**Problem Specification**

- **Objective:** Construct Secure, Efficient, Robust & Scalable Key Management Schemes in MANETs, for Secure Group Communications
- **Secure:** enables/protects data exchange (it reaches all intended recipients, and only they must “read” it)
- **Efficient:** overhead caused by security (Communication, Computation, Storage, Delay Costs) low as possible
- **Robust:** handles the dynamics of the network
- **Scalable:** handles successfully a large number of nodes

<table>
<thead>
<tr>
<th>Secure, efficient, scalable Group Communications for MANETs</th>
<th>Application: Secure Routing for MANETs</th>
<th>Application: Secure the WEP of IMAC for MANETs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bootstraping</td>
<td>Key Generation</td>
<td>Entity Authentication</td>
</tr>
<tr>
<td>Key Management</td>
<td>Key Distribution</td>
<td>Key Distribution</td>
</tr>
<tr>
<td>Steady State Operations</td>
<td>Leader Ejection</td>
<td></td>
</tr>
</tbody>
</table>

**Motivation**

Focus on the Design of Suitable for MANETs Techniques for Group Formation and Steady State Operations. Emphasis on Incorporation of Entity Authentication to Key Generation/Distribution

**Example 1 (on-line Trusted Entities vs. off-line Trusted Entities)**

<table>
<thead>
<tr>
<th>Wire-line</th>
<th>MANETs</th>
</tr>
</thead>
<tbody>
<tr>
<td>- On-line and known Trusted Entities because:</td>
<td>- Trusted Entities are not available at all times because:</td>
</tr>
<tr>
<td>* Stable infrastructure</td>
<td>* Not stable infrastructure</td>
</tr>
<tr>
<td>* Infinite power</td>
<td>* Dynamic changes in the network</td>
</tr>
<tr>
<td>Effects:</td>
<td>Effects</td>
</tr>
<tr>
<td>- Two unknown nodes can trust each other anytime through Trusted Entities</td>
<td>- Unknown nodes do not have access on demand to Trusted Entities</td>
</tr>
<tr>
<td>- Establishment of trust between them is a difficult task</td>
<td>-</td>
</tr>
</tbody>
</table>

**Key Generation-Distribution Approach:** Octopus Based Protocols

**Original 2d-Octopus (O):** Breaks a Large Group into 2 Smaller Subgroups, Requires Three Steps:
1. Subgroup Key Establishment by means of a Centralized Scheme
2. Group Key Establishment from Partial Subgroup Keys via Hypercube
3. Each Subgroup Leader Communicates Group Key to All its Members

- GDH2 based 2d-Octopus (MO): Maintains 2nd Step Intact, Substitutes the Centralized Scheme of the 1st & 3rd Step with GDH2. The $M_i$ Member of GDH2 becomes Subgroup Leader for MO
- TGDH based 2d-Octopus (MOT): Maintains 2nd Step Intact, Substitutes the Centralized Scheme of the 1st & 3rd Step with TGDH. The sponsor of TGDH becomes Subgroup Leader for MOT

**Advantages**

- **Flexibility:** Select Key Generation/Distribution Subgroup Protocols Freely
- **Independence of Subgroups:** Different Key Generation Protocols May Be Applied to Each Subgroup
- **Localization:** Subgroups Deployed in a Relatively Restricted Network Area
- **Adaptability:** No Restrictions on Subgroup Size subject to Network Topology
- **Efficiency:** Less Communication OH & Bandwidth Consumption per Subgroup
- **Robustness:** Faulty Subgroup Leaders Tolerated or Replaced Dynamically

**Motivation**

**Entity Authentication Approaches for MANETs**

- **Modification of Lamport Hash for Entity Authentication/Re-Authentication Original Lamport:** Simple Password Protocol for Wire-Line Network, Prevents Eavesdropping, Impersonation, Replay Attacks. No Mutual Authentication

**Our Modification:** Suitable in MANETs, Eliminates Man-In-the-Middle Attack:

**STEP 1:** Authenticated DH Key Exchange at Bootstrapping, Obtain Secret Key, use it to Exchange Initial Quantities $\langle n, x, n \rangle$.

**STEP K (\geq 1):** Apply Original Lamport

- **Modified Merkle Trees (MFT) for (Re)Authentication, Privileges Update/Reactivation**

**Assumption:** Hierarchical Structure. Group Divided to Subgroups, MMT Applied per Subgroup and a per Subgroup CA Generator

**How it works:** Member $i$ Creates Secret Share $m_i$ from 2-Party DH Exchange with Leader. Leader Hashes these Values, and Sends Back to Node $\log N$ Values

**Results**

**Step I:** Comparative Performance Evaluation of Key Generation for protocols OFT, MO, MOT & original Octopus

**Figure 1:** OFT is by far the worst. MOT has the best performance as far as Initial Communication, then MO reduces overhead to almost half in regard to OFT

**Figure 2:** MO & MOT are by far the worst. MOT handles the dynamics of the network

**Figure 3:** MOT has the best performance as far as Initial Communication, then MO

**Figure 4:** (O) is by far the worst. MOT is the winner for addition computation. MO and OFT behave similarly. Same graph for deletion when d=4

**CONCLUSIONS – FUTURE WORK**

- Continue Design & Develop Efficient, Robust & Scalable Algorithms for Key Generation-Distribution, Entity Authentication, Steady State Operations of Key Management (KM)
- Develop Analytical Mobility Models from Realistic Mobility Patterns, Incorporate them in KM, Evaluate Effect of Mobility on KM Schemes
- Implementation of KM in Linux Platforms, to be Tested Over AODV