A Secured Mobile Ad Hoc Network Using Certificateless Effective key Management (CL-EKM)

Dr. M.Gobi
Assistant Professor,Department of Computer Science,Chikkanna Government Arts College, Tirupur.
Email: mgobimail@yahoo.com
L.Moorthi
Research Scholar,Department of Computer Science, Chikkanna Government Arts College, Tirupur.
Email: lmoorthi24@gmail.com

----------**ABSTRACT**----------
The Security Mobile Ad Hoc Networks (MANET) is a crucial task for their good deployments. One fundamental aspect of providing confidentiality, integrity and authentication is key management. MANET have been deployed for a wide variety of applications, including military sensing and tracking, patient status monitoring, traffic flow monitoring, where sensory devices often move between different locations. Securing data and communications requires suitable encryption key protocols. In this paper, we propose a certificateless effective key management (CL-EKM) protocol for secure communication in MANET characterized by node mobility. The CL-EKM supports efficient key updates. The protocol also supports efficient key revocation for compromised nodes and minimizes the impact of a node compromise on the security of other communication links. A security analysis of our scheme shows that our protocol is effective in defending against various attacks. The performance results prove the effectiveness of our proposed key management scheme CL-EKM in MANET.

Keywords - **Group Confidentiality, Key Management, MANETs, Network Security.**

---

1. Introduction

MANET is a network consisting of collection of nodes capable of communicating with one another without the assistance of network infrastructure. In a MANET, each mobile node acts as a router. The main advantage of MANET is that it can operate in isolation or in coordination with wired infrastructure. If a message is sent out through a general tunnel without encryption, it may suffer malicious attacks.

Each node, which acts like a mobile router, has full control over the data that pass through it. Some of these are malicious nodes, which enter the network during establishment phase while others may originate indigenously by compromising an existing benevolent node. These malicious nodes can carry out both passive and active attacks against the network. In passive attacks, a malicious node only eavesdrops upon packet contents without disrupting the network operation, while active attacks can fabricate, modify or drop a packets. Because of these attacks, security is necessary to guard against attacks.

Cryptography is an important and powerful tool for security services, namely authentication, confidentiality, integrity and non-repudiation. Key management is a basic part of any secure communication. Key management deals with key generation, storage, distribution, updating, and revocation and certificate services, in accordance with security policies. Absence of secure key management makes a network vulnerable to attack. Key management schemes usually focus on improving security and optimizing the key storage. The limited resources and mobility of nodes are bottleneck of MANET security. An effective key management system can solve this problem. In mobile ad hoc network, a group can hasten message delivery and prevent bandwidth waste effectively. Group confidentiality is one of the issues in group key management used in assuring secure multicast group communication where limited broadcast is used. Group confidentiality requires that only valid group users could decrypt the multicast data even if the data is broadcast to the entire network.

In this paper, we proposed a certificateless effective key management scheme (CL-EKM) which does not require online certification authority for secure broadcast communication. Rest of the paper is organized as follows: summarizes some of the previous works that have been proposed for key management in MANETs and the advantages as well disadvantages of such works.a discuss about the proposed scheme and working of proposed new scheme is discussed for secure group communication in MANET. The effectiveness of proposed scheme is described in this paper.

2. Related Work

Many of the security solution have been proposed for MANET. Some of the research papers focus on either secure routing transmission or key management in MANET are described below:

A distribution of symmetric key generation system (SKGS) based on key pre-distribution scheme is given in. In SKGS, a central server is responsible for the creation and distribution of nodal key chains. Drawback of this scheme is nodes can derive future key from the key chain which they receive from the main server and decrypt future traffic, hence lack of backward secrecy. Another problem in the SKGS scheme is single point of failure of the central server.

In reference, a secure key management scheme is proposed based on *(t, n)* thresholds cryptography has been presented. Where *n* is the total number of nodes in network and *t* is the number of nodes required to generate certificate. The system can tolerate *t-1* compromised servers, however, this scheme does not describe how a node can contact *t* servers securely when server are scattered in a large area and minimum *t* number of servers nodes have to present on the ground every time, otherwise a new node cannot join network until *t* servers nodes available. Communication to *t* nodes increases the congestion in network.

