Trust- and Clustering-Based Authentication Service in Mobile Ad Hoc Networks

Presented by Edith Ngai
Supervised by Prof. Michael R. Lyu

Mphil thesis defense
18 June 2004
Outline

- Introduction
- Related Work
- Architecture and Models
- Trust- and Clustering-Based Authentication Service
- Simulation Results
- Conclusion
Mobile Ad Hoc Network

An ad-hoc network (of wireless nodes) is a temporarily formed network, created, operated and managed by the nodes themselves.

It is also often termed an infrastructure-less, self-organized, or spontaneous network.
Characteristics

- Connected with wireless communication
- Dynamic Topology
- Nodes are often mobile
- Vulnerable to security attacks

Applications

- Military: for tactical communications
- Rescue missions: in times of natural disaster
- Commercial use: for sales presentations or meetings
Vulnerabilities

- Unlike conventional networks, nodes of ad hoc networks cannot be secured in locked cabinets.
- Risk in being captured and compromised.
- Wireless communications are vulnerable to eavesdropping and active interference.
- Adversary who hijacks an ad hoc node could paralyze the entire network by disseminating false routing information.
Fundamental security mechanisms rely on the use of appropriate cryptographic keys.

Confidentiality, authentication, integrity, non-repudiation, access control and availability are considered as the main services of a security system.

Authentication service establishes the valid identities of communicating nodes.

The compromise of the authentication service breaks down the whole security system.

We focus on public key authentication service in our work.
Trust and Security

- Trust in wired networks based on trusted certification agencies and authentication servers
- Trust in mobile ad hoc networks is still an open and challenging field
- Ad-hoc networks are based on naive “trust-your neighbour” relationships
- Non-presence of a central trust authority
Related Work

- Partially-distributed certificate authority makes use of a \((k,n)\) threshold scheme to distribute the services of the certificate authority to a set of specialized server nodes.

- Fully-distributed certificate authority extends the idea of the partially-distributed approach by distributing the certificate services to every node.
Related Work

Pretty Good Privacy (PGP) is proposed by following a web-of-trust authentication model. PGP uses digital signatures as its form of introduction. When any user signs for another user's key, he or she becomes an introducer of that key. As this process goes on, a web of trust is established.

Self-issued certificates distribute certificates by users themselves without the involvement of any certificate authority.
Our Work

- Propose a secure public key authentication service in mobile ad hoc networks with malicious nodes
- Prevent nodes from obtaining false public keys of the others
- Engage a network model and a trust model
- Design security operations including public key certification, identification of malicious nodes, and trust value update
Architecture

- Public Key Certification
- Trust Value Update
- Security Operations
- Intra-group Trust Relationship
- Inter-group Trust Relationship
- Trust Model
- Cluster
- Cluster
- Cluster
- Cluster
- Network Model
- Mobile Hosts
- Nodes
The Network Model

- Obtain a hierarchical organization of a network
- Minimize the amount of storage for communication information
- Optimize the use of network bandwidth
- Limit direct monitoring capability to neighboring nodes
- Allow monitoring work to proceed more naturally
- Improve network security
The Network Model

- Unique cluster ID
- Balance cluster sizes
The Trust Model

- Define a fully-distributed trust management algorithm that is based on the web-of-trust model, in which any user can act as a certifying authority.
- This model uses digital signatures as its form of introduction. Any node signs another's public key with its own private key to establish a web of trust.
- Our trust model does not have any trust root certificate. It just relies on direct trust and groups of introducers in certification.
The Trust Model

- Define the authentication metric as a continuous value between 0.0 and 1.0.
- Define a direct trust relationship as the trust relationship between two nodes in the same group and a recommendation trust as the trust relationship between nodes of different groups.
- The first formula calculates the trust value of a new recommendation path:
  \[ V_{1} \Theta V_{2} = 1 - (1 - V_{2})^{V_{1}} \]
- The second formula draws a consistent conclusion when there are several derived trust relationships between two entities:
  \[ V_{com} = 1 - \prod_{i=1}^{m} \sqrt[n_{i}]{\prod_{j=1}^{n_{i}} (1 - V_{i, j})} \]
Clustering Structure Formation

1. Obtain its trust value from neighboring nodes
2. Run the FloodMax algorithm for d rounds
3. Run the FloodMin algorithm for d rounds
4. Select clusterhead
Clustering Structure Maintenance

- Maintain a balance clustering structure for supporting our trust model and security operations
- Adapt to the mobility of nodes
- Handle leave and join from one cluster to another
- Each node requests for the cluster ID of its neighboring nodes periodically
- In each cycle, a node collects this information and updates its cluster ID in different approaches
Approach 1

A node updates its cluster ID by joining the neighboring cluster with minimum size in each cycle.

