An Efficient Key Revocation for Identity Based Key Management in MANET

Pragnesh Patel  
Computer Engineering Department,  
L. D. College of Engineering,  
Ahmedabad, Gujarat, India  
pragnesh.4400@gmail.com

Vimalkumar B. Vaghela  
Assistant Professor  
Computer Engineering Department,  
L. D. College of Engineering,  
Ahmedabad, Gujarat, India  
vimalvaghela@gmail.com

Purvi Ramanuj  
Assistant Professor  
IT Department,  
Shantilal Shah Engineering College,  
Bhavnagar, Gujarat, India  
purviramanuj@yahoo.com

Abstract

Today Mobile Ad Hoc Networks (MANETs) are secured using the identity based cryptographic (IBC) scheme due to their efficient key management properties. Existing key revocation scheme performs basic three functionality like, Neighbourhood Watch, Harakiri message, Accusation Scheme. In accusation scheme each node maintains a key revocation list (KRL) by monitoring node in their communication range and securely propagates its observations. In existing key revocation scheme each node sends entire key revocation list to their neighbour nodes for propagating minor update in the key revocation list. So existing key revocation scheme increases the required computational time to update the key revocation list at receiver end. In this paper we have proposed a scheme which will reduce the computational time to update KRL at receiver end by sending only updated portion of the list to their neighbour nodes.

Keywords— MANET, Identity Based Cryptographic (IBC), Identity Based Key Management, Key Revocation List (KRL).

I. INTRODUCTION

As there is rapid increase in development of mobile wireless applications which requires that network should be formed spontaneously by participating devices themselves. Such networks are referred to as a Mobile Ad Hoc Networks. The concept of MANETs is to enable connectivity between any arbitrary groups of mobile devices everywhere at anytime. It is very tough task to implement security in MANETs due its special properties like, lack of infrastructure, absence of trusted third parties, and constrains about devices and communication channels. It is quite difficult to achieve secure key distribution using symmetric schemes in MANETs. So use of public key infrastructure (PKI) is desirable in MANETs. But use of PKI leads to many computational and communication overload on network in ad hoc environment.

Now days Identity Based Schemes (IBC) is a good alternative to secure ad hoc networks due to their efficient key management key management. Existing key management scheme has key revocation and renewal schemes to meet the requirements and challenges of MANETs. In existing key revocation scheme each node uses neighbourhood watch scheme to monitor nodes in communication range for suspicious behaviour. This observations are securely propagates to m-hop neighbourhood. The public key of node is revoked if it receives at least threshold no of accusation against the node.

But the existing scheme have its own limitation like, in existing scheme each node entire key revocation list to their neighbour node for propagating minor update in the list. This increases computational time at receiver end for updating the key revocation list. The propose scheme in this paper tries to reduce computational time at receiver end.

The sections of paper are organized as follows: In section II we discuss about Existing Revocation Scheme and their limitation. Section III describes proposed scheme. Section IV Implementation and Results and In Section V drawn some conclusion.

II. EXISTING SCHEME

The existing key revocation scheme is based on IBC from the well pairing. The IBC scheme provides efficient key management that helps in reducing computational, communicational and memory cost. The necessary assumptions for network and its nodes in IBC key revocation scheme can be summarized as follows: 1. Bidirectional communicational links, 2. Nodes are in promiscuous mode, 3. Each node has a unique identity, 4. Nodes know identity of their one-hop neighbours. 5. Node obtains public and private key pair from offline-KGC prior to joining the network.

Every node in MANETs needs to be able to verify whether a public key is revoked. General schemes designed for MANETs, revocation referred to embedding expire date in public key. But it is not sufficient because sometimes nodes need to revoke their keys before they expire. In existing scheme keys are revoked either if a node notices that its own keys has been compromised or if a group of at least threshold no of nodes observes that another node behaves suspiciously.