Reference proposes a threshold cryptography based scheme suited for MANET to provide robustness and defense against single point of failure in the central server.But main drawback of threshold cryptography is the difficulty in applying the distributive function.

Bing Wua els propose a secure and efficient key management (SEKM) framework for MANETs. They build a public key infrastructure (PKI) by applying a secret sharing scheme and using an underlying multi-cast server groups. Problem with scheme is mobile nodes needs large storage to carry certificates and to perform public key cryptography based large computation.

S. Capkun et al suggested a method based on the users issuing certificates to each other based on personal acquaintance. These certificates are used to bind a public key and node identity. Every node should collect and maintain an up-to-date certificate repository. Certificate conflict is just another example of a potential problem inthis scheme.

In this paper, we proposed a CL-EKM which uses identity based cryptography for secure group communication in MANET. This scheme does not need any online certification authorities.

3. Overview of the Certificateless Effective Key Management Scheme

In this paper, we propose a Certificateless Key Management scheme (CL-EKM) that supports the establishment of four types of keys, namely: a certificateless public/private key pair, an individual key, a pairwise key, and a cluster key. This scheme also utilizes the main algorithms of the CL-HSC scheme in deriving certificateless public/private keys and pairwise keys. We briefly describe the major notations used in the paper (See Table I), the purpose of these keys and how they are setup.

• Certificateless Public/Private Key: Before a node is deployed, the KGC at the BS generates a unique certificateless private/public key pair and installs the keys in the node. This key pair is used to generate a mutually authenticated pairwise key.

• Individual Node Key: Each node shares a unique individual key with BS. For example, a *L*-sensor can use the individual key to encrypt an alert message sent to the BS, or if it fails to communicate with the *H*-sensor. An *H*-sensor can use its individual key to encrypt the message corresponding to changes in the cluster. The BS can also use this key to encrypt any sensitive data, such as compromised node information or commands. Before a node is deployed, the BS assigns the node the individual key.

• Pairwise Key: Each node shares a different pairwise key with each of its neighboring nodes for secure communications and authentication of these nodes. For example, in order to join a cluster, a *L*-sensor should share a pairwise key with the *H*-sensor. Then, the *H*-sensor can securely encrypt and distribute its cluster key to the *L*-sensor by using the pairwise key. In an aggregation supportive MANET, the *L*-sensor can use its pairwise key to securely transmit the sensed data to the *H*-sensor. Each node can dynamically establish the pairwise key between itself and another node using their respective certificateless public/private key pairs.

• Cluster Key: All nodes in a cluster share a key, named as cluster key. The cluster key is mainly used for securing broadcast messages in a cluster, e.g., sensitive commands or the change of member status in a cluster. Only the cluster head can update the cluster key when a *L*-sensor leaves or joins the cluster.

3.1. The Details of CL-EKM

The CL-EKM is comprised of 7 phases: *system setup*, *pairwise key generation*, *cluster formation*, *key update*, *node movement*, *key revocation*, and *addition of a new node*.

Table I : List of Notations

L_i	Unique identifier of an L-sensor node n_{L_i}
H_j	Unique identifier of an H-sensor node n_{H_j}
ID_{BS}	Identifier of the Base Station (BS)
q	A k bit primer number
P_{pub}	A system public key of KGC, $P_{pub} = xP$
x	A master private key of KGC
P	Point generator of an additive cyclic group G_q
$\mathbb{Z}^*_q$	The multiplicative group of integers modulo q
E/F_q	The selected elliptic curve over the field F_q: $y^2 = x^3 + ax + b \bmod q$, $a, b, x, y \in F_q$
pk_A	full public key of any node n_A, $pk_A = (P_A, R_A)$
sk_A	full private key of any node n_A, $sk_A = (d_A, x_A)$
K^0_A	Individual key of any node n_A
K_{AB}	Pairwise master key between n_A and n_B
k_{AB}	Pairwise encryption key between n_A and n_B
GK_j	Cluster key shared among the nodes in the j-th cluster
$\mathfrak{M}$	List of all the legitimate nodes in the network, maintained by the BS
$\mathfrak{R}$	List of revoked nodes, maintained by the BS
$HMAC(k, m)$	Message authentication code of m using key k
$E_k(m)$	A symmetric key encryption algorithm to encrypt a message m with a key k.

