Wandering Nodes
Key Management for Low-power MANETs

Stefaan Seys
Bart Preneel

Katholieke Universiteit Leuven/Dept. Elektrotechniek (ESAT)
COmputer Security and Industrial Cryptography (COSIC)
http://www.esat.kuleuven.ac.be/scd/

Columbus, Ohio, USA – June 10, 2005
Outline

- Setting
- Wandering Nodes Key Management Protocol
- Performance Evaluation
- Security Evaluation
- Conclusions
Setting
Mobile Ad Hoc Network

Assumption: Nodes in network do not "jump" from one location to another, but wander through the network.
Assumptions

- Multi-hop routing
- Dense network
- Mobile nodes, but movement is *continuous*
- Low power and peanut CPU (excludes public key cryptosystems)
Examples

- Soldiers moving through battle field
- Batch of sensors dropped in sewer to track waste water
- Batch of sensors released in tornado to analyze it
- Etc.
The Hardware

Example: WINS sensor (StrongARM based)
## Power Consumption

<table>
<thead>
<tr>
<th>AES block cipher (key = 128 bit)</th>
<th>block size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128 bit</td>
</tr>
<tr>
<td>Encryption (mJoules)</td>
<td>0.31</td>
</tr>
<tr>
<td>Decryption (mJoules)</td>
<td>0.36</td>
</tr>
<tr>
<td>Elliptic Curve point-multiplication</td>
<td>Galois field size (2^n)</td>
</tr>
<tr>
<td>Power consumption (mJoules)</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Radio communications (@ 100 kbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission mode</td>
</tr>
<tr>
<td>Receive mode</td>
</tr>
</tbody>
</table>

mJoules on a StrongARM SA1100, 32-bit RISC, 900 MHz RF link

*source:* A. Hodjat and I. Verbauwhede, IEEE CAS Workshop on Wireless Communications and Networking, 2002
Power consumption (2)

- 1 AES block encryption ≈ decryption = 1 eu
- ECDSA generation = ECIES decryption = 3000 eu
- ECDSA verification = ECIES encryption = 6000 eu
- Transmitting 1 bit = $32 \times 10^{-3}$ eu
- Receiving 1 bit = $21 \times 10^{-3}$ eu
Comparison of RSA, ECDSA en DSA

Public Key Timings (ms) for RSA 1024 bit equivalents

- RSA (e=3)
- ECDSA
- DSA

[ On a RIM 950, Intel P3 @ 10 MHz (from 2000) ]
Goals of our Protocol

- Enable efficient broadcast authentication (without public key cryptography)
  - Route establishment
  - Robust data delivery

- Enable any two nodes to establish a shared secret link key
  - End-to-end encryption is more efficient

- Establish link keys within neighborhood
- Establish broadcast keys with 1-hop neighborhood
Wandering Nodes Protocol
Neighbourhood

- Neighbourhood of node $W$ is collection of all nodes $W$ can reach in $h$ hops or less = $\{N_i : \text{dist}(N_i, W) \leq h\}$.
- Node keeps following information about neighbourhood:
  - IDs
  - Shared secret
  - Keying material for group key scheme used for broadcast
  - Keying material of the broadcast groups of its immediate neighbours
High Level Protocol Overview

- Assume that initially nodes share keys with all nodes in their neighbourhood.
- As nodes move, their neighbourhood changes: new keys are established with new nodes, using existing keys (with other neighbours).
- This works as long as at least one node is shared between two subsequent neighbourhoods.
Bootstrapping the system

- Before nodes are deployed, we use a **key pre-distribution scheme** in order to bootstrap the system.

- Example 1: a single master key (not very secure)

- Example 2: 75 keys out of 10,000
  => 2 nodes share a key with 50% probability
Establishing new link keys

- Employ *secret sharing* to make the scheme more secure and robust.

- W shares a key with A and B.
- C shares a key with A and B
- W contacts C through nodes A and B
- C generates new secret K and splits it in $K_1 + K_2 = K$
- C transmits share $K_1$ to W through A and share $K_2$ through B
- W computes $K = K_1 + K_2$
- When more routes are available, a general (m,n) secret sharing scheme can be applied
Shamir’s Secret Sharing

- **Threshold cryptography** provides robustness and security.

\[ S = (s_1, s_2, s_3, \ldots, s_n) \]

- Example:

Shamir secret sharing, with polynomial of degree = 3
(4 points required)
Implementation based on DSR

- W creates RREQ targeted at C
- This RREQ is forwarded by A and B, creating two routes to C
- The authenticity (confidentiality) is protected using the already established broadcast keys of W, A and B

- C collects different RREPs
- C generates secret shares and embeds one share in each RREP
- The authenticity (confidentiality) is protected using the already established link keys
Performance Evaluation
Simulation in event driven Java-simulator

- 101 nodes that travel in random “zig-zag” path with average speed of 3.5 m/s; range = 0.5m; neighbourhood radius = 2 and neighbourhood update every 100ms
Performance Evaluation

- 10 neighborhood updates every second
  - Establish new keys with new nodes in neighborhood
  - Establish new broadcast keys with new in 1-hop neighborhood (using EHBT [Rafaeli, 2001])

- For this example, the total computational overhead is 880 eu/s, for communications this is 8870 eu/s

- Comparison: a broadcast authenticated with a ECDSA signature directed at the average neighborhood 17 nodes requires $3000 + 17 \times 6000 = 105000$ eu
Security Evaluation
Evolution of compromised keys (1)

- Suppose:
  - Total nodes in network = $K$
  - Every node has $k$ neighbours
  - Path length (hops) for key updates = $l$
  - $f(t)$ fraction of compromised keys at time $t$

- Initially every node shares a key with its neighbours, resulting in $kK$ link keys.
- At time $t_c$ the attacker learns a fraction $c$ (= $ckK$) of these keys
- Attacker can now decrypt all data encrypted with these keys
Evolution of compromised keys (2)

- A new link key established using a \((m,n)\) secret sharing scheme will be compromised if at least \(m\) paths of length \(l\) contain at least one known link key.

- If fraction \(f(t)\) is known, then the probability that an attacker will learn a new key is

\[
P(t) = \sum_{i=m}^{n} \binom{n}{i} p^i (1 - p)^{n-i} \quad \text{with} \quad p = 1 - (1 - f(t))^l
\]

Probability that a path of length \(l\) is compromised.
Evolution of compromised keys (3)

- There are $Kk$ link keys, every time one is replaced by a new one, $f()$ changes as follows:

$$f(t + 1) = \frac{Kk - 1}{Kk} f(t) + \frac{1}{Kk} P(t)$$

- It follows that $f(t+1) < f(t)$ if $P(t) < f(t)$
  => only three possible states for the system

- Maximum supported $f$ for different parameters:

<table>
<thead>
<tr>
<th>$(m, n)$-scheme path length</th>
<th>(4, 5) $l = 5$</th>
<th>(4, 5) $l = 6$</th>
<th>(3, 5) $l = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>max. supported $f$</td>
<td>10%</td>
<td>8%</td>
<td>4%</td>
</tr>
</tbody>
</table>
Conclusions

- We have proposed an efficient key management scheme designed for mobile, dense, wireless ad hoc networks
- Implementation based on DSR
- The scheme has an acceptable overhead
- We have shown how the number of compromised link keys evolves through time
Thank you