In order to provide key revocation in IBC schemes in MANETs, existing scheme had introduced three algorithms. First, nodes observe the nodes in their neighbourhood for suspicious behaviour using Algorithm 1: Neighbourhood watch. Second, nodes need to able to revoke their own public keys using Algorithm 2: Harakiri. Third, nodes securely inform each other about suspicious observations and generate key revocation lists in Algorithm 3: Accusation scheme.

---

Available online@www.ijcta.com
Alg.1 Neighbourhood watch \cite{3}

The neighbourhood watch scheme is local monitoring scheme, in which node monitors their 1-hop neighbour nodes for suspicious behaviour. Suspicious behaviour could be frequent packet drops or large no of packets sends. This suspicious behaviour could be due to node has been compromised and control by other node or node becomes selfish.

For an easier representation and without loss of generality, we denote ID_i's one-hop neighbours as ID_j ∈ N_i with j ∈ {1, ..., N_i} where N_i is the number of one-hop neighbours. User ID_i stores so-called accusation values a_i,j for each ID_j ∈ N_i together with the expiry date t_i,j and version number v_i,j of the current public key Q_j. A node ID_i sets its accusation values a_i,j = 1 if ID_j observed ID_i to behave suspiciously, otherwise a_i,j = 0. The accusation values that a node ID_i creates from its neighborhood watch can be represented as an accusation matrix:

\[
AM^i = \begin{pmatrix}
ID_1 & (t_i^1, v_i^1) & a_{i,1}^1 \\
\vdots & \vdots & \vdots \\
ID_{N_i} & (t_i^{N_i}, v_i^{N_i}) & a_{i,N_i}^{N_i}
\end{pmatrix}
\]

With a_{i,j} ∈ \{0,1\} and j ∈ \{1, ..., N_i\}.

Each row vector r^i(ID_j) in AM^i, we use r_i^j for short in the rest of the paper, corresponds to a neighbour ID_j ∈ N_i and the accusation values a_{i,j} for the current public key Q_j with expiry date t_i,j and version number v_i,j. We refer to the third column in AM^i as column vector c_i^j, which is the vector that contains all accusations made by ID_i. The accusation values are updated every time ID_i suspicious behaviour. Once the a_{i,j} is set, the value will not be reset to zero until a new public key Q_j is received.

Alg.2 Harakiri \cite{3}

When a node ID_i realizes that its private key d_i has been compromised, it broadcasts a harakiri message hm_i, with

\[
hm_i = (ID_i, Q_i, d_i, (t_i, v_i), "revoke", hopcount)
\]

To its m-hop neighbourhood m-N_i. Node ID_i initially sets hop count = m and sends the message to all one-hop neighbours N_i. The receivers ID_j verify if the harakiri is authentic, by checking whether below equation is true,

\[
K_j^i = h(d_i, Q_j)
\]

The check verifies whether the broadcasted private key d_i really corresponds to the public key Q_j. Therefore, a recipient of hm_i, say node ID_j, looks up whether it is in possession of the public key Q_j and the pre-shared key K_j^i and if so, uses the K_j^i to check whether above equation is true. If ID_j is not in the possession of these keys, ID_j first computes Q_j according to (2), to check whether ID_i and Q_j correspond to each other. If this check is successful, ID_j derives K_j^i according to (1) and then checks whether (4) is true. If (4) is true, the receive ID_j updates its accusation value a_{j,i} = 1, decrements hop count and broadcasts the message again. Otherwise, hm_i is discarded. The broadcasting is repeated until hop count = 0. This ensures that all nodes in an m-hop neighbourhood of the compromised node ID_i receive the harakiri message and thus learn about the key compromise.

Alg.3 Accusation scheme \cite{3}

In this algorithm every node ID_i creates its own key revocation list KRL_i for its m-hop neighbourhood. In this paragraph we will describe how revocation lists are created (Alg.3.1), securely propagated (Alg.3.2) and how nodes use received revocation lists and harakiri messages to update their own revocation lists (Alg.3.3).