3.1.1. System Setup

Before the network deployment, the BS generates system parameters and registers the node by including it in a member list M.

1) Generation of System Parameters: The KGC at the BS runs the following steps by taking a security parameter $k \in Z+$ as the input, and returns a list of system parameter $\Omega = \{Fq , E/Fq , Gq , P, Ppub = x P, h0, h1, h2, h3\}$ and x.

- Choose a k-bit prime q
- Determine the tuple $\{Fq , E/Fq , Gq , P\}$.
- Choose the master private key $x \in_R Z^*q$ and compute the system public key $Ppub = x P$.
- Choose cryptographic hash functions : $\{h0, h1 , h2, h3\}$ so that $h0 : \{0, 1\}^* \times G2q \rightarrow \{0, 1\}^* , h1 : G3q \times \{0, 1\}^* \times Gq \rightarrow \{0, 1\}n, h2 : Gq \times \{0, 1\}^* \times Gq \times \{0, 1\}^* \times Gq \times \{0, 1\}^* \times Gq \rightarrow Z^*q$, and $h3 : Gq \times \{0, 1\}^* \times Gq \times \{0, 1\}^* \times Gq \times \{0, 1\}^* \times Gq \rightarrow Z^* q$.

Here, n is the length of a symmetric key.

The BS publishes Ω and keeps x secret.

2) Node Registration: The BS assigns a unique identifier, denoted by Li , to each L-sensor nLi and a unique identifier, denoted by Hj , to each H-sensor nHj, where $1 \leq i$

$\leq N1$, $1 \leq j \leq N2$, $N = N1 + N2$. Here we describe the certificateless public/private key and individual node key operations for *Li* , the same mechanisms apply for *H*-sensors. During initialization, each node *nLi* chooses a secret value $xLi \in R$ Z^*q and computes $PLi = xLi\ P$. Then, the BS requests the KGC for partial private/public keys of *nLi* with the input parameters *Li* and *PLi*. The *KGC* chooses $rLi \in R$ Z^*q and then computes a pair of partial public/private key *(RLi , dLi)* as below:

$RLi = rLi\ P$

$dLi = rLi + x \cdot h0(Li , RLi , PLi) \bmod q$

The *Li* can validate its private key by checking whether the condition $dLi\ P = RLi + h0(Li , RLi , PLi)Ppub$ holds. *Li* then sets *skLi = (dLi , xLi)* as its full private key and *pkLi = (PLi , RLi)* as its full public key. The BS also chooses a uniform random number $x0 \in Z^*q$ to generate the node's individual key *K0Li* (*K0Hj* for *nHj*). The individual key is computed as an HMAC of *x*0, *Li* as follows

$K0Li = HMAC(x0, Li)$

After the key generation for all the nodes, the BS generates a member list M consisting of identifiers and public keys of all these nodes. It also initializes a revocation list R that enlists the revoked nodes. The public/private key, Ω, and the individual key are installed in the memory of each node.

3.1.2. Pairwise Key Generation

After the network deployment, a node may broadcast an advertisement message to its neighborhood to trigger the pairwise key setup with its neighbors. The advertisement message contains its identifier and public key. At first, two nodes set up a long-term pairwise master key between them, which is then used to derive the pairwise encryption key. The pairwise encryption key is short-term and can be used as a session key to encrypt sensed data.

1) Pairwise Master Key Establishment: In this paragraph, we describe the protocol for establishing a pairwise master key between any two nodes *nA* and *nB* with unique IDs *A* and *B*, respectively.We utilize the CL-HSC scheme as a building block. When *nA* receives an advertisement message from *nB*, it executes the following encapsulation process to generate a long-term pairwise master key *KAB* and the encapsulated key information, $\phi A = (UA,WA)$.

