349, 1987.

[47] S. Tsujii and T. Itoh. An id-based cryptosystem based on the discrete logarithm problem. *Selected Areas in Communications, IEEE Journal*, 7(4):467–473, 1989.

[48] Xiaofeng Wang and Shangping Wang. An identity-based mediated signature scheme without trusted pkg. *Informatica 35*, pages 83–90, 2011.

[49] Brent Waters. Efficient identity based encryption without random oracle. *Eurocrypt2005,LNCS 3494*, pages 114–127, 2005.

[50] Xun Yi. An identity-based signature scheme from the weil pairing. *IEEE Communications Letters, IEEE*, 7(2):76–78, 2003.

[51] Richard Yu, Helen Tang, Peter.C Mason, and Fei Wang. A hierarchical identity based key management scheme in tactical mobile ad hoc networks. *Network and Service Management, IEEE Transactions*, pages 258–267, 2010.

[52] Jianhong Zhang and Xue Liu. An efficient strong id-based signature scheme with unforgeability. *2010 Fifth International Conference on Frontier of Computer Science and Technology (FCST)*, pages 239–245, 2010.

[53] Leyou Zhang and Yupu Hu. A new construction of short hierarchical identity-based signature in the standard model. *IJCSNS International Journal of Computer Science and Network S 180 ecurity*, 9(4):128–129, 2009.

[54] Huawei Zhao and Ruixia Liu. Hiberarchy cluster model and key management in wireless sensor network. *Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing*, pages 3590–3593, 2009.

[55] Lidong Zhou and Z.J Haas. Securing ad hoc networks. *Network, IEEE*, 13(6):24–30, 1999.

[56] Rui Zhou and Hua Yang. A hybrid key management scheme for heterogeneous wireless sensor networks based on ecc and trivariate symmetric polynomial. *2011 International Conference on Uncertainty Reasoning and Knowledge Engineering (URKE)*, pages 251–255, 2011.

[57] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. *Proceedings of the 10th ACM conference on Computer and communications security, Washington D.C. ,USA*, pages 62–72, 2003.