Alg.3.1 creating a key revocation list KRL: Each node ID_i creates a key revocation list KRL_i of the following format:

\[
KRL_i = \begin{pmatrix}
ID_i & (t_i, v_i) & R_i^1 a_{i,1}^{1} & \ldots & R_i^{M_i} a_{i,M_i}^{M_i} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
ID_{M_i} & (t_{M_i}, v_{M_i}) & R_{M_i} a_{M_i,1}^{M_i} & \ldots & a_{M_i,M_i}^{M_i}
\end{pmatrix}
\]

With a_{i,j,k} ∈ \{0,1,−\} and j,k ∈ \{1, ..., M_i\} where M_i is the number of nodes in ID_i’s m-hop neighbourhood m-Ni, including ID_i itself. The accusation value a_{i,j,k} = − indicates that ID_j and ID_k are more than m hops away from each other, and thus cannot give a statement about each others trustworthiness. Each row vector r^i(ID_j), short r_i^j, in KRL_i corresponds to a node ID_j ∈ m − N_i, where the row contains the accusation value a_{i,j,k} from all nodes ID_k ∈ m − N_i against a node ID_j. Each column vector c_i^j(1(ID_j)), short c_i^j, in KRL_i contains all the accusations a_{i,j,k} made by node ID_j against all nodes ID_k ∈ m − N_i. The index i denotes that the values are the current values in ID_i’s KRL_i, where other nodes might have different values. For example, a_{i,j,k} ≠ a_{i,j,k} for i ≠ l in some cases. Discrepancies in accusation values can exist, because accusation values may be more or less up to date, and nodes have different m-hop neighbourhoods, and thus receive different accusation and harakiri messages.

The first field in each row r_i^j in KRL_i contains the identity of node ID_j, the second field the expiry date t_i^j and version number v_i^j of the most recent key Q_j that ID_i
knows of. The fields $4 - (M_i + 3)$ contain the accusation values $a_{j,k}^i$ - $a_{j,M_i}$, where value $a_{j,k}^i = 1$ indicates that node $ID_k$ accused node $ID_j$, and $a_{j,k}^i = 0$ otherwise. The third field contains a 1-bit flag $R_j^i$ that, when set, indicates that node $ID_i$ considers the public key $Q_j$ of node $ID_j$ as revoked. The revocation flag $R_j^i$ in $ID_i$'s key revocation list $KRL_i$ are set, i.e. $R_j^i = 1$, if one of the following four conditions is true:

(Cond.1): $a_{j,i}^i = 1$, i.e. node $ID_i$ observed itself the malicious behaviour of node $ID_j$ during the neighbourhood watch (Alg.1). This follows that $ID_i$ and $ID_j$ are 1-hop neighbours.

(Cond.2): $t_j^i$ is expired, i.e. the current copy of the $ID_j$'s public key $Q_j$ is expired.

(Cond.3): $a_{i,j}^i = 1$, i.e. $ID_i$ received an authentic harakiri message $hm_j$ from $ID_j$.

(Cond.4): $A_j^i = \sum_{k=1}^{M_i} a_{j,k}^i > \delta$ for all $k, s.t R_k^i = 0$.

i.e. add all accusation values $a_{j,k}^i$ of row vector $t_j^i$ from non-revoked nodes $ID_k$ and check whether the sum is greater than $\delta$. In other words, the public key $Q_j$ is revoked if node received more than $\delta$ accusations from trustworthy nodes $ID_k$ for a suspicious node $ID_j$. Note that "-“is treated as zero value in the sum. If none of the four conditions applies, $R_j^i = 0$, i.e. $ID_j$ and its current public key $Q_j$ are considered to be trustworthy.