- Choose $lA \in R$ Z^*q and compute $UA = lAP$.
- Compute

$TA = lA \cdot h0(B, RB, PB)Ppub + lA \cdot RB \bmod q$

$KAB = h1(UA, TA, lA \cdot PB, B, PB)$

- Compute

$h = h2(UA, \tau A, TA, A, PA, B, PB)$

$h' = h3(UA, \tau A, TA, A, PA, B, PB)$

$WA = dA + lA \cdot h + xA \cdot h'$

where τA is a random string to give a freshness.

- Output *KAB* and $\phi A = (UA,WA)$.

Then, *nA* sends *A*, *pkA*, τA and ϕA to *nB*. *nB* then performs ***decapsulation*** to obtain *KAB*.

- Compute $TA = dB \cdot UA$.

Note: Because of $dB = rB + x \cdot h0(B, RB, PB)$ and $UA = lAP \bmod q$, TA is computed as $TA = (rB + x \cdot h0(B, RB,B)) \cdot lAP \bmod q = lA \cdot h0(B, RB, PB)Ppub + lA \cdot RB \bmod q$,

- Compute $h = h2(UA, \tau A\ TA, A, PA, B, PB)$ and $h' = h3(UA, \tau A, TA, A, PA, B, PB)$.
- If $WA \cdot P = RA + h0(A, RA, PA) \cdot Ppub + h \cdot UA + h' \cdot PA$, output $KAB = h1(UA, TA, xB \cdot UA, B, PB)$. Otherwise, output invalid.

Table II: Cluster Formation Process

Node Discovery and Authentication

$n_{H_j} \rightarrow *$: $\langle H_j, pk_{H_j} \rangle$

(for $i = 1, \ldots, n$)

$n_{L_i} \leftrightarrow n_{H_j}$: Perform *Pairwise Key Generation* phase

Cluster Key Generation

(for $i = 1, \ldots, n$)

n_{H_j} : Generate GK_j, Compute $C_2 = E_{k_{L_i H_j}}(GK_j, H_j, L_i)$

$n_{H_j} \rightarrow n_{L_i}$: $\langle H_j, C_2 \rangle$

n_{L_i} : Decrypt C_2 to get GK_j and Compute $C_3 = E_{k_{L_i H_j}}(L_i, HMAC(k_{L_i H_j}, GK_j))$

$n_{L_i} \rightarrow n_{H_j}$: $\langle L_i, C_3 \rangle$

n_{H_j} : Decrypt C_3 and Check the validity

Membership Validation

n_{H_j} : Compute $C_4 = E_{K_{H_j}^0}(H_j, \mathfrak{M}_j)$, $C_5 = E_{GK_j}(H_j, \mathfrak{M}_j)$

$n_{H_j} \rightarrow$ BS : $\langle H_j, C_4 \rangle$

BS : Check $\mathfrak{M}_j$

BS $\rightarrow n_{H_j}$: $\langle Acknowledgement \rangle$

$n_{H_j} \rightarrow *$: $\langle C_5 \rangle$

2) Pairwise Encryption Key Establishment : Once *nA* and *nB* set the pairwise master key *KAB*, they generate an HMAC of *KAB* and a nonce $r \in R$ Z^*q . The HMAC is then validated by both *nA* and *nB*. If the validation is successful, the HMAC value is established as the short-term *pairwise encryption key kAB*. The process is summarized below:

- *nB* chooses a random nonce $r \in R$ Z^*q , computes $kAB = HMAC(KAB, r)$ and $C1 = EkAB (r, A, B)$. Then, *nB* sends *r* and *C*1 to *nA*.
- When *nA* receives *r* and *C*1, it computes $kAB = HMAC(KAB, r)$ and decrypts *C*1. Then it validates *r* , *A* and *B* and if valid confirms that *nB* knows *KAB* and it can compute *kAB*.

3.1.3. Cluster Formation

Once the nodes are deployed, each *H*-sensor discovers neighboring *L*-sensors through *beacon* message exchanges and then proceeds to authenticate them. If the authentication is successful, the *H*-sensor forms a cluster with the authenticated *L*-sensors and they share a common cluster key. The *H*-sensor also establishes a pairwise key with each member of the cluster. To simplify the discussion, we focus on the operations within one cluster and consider the *j* th cluster. We also assume that the cluster head *H*-sensor is *nHj* with $nLi\ (1 \leq i \leq n)$ as cluster members. *nHj* establishes a cluster key *GKj* for secure communication in the cluster. Table II shows the cluster formation process.

