Key Distribution in Sensor Networks
Data integrity, authentication

Using PK crypto in distributed networks is:
- simple
- effective
  - enables broadcast authentication
  - distribution of new keys and insertion of new nodes is straightforward
- expensive
Symmetric-key and PK crypto in sensor nets

• Use PK for all operations
  + simple key distribution
  + simple broadcast authentication
  – sensors need to be able to perform PK crypto

• PK for key establishment (DH) and SK for the rest
  + simple key distribution
  – no efficient broadcast authentication
  – sensors need to be able to perform SK and PK crypto

• Use SK for all operations
  – **key distribution becomes an issue**
  – **no efficient broadcast authentication**
  + sensors need to be able to perform only SK crypto
(S)Key distribution in sensor networks [Eschenauer, Gligor]

1 key for all network nodes
+ low storage (1key)
+ efficient broadcast authentication
- no resilience to compromise
- easy to add new nodes
(S)Key distribution in sensor networks [Eschenauer, Gligor]

Each node pair has a different key
- high storage (n keys)
- inefficient broadcast authentication
+ resilience to node compromise
- expensive to add new nodes
Some node pairs end-up with the same keys
- lower storage ($\sqrt{n}$ keys)
- inefficient broadcast authentication
+ some resilience to node compromise
+ easy to add new nodes
Main idea:
– instead of preloading $n$ keys in each node, preload just a small subset of values ($k<<n$) that make sure that most nodes (probabilistic) or all nodes (deterministic) establish keys.

Placed between two extremes:
– single master key ($1$)
– distinct pair-wise keys for all node pairs ($n^2$)

Main issues
– Computation (per key established)
– Communication (per key established)
– Memory (sensor storage)
– Key sharing graph connectivity
– Resiliency (how many sensors need to be compromised before the entire pool is disclosed)
– Scalability
Basic probabilistic key pre-distribution

- Eschenauer and Gligor (EG), CCS 2002

$k$ keys in the pool; $\sqrt{k}$ stored per node
• **Key setup prior to deployment:**
  keys are generated and loaded into memory (the whole pool is known only to the authority)

• **Shared-key discovery after deployment:**
  each sensor node broadcasts a key identifier list to *one-hop neighborhood* (more than one pair may share the same key)

• **Path-key establishment:**
  if two sensor nodes still do not share a key
Figure 2: Probability of sharing at least one key when two nodes choose $k$ keys from a pool of size $P$. 
Key graph $G_k(V,E)$ is defined as follows:
- $V$ represents all the nodes in the sensor net
- For any two nodes $i$ and $j$ in $V$, there exists an edge between them if and only if:
  - 1) $i$ and $j$ share at least one common key

Key sharing graph $G_{sk}(V,E')$
- $i$ and $j$ have an edge if and only if
  - 1) And 2) They are within wireless transmission range
Key graph $G_k(V,E)$ is defined as follows:
- $V$ represents all the nodes in the sensor net
- For any two nodes $i$ and $j$ in $V$, there exists an edge between them if and only if:
  - 1) $i$ and $j$ share at least one common key

Key sharing graph $G_{sk}(V,E')$
- $i$ and $j$ have an edge if and only if
  - 1) And 2) They are within wireless transmission range

Better connected Key sharing graph = increased communication ability/security
Better connected key graph = increased vulnerability to compromise ...
Connectivity vs. Resiliency

- The contradictory requirement on Key Pool size $|P|$
  - Larger key pool size – better resiliency
  - Smaller key pool size – better connectivity
- The key pool size is restricted by network size
  - $|P| < \frac{k^2}{\ln(1/(1-p))}$
    - $p$ is the probability that two nodes share a key ($k$ – number of stored keys)
  - $p > \Omega(\ln N)/n$
    - $N$ is the number of sensor nodes in the network and $n$ is the average node degree.
- As $N$ increases, in order to maintain connectivity, $p$ would increase, which leads to shrink in $|P|$
- Property of resiliency does not scale with network size
  - $p$ should be non decreasing as network enlarges.
  - compromising $k$ nodes compromises $kp$ links
Deterministic Approaches

- Used to design the key pool and the key chains to provide better connectivity
  - Matrix Based Scheme [Blom 1985]
  - Polynomial Based Key Generation [Blundo et al. 1992]
Deterministic approaches: Blom’s Scheme [B]

- Public matrix $G$
- Private matrix $D$ (symmetric).

Let $A = (D \ G)^T$

$$A \ G = (D \ G)^T \ G = G^T \ D^T \ G = G^T \ D \ G = (A \ G)^T$$
[B] Scheme

\[ A = (D G)^T \]

Node i carries:

Node j carries:
[B] $\lambda$-secure Property

Undesirable Situation:
if
$u \cdot G(i) + v \cdot G(j) = G(k)$
then
$u \cdot A(i) + v \cdot A(j) = A(k)$

this would allow colluding nodes (i and j) to impersonate other nodes (k)
[B] \( \lambda \)-secure Property

- **ALL** \( \lambda+1 \) columns in G are linear independent.
  - Different from saying that G has rank \( \lambda+1 \)
  - **Rank:** there are \( \lambda+1 \) linearly independent columns
- Can tolerate compromise up to \( \lambda \) nodes.
  - Once \( \lambda+1 \) nodes are compromised, the rest can be calculated if these \( \lambda+1 \) columns are linear independent.
- How to find such a matrix G?
[B] Vandermonde Matrix

\[
G = \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & s^2 & s^3 & \cdots & s^N \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \cdots & (s^N)^\lambda
\end{pmatrix}
\]
[B] Properties of Blom Scheme

• Blom’s Scheme
  – Network size is N
  – Any pair of nodes can **directly** find a secret key
  – Tolerate compromise up to $\lambda$ nodes
  – Need to store $\lambda+2$ keys
Key distribution schemes for sensor networks


<table>
<thead>
<tr>
<th>Problem</th>
<th>Approach</th>
<th>Mechanism</th>
<th>Keying style</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair-wise</td>
<td>Probabilistic</td>
<td>Pre-distribution</td>
<td>Random key-chain</td>
<td>C, E, F, J K, N, S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pair-wise key</td>
<td>E</td>
</tr>
<tr>
<td>Deterministic</td>
<td></td>
<td>Pre-distribution</td>
<td>Pair-wise key</td>
<td>G, M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Combinatorial</td>
<td>P, Q</td>
</tr>
<tr>
<td></td>
<td>Dynamic Key</td>
<td>Master key</td>
<td>D, L</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generation</td>
<td>Key matrix</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dynamic Key</td>
<td>Polynomial</td>
<td>B, G</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hybrid</td>
<td>Pre-distribution</td>
<td>Combinatorial</td>
<td>P, Q</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dynamic Key</td>
<td>Key matrix</td>
<td>H, M, R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generation</td>
<td>Polynomial</td>
<td>I, R</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polynomial</td>
<td>B, R</td>
<td></td>
</tr>
</tbody>
</table>