- 40 nodes in the network
- Keeps balance cluster sizes

Algorithm 5 Clustering Structure Maintaining - Approach 1

1: for each cycle do
2:   for each node n do
3:     $v_n \rightarrow v_{neighbors} < v_n, REQ_{ClusterID} >$
4:     $v_{neighbors} \rightarrow v_n < v_n, v_{j}, ClusterID >$
5:   $min_{size} = size \ of \ ClusterID$
6:   $min_{cluster} = ClusterID$
7:   for $\forall ClusterID_i$ do
8:     if $min_{size} < size \ of \ ClusterID_i$ then
9:       $min_{size} = size \ of \ ClusterID_i$
10:      $min_{cluster} = ClusterID_i$
11:   end if
12: end for
13: end for
14: joins the $min_{cluster}$; 
15: end for
Approach 2

A node joins the neighboring cluster with minimum size only if it leaves the original cluster.

- Converge to one cluster
- Due to the imbalance cluster sizes after cluster formation?

---

### Algorithm 6: Clustering Structure Maintaining - Approach 2

1. 
2. 
3. $v_n \rightarrow v_{\text{neighbours}} < v_n, \text{REQ}\_\text{ClusterID}>$
4. $v_{\text{neighbours}} \rightarrow v_n < v_n, \text{ClusterID}>$
5. If $\text{ClusterID}_j \neq \forall \text{ClusterID}$
6. $\text{min}_\text{size} = \text{size of ClusterID}_j$
7. $\text{min}_\text{cluster} = \text{ClusterID}_j$
8. For $\forall \text{ClusterID}_j$
9. If $\text{min}_\text{size} < \text{size of ClusterID}_j$
10. $\text{min}_\text{size} = \text{size of ClusterID}_j$
11. $\text{min}_\text{cluster} = \text{ClusterID}_j$
12. End if
13. End for
14. End if
15. Joins the $\text{min}_\text{cluster}$;
16. End for
17. End for

---

Cluster Size to Round in Approach 2

![Graph showing cluster size over rounds](image-url)
Approach 3

A node joins the neighboring cluster with minimum size only if it leaves the original cluster or the sizes of the neighboring clusters are not within certain range.

- Keeps balance cluster sizes

Algorithm 7 Clustering Structure Maintaining - Approach 3

```
1: for each cycle do
2:   for each node n do
3:     v_n → v_{neighbor}; <v_n, REQ_ClusterID>;
4:     v_{neighbor} → v_n; <v_n, Cluster ID_j>;
5:     if ClusterID_n ≠ ∀ClusterID_j or ∃(S ≤ size of ClusterID_j ≤ L)
6:       then:
7:           min_{size} = size of ClusterID_f
8:           min_{cluster} = ClusterID_f
9:           for ∀ClusterID_j do
10:              if min_{size} < size of ClusterID_j then
11:                 min_{size} = size of ClusterID_f
12:                 min_{cluster} = ClusterID_f
13:           end if
14:       end if
15:     Joins the min_{cluster};
16: end for
17: end for
```
Changes in Membership

No. of Changes in Membership to Round

No. of Nodes who join a new cluster

No. of Round

No. of Rounds Nodes Stay

Frequency to No. of Rounds Nodes Stay

Approach 3
Authentication Service

- Public key certification
- Identification of Malicious Nodes
- Trust value update

- Selects a number of trustable nodes as introducers
- Sends out request messages to introducers
- Collects and compares all the public key certificates received
- Selects the public key of t with majority votes
- Discovers malicious introducer?
  - No: Calculates trust value of t
  - Yes: Isolates malicious introducer
- Updates trust table
Authentication Service

1. Looks up the group ID of \( t \), \( \Phi_t \).
2. Sorts the trust values of nodes belonging to group \( \Phi_t \) in the trust table. Let \( i_1, i_2, ..., i_n \in I \), where \( i_1, i_2, ..., i_n \) denote nodes with the highest trust values in group \( \Phi_t \).
3. Sends request messages to nodes in \( I_\omega \).
4. Collects the reply messages \( m \in M \) from \( i_1, i_2, ..., i_n \), where \( m = \{ Pk_i, V_{ik}, t, ... \}^{Sk_{ik}} \). \( Pk_i \) denotes the public key of node \( i_k \), \( V_{ik}, t \) denotes the trust value from \( i_k \) to \( t \), and \( Sk_{ik} \) denotes the secret key of \( i_k \). The reply message is signed by the secret key of \( i_k \), \( Sk_{ik} \).
5. Compares the public keys received and selects \( Pk_t \) with the majority votes. Let \( I_{good} \in I_{good} \) and \( I_{bad} \in I_{bad} \), where \( I_{good} \) are the nodes that thought to be honest (agree on \( Pk_t \) with the majority) and \( I_{bad} \) are the remaining nodes that thought to be dishonest.
6. Reduces the trust values of \( I_{bad} \) to zero. Computes and updates the trust value of \( t \), \( V_t \), with the following formulae:

\[
V_s, i_k, t = V_s, i_k \Theta V_{ik}, t = 1 - (1 - V_{ik}, t) V_s, i_k \quad \text{and} \\
V_t = 1 - \prod_{i=1}^{n} (1 - V_s, i_k, t) \quad \text{where} \ i_k \ \text{denote the nodes in} \ I_{good} \ \text{and} \ n \ \text{denotes the number of nodes in} \ I_{good}.
\]
Public Key Certification

- Authentication in our network relies on the public key certificates signed by some trust-worthy nodes.
- Nodes in the same group always know each other better by means of their monitoring components and the short distances among them.
Public Key Certification

Algorithm 4 Request for public key certificates

Given $v_i$ belongs to $CLUST_A$ and $v_j$ belongs to $CLUST_B$. A node $v_i$ requests for the public key certificate of node $v_j$:

\begin{enumerate}
  \item \textbf{if} ($CLUST_A == CLUST_B$) \textbf{then}
    \begin{enumerate}
      \item $v_i$ sends request to neighbors $v_k$:
      \begin{enumerate}
        \item $v_i \rightarrow v_k \colon \langle v_i, v_j, REQ\_CERT \rangle$;
        \item $v_k \rightarrow v_i \colon \langle v_j, T_{v_k \rightarrow j}, PK_j, \ldots \rangle_{SK_{v_k}}$;
        \item $v_i$ updates $PK_j$ and $T_j$;
      \end{enumerate}
    \end{enumerate}
  \item \textbf{else}
    \begin{enumerate}
      \item $v_i$ selects trust-worthy nodes in $CLUST_B$ as introducers $i_k$:
      \begin{enumerate}
        \item $v_i \rightarrow i_k \colon \langle v_i, v_j, REQ\_CERT \rangle$;
        \item $i_k \rightarrow v_i \colon \langle v_j, T_{i_k \rightarrow j}, PK_j, \ldots \rangle_{SK_{i_k}}$;
        \item $v_i$ compares the $PK_j$ from the received certificates and update $PK_j$ in their repository;
        \item $v_i$ calculates and updates $T_j$;
      \end{enumerate}
    \end{enumerate}
\end{enumerate}

\textbf{Send request to neighbors if target node in same cluster}

\textbf{Send request to introducers if target node in different cluster}
Identification of Malicious Nodes

- Identify malicious neighboring nodes by monitoring their behaviors
- Identify introducers who provide public key certificates different from the others
- Identify target node as malicious if the trust values provided from the introducers indicate that
Trust Value Update

\[ V_{s, ik, t} = V_{s, ik} \bowtie V_{ik, t} = 1 - (1 - V_{ik, t})^V_{s, ik} \]

\[ V_t = 1 - \prod_{k=1}^{n} (1 - V_{s, ik, t}) \]
Parameters Setting

- Network simulator Glomosim
- Evaluate the effectiveness in providing secure public key authentication in the presence of malicious nodes

<table>
<thead>
<tr>
<th>Network</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Network size</td>
<td>600m x 600m</td>
</tr>
<tr>
<td>No. of nodes</td>
<td>100</td>
</tr>
<tr>
<td>No. of groups</td>
<td>5</td>
</tr>
<tr>
<td>% of trustable nodes at initialization</td>
<td>p</td>
</tr>
<tr>
<td>% of malicious nodes</td>
<td>m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mobility</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobility</td>
<td>Random-Waypoint</td>
</tr>
<tr>
<td>Pause Time</td>
<td>20s</td>
</tr>
<tr>
<td>Maximum speed</td>
<td>10m/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PublicKeyCertification</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. no. of introducers for each request</td>
<td>3</td>
</tr>
<tr>
<td>Min. no. of reply for each request</td>
<td>1</td>
</tr>
<tr>
<td>No. of query cycles</td>
<td>80</td>
</tr>
<tr>
<td>No. of requests per cycles</td>
<td>100</td>
</tr>
<tr>
<td>Simulation Time</td>
<td>100000s</td>
</tr>
</tbody>
</table>
Simulation Metrics