1) Node Discovery and Authentication: For node discovery, *nHj* broadcasts an advertisement message containing *Hj* and *pkHj*. Once *nLi* within *Hj* 's radio range receives the advertisement, it checks *Hj* and *pkHj* , and initiates the *Pairwise Key Generation* procedure. Note that *nLi* may receive multiple advertisement messages if it is

within the range of more than one *H*- sensor. However, *nLi* must choose one *H*-sensor, may be by prioritizing over the proximity and signal strength. *Additionally, nLi can record other H-sensor advertisements as backup* cluster heads in the event that the primary cluster head is disabled. If *nLi* selects multiple cluster heads and sends a response to all of them, it is considered as a compromised node. *nLi* and *nHj* perform the *Pairwise Key Generation* procedure to obtain a pairwise master key, *KLi Hj* and a pairwise encryption key, *kLi Hj* .

2)Cluster Key Generation: nHj chooses $x_j \in_R Z_q^*$ to generate a cluster key *GKj* as follows

$GKj = HMAC(x_j , Hj)$

Then, *nHj* computes $C2 = EkLi\ Hj\ (GKj , Hj , Li)$ to distribute the *GKj*. Then *nHj* sends *Hj* and *C*2 to *nLi* . *nLi* decrypts *C*2 to recover *Hj , Li* and *GKj* by using *kLi Hj*. If *nLi* fails to check *H j , Li* , it discards the message and reports *nHj* to the BS as an illegitimate cluster head. Otherwise, *nLi* confirms that *nHj* is valid and can compute *GKj* . Then, *nLi* stores *GKj* as a cluster key. Next, *nLi* computes *HMAC(k Li H j ,GK j)* and $C3 = Ek\ Li\ Hj\ (Li, HMAC\ (kLi\ Hj\ ,GKj\ ))$. It transmits *C*3 and *Li* to *nHj*. After *nHj* receives messages from *nLi* , it decrypts *C*3 by using *kLi Hj* . Then it checks *Li* and the validity of *HMAC(kLiHj,GK j)*. If the validity check fails, *nHj* discards the message. Otherwise, *nHj* can confirm that *nLi* shares the valid *GKj* and *kHj Li* . *nHj* adds *Li* and *pkLi* on member list of the *j* th cluster, M*j* .

3) Membership Validation: After discovering all the neighboring nodes *nLi* $(1 \le i \le n)$ in the *j* th cluster, *nHj* computes $C4 = EK0H\ j\ (Hj\ ,Mj)$ and transmits *C*4 and *Hj* to the BS. After receiving messages from *nHj* , the BS checks the validity of the nodes listed in M*j* . If all nodes are legitimate, the BS sends an acknowledgement to *nHj* . Otherwise, the BS rejects M*j* and investigates the identities of invalid nodes (false or duplicate ID). Then, the BS adds the identities of invalid nodes to the revocation list and reports it to *nHj* . Upon receiving the acknowledge message, *nHj* computes $C5 = EGKj\ (Hj\ ,Mj)$ and broadcasts *C*5 to all the nodes in *j* th cluster.

3.1.4. Key Update

In order to protect against cryptanalysis and mitigate damage from compromised keys, frequent encryption key updates are commonly required. In this section we provide the pairwise key update and cluster key update operations.

1) Pairwise Key Update: To update a pairwise encryption key, two nodes which shared the pairwise key perform a *Pairwise Encryption Key Establishment* process. On the other hand, the pairwise master key does not require periodical updates, because it is not directly used to encrypt each session message. As long as the nodes are not compromised, the pairwise master keys cannot be exposed. However, if a pairwise master key is modified or needs to be updated according to the policy of the BS, the *Pairwise Master Key Establishment* process must be executed.