Alg.3.2 propagating accusations: In this algorithm nodes securely propagate accusations through the network. Every time a node $ID_i$ updates its accusation matrix $AM_i^i$ because it observed some suspicious behaviour in its neighbourhood watch, $ID_i$ sends an accusation message to its one-hop neighbours. Similarly, every time $ID_i$ updates its key revocation list $KRL_i$ because it received accusations from other nodes, $ID_i$ sends an accusation message to its neighbours. The accusation messages send by $ID_i$ have the following format:

$$am_{r,j} = (f_{s,i}(ID_s, am_i), (ID_j, am_j)) \text{ for all } D_j \in N_i$$

Where $am_i = AM_i^i$ for updates from the neighbourhood watch of $ID_i$, and $am_i = KRL_i^i$ for updates of the revocation list caused by accusation messages that $ID_i$ received from other nodes. Optionally, $am_i$ contains only the updated values to reduce bandwidth. The accusation messages $am_{i,j}$ are secured using the pre-shared keys $K_{i,j}$ for all $ID_j \in N_i$ and

then unicast to each one-hop neighbour $ID_j$. The pre-shared keys serve as input in a secure hash function $f(\cdot)$ to authenticate the message. Upon receiving $am_{i,j}$, a neighbouring node $ID_j$ verifies the received message using its pre-shared key $K_{i,j}$. If the verification is successful, $ID_j$ updates its key revocation list $KRL_j^i$ accordingly, as we will explain in the next paragraph.

Alg.3.3 Updating key revocation lists: Every time a node $ID_i$ receives a harakiri message $hm_j$ from $ID_j$, the node verifies the message as described in Alg.2 and if this verification is successful, node $ID_i$ sets $a_{i,j}^i = 1$ and thus $R_j^i = 1$ in its revocation list $KRL_i$. If a node $ID_i$ receives an accusation message $am_{i,j}$ of an one-hop neighbour $ID_j$, node $ID_i$ performs the following steps to update its key revocation list $KRL_i$.

(Step 1): check whether $R_j^i = 0$, i.e. whether $ID_i$ considers $ID_j$ being trustworthy; if yes continuing, else discard $am_{i,j}$ and stop.

(Step 2): verify authenticity of $am_{j,i}$ using the pre-shared key $K_{j,i}$ as described in Alg.3.2; if verification is successful continue, else discard $am_{j,i}$ and stop.

(Step 3): extract column vector $c_j^i$ of $AM_i^i$ or $KRL_i^i$ from $am_{j,i}$ to update column vector $c_j^i$ in $KRL_i^i$ i.e. adopt the accusation values from $ID_j$'s neighbourhood watch. Note that $ID_i$ uses only accusation values that are addressed to nodes in $ID_i$'s own m-hop neighbourhood, other accusation values are discarded.

(Step 4): discard all columns $c_k^i$ from $KRL_i^i$ for: $k = i$ because that is $ID_i$'s own accusation vector; $k = j$ because that was used in step 3; $k = 1$ for all $R_j^i = 1$ with $l \in \{1, \ldots , M_i\}$ because $ID_i$ does not trust nodes $ID_l$; $k = r$ for all $am_{r,j}$ that were accepted in step 2; $k = s$ for all $ID_s \notin N_i$, i.e. nodes that are more than m hops away. Save all other columns $c_k^i$.

Now $ID_i$ repeats steps 1-4 for all received accusation messages $am_{j,i}$. Let’s say $ID_i$ saved $d_k$ column vectors $c_k^i$ from $d_k$ different nodes $ID_j$ for the same $ID_k$ in step 4. Now for every $d_k$ , $ID_i$ performs step 5 below.