- Successful rate
- Fail rate
- Unreachable rate
- False-positive error rate
- False-negative error rate

<table>
<thead>
<tr>
<th>ID</th>
<th>Cases</th>
<th>Successful Rate</th>
<th>Fail Rate</th>
<th>Unreachable Rate</th>
<th>False + Rate</th>
<th>False – Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not enough Introducers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>OOO</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>OOX</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>OXX</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>XXX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>OO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>OX</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>XX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>8</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>10</td>
<td>No Reply</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>
Ratings to Malicious Nodes

![Graphs showing ratings to % of Malicious Nodes at p=0.4 and p=0.7](image)

- (a) Ratings to % of Malicious Nodes at p=0.4
- (b) Ratings to % of Malicious Nodes at p=0.7

- successful rate
- fail rate
- unreachable rate
- false-positive
- false-negative
Ratings to Trustable Nodes at Initialization

Simulations and Results
Comparison to PGP with fixed \( m \)
Comparison to PGP with fixed $p$

Simulations and Results
Parameters Setting

- This experiment includes the neighbor monitoring, clustering formation and maintenance algorithm.

<table>
<thead>
<tr>
<th>Network</th>
<th>1500m x 1500m or 3000m x 3000m</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of nodes</td>
<td>n</td>
</tr>
<tr>
<td>% of malicious nodes</td>
<td>m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mobility</th>
<th>Random-Waypoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause Time</td>
<td>20s</td>
</tr>
<tr>
<td>Max. speed</td>
<td>10m/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clustering</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D-hops</td>
<td>3</td>
</tr>
<tr>
<td>Min. cluster size</td>
<td>S</td>
</tr>
<tr>
<td>Max. cluster size</td>
<td>L</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Neighbor Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of cycles required to identify malicious neighbors</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PublicKeyCertification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. no. of introducers for each request</td>
</tr>
<tr>
<td>Min. no. of reply for each request</td>
</tr>
<tr>
<td>No. of cycles</td>
</tr>
<tr>
<td>Simulation Time per cycle</td>
</tr>
</tbody>
</table>
Neighbor Monitoring

Rates to No. of rounds with \( n = 40 \) with \( m = 0.3 \)

Rates to No. of rounds with \( n = 40 \) and \( m = 0.7 \)

Simulations and Results
Identify Suspicious Nodes in cases 2,3,4,6,7

<table>
<thead>
<tr>
<th>ID</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not enough Introducers</td>
</tr>
<tr>
<td>1</td>
<td>OOO</td>
</tr>
<tr>
<td>2</td>
<td>OOX</td>
</tr>
<tr>
<td>3</td>
<td>OXX</td>
</tr>
<tr>
<td>4</td>
<td>XXX</td>
</tr>
<tr>
<td>5</td>
<td>OO</td>
</tr>
<tr>
<td>6</td>
<td>OX</td>
</tr>
<tr>
<td>7</td>
<td>XX</td>
</tr>
<tr>
<td>8</td>
<td>O</td>
</tr>
<tr>
<td>9</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>No Reply</td>
</tr>
</tbody>
</table>

Simulations and Results
Identify Malicious Nodes in cases 2, 4, 7

<table>
<thead>
<tr>
<th>ID</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not enough Introducers</td>
</tr>
<tr>
<td>1</td>
<td>OOO</td>
</tr>
<tr>
<td>2</td>
<td>OOX</td>
</tr>
<tr>
<td>3</td>
<td>OXX</td>
</tr>
<tr>
<td>4</td>
<td>XXX</td>
</tr>
<tr>
<td>5</td>
<td>OO</td>
</tr>
<tr>
<td>6</td>
<td>OX</td>
</tr>
<tr>
<td>7</td>
<td>XX</td>
</tr>
<tr>
<td>8</td>
<td>O</td>
</tr>
<tr>
<td>9</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>No Reply</td>
</tr>
</tbody>
</table>

Simulations and Results
Future Work

- Colluding Nodes
  - Revise trust values of nodes after real experiences with the public keys
- Trust Values Combination
  - New equations for trust values update
- Overhead
  - Evaluate the costs of the proposed scheme
- Address the problem of multiple identities
Conclusions

- We developed a trust- and clustering-based public key authentication mechanism
- We defined the network model as clustering-based and with a balance structure
- We defined a trust model that allows nodes to monitor and rate each other with quantitative trust values
- The authentication protocol proposed involves new security operations on public key certification, update of trust table, discovery and isolation of malicious nodes
- We conducted security evaluation
- We compared our approach with the PGP approach to demonstrate the effectiveness of our scheme