2) Cluster Key Update:Only cluster head *H*-sensors can update their cluster key. If a *L*-sensor attempts to change the cluster key, the node is considered a malicious node. The operation for any *j* th cluster is described as follows: 1) *nHj* chooses $x'_j \in_R Z_q^*$ and computes a new cluster key $GK'j = HMAC(x'j , Hj)$. *nHj* also generates an *Update* message including *HMAC(GK',Update)* and computes $C6 = EGKj\ (GK', HMAC(GK'j\ ,Update))$. Then, *nHj* transmits *Update* and *C*6 to its cluster members. 2) Each member *nLi* decrypts *C*6 using the *GKj* , verifies *HMAC(GK'j ,Update)* and updates a cluster key as *GK'j* . Then, each *nLi* sends the acknowledgement message to *nHj*

3.1.5. Node Movement

When a node moves between clusters, the *H*-sensors must properly manage the cluster keys to ensure the forward/backward secrecy. Thus, the *H*-sensor updates the cluster key and notifies the BS of the changed node status. Through this report, the BS can immediately update the node status in the M. We denote a moving node as *nLm* .

1) Node Leave: A node may leave a cluster due to node failure, location change or intermittent communication failure. There are both proactive and reactive ways for the cluster head to detect when a node leaves the cluster. The proactive case occurs when the node *nLm* actively decides to leave the cluster and notifies the cluster head *nHj* or the cluster head decides to revoke the node. Since in this case *nHj* can confirm that the node has left, it transmits a report $EK0Hj\ (NodeLeave, Lm)$ to inform the BS that *nLm* has left the cluster. After receiving the report, the BS updates the status of *nLm* in M and sends an acknowledgement to *nHj* . The reactive case occurs when the cluster head *nHj* fails to communicate with *nLm*. It may happen that a node dies out of battery power, fails to connect to *nHj* due to interference or obstacles, is captured by the attacker or is moved unintentionally. Since the nodes in a cluster periodically exchange lightweight *beacon* messages, *nHj* can detect a disappeared node *nLm* when it does not receive the *beacon* message from *nLm* for a predetermined time period. So, *nHj* reports the status of the node *nLm* to the BS by sending $EK0H\ j\ (NodeDisappear,Lm)$. When the BS receives the report, it updates the status of *nLm* in the M and acknowledges to *nHj*. Once *nHj* receives the acknowledgement from the BS, it changes its cluster key with the following operations: 1) *nHj* chooses a new cluster key *GK'j* and computes *EkLi Hj (GK'j , NodeLeave, Lm)* using pairwise session keys with each node in its cluster, except *nLm* . 2) Then, *nHj* sends *EkLi Hj (GK'j , NodeLeave, Lm)* to each member node except *nLm* . 3) Each *nLi* decrypts it using *kLi Hj* and updates the cluster key as *GK'j* .

2) Node Join: Once the moving node *nLm* leaves a cluster, it may join other clusters or return to the previous cluster after some period. For the sake of simplicity, we assume that *nLm* wants to join the *l*th cluster or return to the *j* th cluster.

(i) Join a New Cluster: *nLm* sends a join request which contains *Ln*+1 and *pkLn*+1 to join a *l*th cluster. After *nHl* receives the join request, *nLm* and *nHl* perform *Pairwise Key Generation* procedure to generate *KLm Hl* and *kLm Hl* ,respectively. Next, *nHl* transmits $EK0H\ l\ (NodeJoin, Lm)$ to the BS. The BS decrypts the message and validates whether *nLm* is a legitimate node or not and sends an acknowledgement to *nHl* if successful. The BS also updates the node member list, M. In case of node validation failure at the BS, *nHl* stops this processand revokes the

pairwise key with *nLm*. Once *nHl* receives the acknowledgement, it performs the *Cluster Key Update* process with all other nodes in the cluster.*nHl* also computes *EkLm Hl (GK'l , Hl , Lm)*, and sends it to the newly joined node *nLm* .