(Step 5): use all $d_k$ saved column vectors $c_k^i$ from step 4 to update $D_j$’s column vector $c_j^i$ in $KRL_i^i$. The update is done by using the majority vote for each element in the column
vector, i.e. the majority for each accusation value \( a_{l,k}^j \) with \( l \in \{1, ..., M_i \} \) is computed. For simplicity, we assume \( ID_l \) saved \( d_k \) column vectors \( c_k^j \) from \( d_k \) neighbors \( ID_j \) with \( j \in \{1, ..., d_k \} \) in step 4. Now \( ID_l \) computes,

\[
a_{l,k}^j = \begin{cases} 
1 & \text{if } \sum_{j=1}^{d_k} a_{l,k}^j > \frac{d_k}{2} \text{ with } l \in \{1, ..., M_i \} \text{ and } j \in \{1, ..., d_k \} \\
0 & \text{if } \sum_{j=1}^{d_k} a_{l,k}^j < \frac{d_k}{2} \text{ with } l \in \{1, ..., M_i \} \text{ and } j \in \{1, ..., d_k \} \\
\text{otherwise}
\end{cases}
\]

Again, only values \( a_{l,k}^j \) for nodes \( ID_l \) that are within \( d_k \)'s m-hop range are considered, others are discarded. If no majority can be found, the accusation value in KRL remains unchanged. Node \( ID_l \) repeats this for all column vectors \( c_k \).

Though the above described key revocation scheme is efficient it has its own limitations. In existing scheme malicious node can overhear accusation against them and use this knowledge to keep accusation against them below threshold. The harakiri message contains the private key which affects the security of all previous messages those are either signed under the private key or encrypted under public key. In this existing scheme each node send entire key revocation list to their neighbour nodes for propagating minor updates in the list, so it will definitely increase the computational time to update the key revocation list at receiver end.

As we have seen the limitation of existing scheme above in this article I am going to propose a scheme that may increase the efficiency of the key revocation algorithm. As described above in existing scheme each node sends entire key revocation list to their neighbour nodes for propagating minor updates in the list, so it may decrease the computational time at receiver end for updating the key revocation list. In the next section we are going to discuss the proposed scheme methodology and its algorithm.

### III. PROPOSED SCHEME

The basic idea of proposed scheme is to reduce computational time for updating key revocation list at receiver end. The efficiency of proposed scheme will increased only by sending updated accusation values instead of entire key revocation list. So the proposed key revocation list update algorithm will undergo less iteration than the existing one. And computational time will be saved for that iteration. The proposed scheme methodology and flowchart is as show below.

Let \( ID_l \) received an accusation message \( msg_i \) from

**Flowchart:**

```
Received message from node i

\( R_l^i = 0 \)

Yes

\( R_l^i \)

drop

No

Extract column value of \( c \) from list to update & create list

Set \( k = 1 \)

\( k < M_j \)

Yes

\( k \neq i, k \neq j \), \( R_j^i = 0 \), \( k \in N_j, k \in M_j \)

\( k++ \)

No

Update column of \( c^j \) according to majority & send

Store column value of \( c^j \), \( k++ \), \( d_k \)

Yes

Create message & send

Set \( k = 1 \)

\( k < M_j \)

No

```

**Step 1.** If \( msg_i \) is neighborhood watch or update message check \( R_l^i \).

**Step 2.** If \( R_l^i \) is one then drop the packet else verify it.

**Step 3.** Update key revocation list and create entry in update list.

**Step 4.** Create an update message having updated list instead of key revocation list, send message to neighbours.
Different Notation used in Paper\(^{[8]}\)

<table>
<thead>
<tr>
<th>Notations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N)</td>
<td>Set of all network nodes.</td>
</tr>
<tr>
<td>(A_M^i)</td>
<td>Accusation Matrix of node (i)</td>
</tr>
<tr>
<td>(a_{i,j}^i)</td>
<td>Accusation value observed by node (i) for node (j)</td>
</tr>
<tr>
<td>(N_i)</td>
<td>No of 1-hop neighbours of node (i)</td>
</tr>
<tr>
<td>(h_{mi})</td>
<td>Harakiri message send by node (i)</td>
</tr>
<tr>
<td>(Q_i)</td>
<td>Public key of node (i)</td>
</tr>
<tr>
<td>(d_i)</td>
<td>Private key of node (i)</td>
</tr>
<tr>
<td>(KRL_i)</td>
<td>Key Revocation List of node (i)</td>
</tr>
<tr>
<td>(ID_i)</td>
<td>Identity of node (i)</td>
</tr>
<tr>
<td>(M_i)</td>
<td>No of (m)-hop neighbour of node (i)</td>
</tr>
<tr>
<td>(a_{mi,j})</td>
<td>Accusation message sent by node (i) to node (j)</td>
</tr>
<tr>
<td>(c_k^i)</td>
<td>(k^{th}) column vector of node (i)'s key revocation list</td>
</tr>
<tr>
<td>(d_k)</td>
<td>Counter for received (k) vectors.</td>
</tr>
<tr>
<td>(R_j^i)</td>
<td>1-bit flag , when set it indicates that node (i) consider public key (Q_j) of node (j) is revoked.</td>
</tr>
</tbody>
</table>

IV. IMPLEMENTATION AND RESULT

In implementation of existing and proposed algorithm in java I have created network scenario with different no of nodes in network to take results. For taking results I have created network with different no of nodes like (10, 20, 30, 50, 60,…). I have taken results based on total no of nodes in network, no of source nodes in network and time taken by destination node to update its key revocation list using existing and the proposed algorithms. In this paper I have included computational time comparison for existing and proposed scheme for some of scenarios.

Scenario 1: Execution time comparison to update key revocation list for network consist of 40 nodes with different no of source nodes. Here execution time is computed in nanosecond.

<table>
<thead>
<tr>
<th>Total nodes in network=40</th>
<th>Existing Scheme Time</th>
<th>Proposed scheme Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of source nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.5841212E8</td>
<td>3.6351138E7</td>
</tr>
<tr>
<td>3</td>
<td>1.70330158E8</td>
<td>1.9212587E7</td>
</tr>
<tr>
<td>4</td>
<td>1.7167386E8</td>
<td>3.7589789E7</td>
</tr>
<tr>
<td>7</td>
<td>1.42885911E8</td>
<td>4.3329544E7</td>
</tr>
<tr>
<td>10</td>
<td>1.52776142E8</td>
<td>4.8433161E7</td>
</tr>
</tbody>
</table>

Table1: Comparison of execution time for scenario 1.

Figure1 : Comparison chart of execution time for scenario1.

Scenario 2: Execution time comparison to update key revocation list for network consist of 60 nodes with different no of source nodes. Here execution time is computed in nanosecond.

<table>
<thead>
<tr>
<th>Total nodes in network=60</th>
<th>Existing Scheme Time</th>
<th>Proposed scheme Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of source nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7.23615385E8</td>
<td>1.02500269E8</td>
</tr>
<tr>
<td>3</td>
<td>6.98923727E8</td>
<td>5.8863534E7</td>
</tr>
<tr>
<td>7</td>
<td>6.99411817E8</td>
<td>5.7947511E7</td>
</tr>
<tr>
<td>10</td>
<td>7.103432E8</td>
<td>8.0187625E7</td>
</tr>
<tr>
<td>11</td>
<td>6.8439237E8</td>
<td>7.7781731E7</td>
</tr>
</tbody>
</table>

Table2: Comparison of execution time for scenario 2.

Figure2: Comparison chart of execution time for scenario 2.
V. CONCLUSION

We have already studied some existing Id based key management schemes. These key management schemes have a simple management process and reduce memory storage cost compared to other traditional methods. Some of the Id based key management schemes doesn’t provides the key revocation and renewal techniques. The computational complexity of Id based key management scheme depends on the implementation of key revocation and renewal algorithm. Existing key revocation scheme uses entire key revocation list in accusation messages. The proposed scheme will take only part of key revocation list in accusation message. From the result shown above we can conclude that the computation time to update key revocation list has been reduced by implementing the proposed scheme.

REFERENCES