(ii) Return to the Previous Cluster: *nLm* sends a join request which contains *Ln*+1 and *pkLn*+1 to join a *j* th cluster. Once *nHj* receives the join request, it checks a timer for *nLm* which is initially set to the *Thold* . *Thold* indicates the waiting time before discarding the pairwise master key when a L-sensor leaves. If *nLm* returns to the *j* th cluster before the timer expires, *nLm* and *nHj* perform only the *Pairwise Encryption Key Establishment* procedure to create a new pairwise encryption key, *k_LmHj*. Otherwise, they perform the *Pairwise Key Generation* procedure to generate a new *K'LmHl* and *k_ LmHl* , respectively. Then, the cluster head *nHj* also updates the cluster key to protect backward key secrecy. Before updating the cluster key, *nHj* transmits *EK*0*H j (NodeReJoin, Lm)* to the BS. Once the BS decrypts the message and determines that *nLm* is a valid node, the BS sends the acknowledgement to *nHl* . The BS then updates the member list M. Once *nHl* receives the acknowledgement, it performs the *Cluster Key Update* process with all other nodes in the cluster. Afterwards, *nHj* computes *Ek_ Lm Hj(GK_j , Hj , Lm)* and sends it to *nLm* .

3.1.6. Key Revocation

We assume that the BS can detect compromised *L-sensors* and *H-sensors*. The BS may have an intrusion detection system or mechanism to detect malicious nodes or adversaries. Although we do not cover how the BS can discover a compromised node or cluster head in this paper, the BS can utilize the updated node status information of each cluster to investigate an abnormal node. In our protocol, a cluster head reports the change of its node status to the BS, such as whenever a node joins or leaves a cluster. Thus, the BS can promptly manage the node status in the member list, M. For instance, the BS can consider a node as compromised if the node disappears for a certain period of time. In that case, the BS must investigate the suspicious node and it can utilize the node fault detection mechanism introduced in. In this procedure, we provide a key revocation process to be used when the BS discovers a compromised node or a compromised cluster head. We denote a compromised node by *nLc* in the *j* th cluster for *a compromise node case* and a compromised head by *nHj* for *a compromise cluster head case*.

1) Compromised Node: The BS generates a *CompNode* message and a *EK*0*HJ (CompNode, Lc)*. Then it sends *EK*0*H j (CompNode, Lc)* to all *nHj , (*1 ≤ *j* ≤ *N*2*)*. After all *H*-sensors decrypt the message, they update the revocation list of their clusters. Then, if related keys with *nLc* exist, the related keys are discarded. Other than *nLc* , *nHj* performs the *Node leave* operations to change the current cluster key with the remaining member nodes.

2) Compromised Cluster Head: After the BS generates a *CompHeader* message and a *EK*0*I (CompHeader, Hj)*, it sends the message to all *nLi(*1 ≤ *i* ≤*n)* in the *j* th cluster. The BS also computes *EK*0*H i (CompHeader, Hj), (*1 ≤ *i* ≤ *N*2, *i* ≠ *j)* and transmits it to all *H*-sensors except *nHj*. Once all nodes decrypt the message, they discard the related keys with *nHj* . Then, each *nLi* attempts to find other neighboring cluster heads and performs the *Join other cluster* steps of the *Node join* process with the neighboring cluster head. If some node *nLi* is unable to find another cluster head node, it must notify the BS by sending *EK*0*L i (FindNewClusteLi)*. The BS proceeds to find the nearest cluster head *nHn* for *nLi* and connect *nHn* with *nLi* . Then, they can perform the *Join other cluster* steps.

3.1.7. Addition of a New Node

Before adding a new node into an existing networks, the BS must ensure that the node is not compromised. The new node *nLn*+1 establishes a full private/public key through the *node registration* phase. Then, the public system parameters, a full private/public key and individual key *K*0*L n*+1 are stored into *nLn*+1 . The BS generates *EK*0*H j(NewNode, Ln*+1*, pkLn*+1*)* and sends it to all *nHj , (*1 ≤ *j* ≤ *N*2*)*. After *nLn*+1 is deployed in the network, it broadcasts an advertisement message which contains *Ln*+1 and *pkLn*+1 to join a neighboring cluster. If multiple *H*-sensors receive *nLn*+1's message, they will transmit a Response message to *nLn*+1 . *nLn*+1 must choose one *H*-sensor for a valid registration. If *nLn*+1 selects *nHj* according to the distance and the strength of signal, it initiates the *Pairwise Key Generation* procedure. In order to provide backward secrecy, *nHj* performs *Cluster Key Update* procedure, where the *Update* message contains *Ln*+1 and *pkLn*+1. Then, *nHj* computes *C*7 = *EkLn*+1 *Hj (GK'j , Hj , Ln*+1*)*, and sends *C*7 and *Hj* to *nLn*+1. After *nLn*+1's registration, *nHj* transmits *EK*0*H j (Nodejoin,n*+1*)* to the BS. Once the BS decrypts the message, it updates the status of the node *nLn*+1 in member list, M.

4. Conclusions

In this paper, we propose the first certificateless effective key management protocol (CL-EKM) for secure communication in MANETs. CL-EKM supports efficient communication for key updates and management when a node leaves or joins a cluster and hence ensures forward and backward key secrecy. The CL-EKM is resilient against node compromise, cloning and impersonation attacks and protects the data confidentiality and integrity. The experimental results demonstrate the efficiency of CL-EKM in resource constrained MANET.

References

[1] Mishra. A, Nadkarni, K, and Patcha, A," Intrusion Detection in wireless ad hoc networks", IEEE Journal Personal Communication on wireless communication, 11(1), 48-60, 2004.

[2] Patwardhan, A., Parker, joshi, A., Iorga, M. & karygiannis, T. , " Routing and intrusion detection in ad hoc networks" in proceeding of the 3rd IEEE international conference on pervasive computing and communication, pp. 191-199, Kauai Island, USA, 2005.

[3] M. R. Alagheband and M. R. Aref, "Dynamic and secure key manage-ment model for hierarchical heterogeneous sensor networks," IET Inf. Secure., vol.

6, no. 4, pp. 271–280, Dec. 2012.

[4] M. Rahman and K. El-Khatib, "Private key agreement and secure communication for heterogeneous sensor networks," J. Parallel Distrib. Comput., vol. 70, no. 8, pp. 858–870, 2010.

[5] Schmoyer, T. R. Lim, Y. X. & Owen, H. L.," wireless intrusion detection and response: a classic study of man in the middle attack", proceedings of the 2004 international conference on wireless communications and networking, pp8830888, Atlanta, USA, 2004.

[6] D. S. Sanchez and H. Baldus, "A deterministic pairwise key pre-distribution scheme for mobile sensor networks," in Proc. 1st Int. Conf. SecureComm, Sep. 2005, pp. 277–288.

[7] I.-H. Chuang, W.-T. Su, C.-Y. Wu, J.-P. Hsu, and Y.-H. Kuo, "Two-layered dynamic key management in mobile and long-lived cluster-based wireless sensor networks," in Proc. IEEE WCNC, Mar. 2007,pp. 4145-4150.

[8] S. Agrawal, R. Roman, M. L. Das, A. Mathuria, and J. Lopez, "A novel key update protocol in mobile sensor networks," in Proc. 8th Int. Conf. ICISS, vol. 7671. 2012, pp. 194–207.

[9] R. Blom, "Optimal class of symmetric key generation systems:, proceeding of EUROCRYPT 84 workshop on Advances in Cryptology: theory and application of cryptographic techniques, pp 335-338, December 1985, France.

[10] H. Nam Nguyen, H. Morino, "A key management scheme for mobile ad hoc network based on Threshold Cryptography for providing fast Authentication and low signaling load", EUC Workshop 2005, LNCS 3823, pp. 905-915, 2005.

[11] Bing Wu, Jie Wu, Eduardo B. Fernandez, Spyros Magliveras, "Secure and Efficient key management in mobile adhoc networks", proceeding of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) IEEE 2005.

[12] S. Capkun, L. Buttyan and J-P Hubaux, "Self organized public key management for mobile adhoc networks", IEEE Transaction on Mobile Computing,2(1), January-March 2003.

[13] Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, p:102-110,610-612, 1996.

[14] Rafeali S. and Hutchison D., "A survey of key management for secure group communication" ACM Computing Surveys, 309-329, 2003.

[15] Daniel Bleichenbacher, Alexander, "May: New Attacks on RSA with Small Secret CRT-Exponents", Public Key Cryptography, pages 1-13, 2006.