Key establishment in constrained devices

Jan-Erik Ekberg

13.10.2006
The state of the art for key exchange is established. But we have classes of fringe devices where those algorithms may not be feasible:

- sensor (networks)
- mobile phone accessories
- home devices like loudspeakers, refrigerators
- automation in general (relays in lamp sockets, ..)
Constraints (1/2)

- Computational resources. Embedded controllers are weak.

- Resource cost. To add an algorithm / hardware block that only is used rarely (key establishment) will carry an overhead.

- Power cost (orthogonal): Battery consumption is for mobile devices one of the foremost constraints → heat dissipation, cost of communication ...

- Manufacturing cost (cannot be over-estimated). Cheap devices are all identical when they leave the factory,
Constraints (2/2)

- No global connectivity

- User interface. Sometimes there is none at all.

- The user (for consumer devices). The western society is moving from fix-it-yourself to return-and-complain, i.e. zero-tolerance.

Technical development will certainly carve out devices from this category and give them “proper” key establishment. But the same evolution will add devices to this class.
## Energy consumption (1/2)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Energy/op (HW)</th>
<th>Energy/op (SW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES(128b)</td>
<td>0.045 $\mu J$</td>
<td>17.9 $\mu J$</td>
</tr>
<tr>
<td>RSA(1024b)</td>
<td>2.41/0.37 $mJ$</td>
<td>546/16 $mJ$</td>
</tr>
<tr>
<td>ECC(163b)</td>
<td>0.66/1.1 $mJ$</td>
<td>134/196 $mJ$</td>
</tr>
</tbody>
</table>
Energy consumption (2/2)

![Graph showing energy consumption for different operations]

- AES128 (HW)
- ECC163 (SW)
- Transmit 128b
- Store 128b/h
Sensor networks keying

- Pre-distribution is the foremost means of key distribution

- A commonly shared key is vulnerable to sensor compromise

- Pairwise shared keys requires storage of $n$ keys
Random keys (1)

- Large pool $P$ of keys. Every device gets a key-ring with $k$ keys from $P$

- A commonly shared key is vulnerable to sensor compromise

- Pairwise shared keys requires storage of $n$ keys
Random keys (2)

In order to establish pairwise keys from the randomly distributed keys we note that the probability depends on the connectedness of the resulting networks.

The probability that two nodes share at least one key (are connected)

\[ p = 1 - \frac{k!(P-k)!(P-k)!}{P!k!(P-2k)!} \]

And the next question is whether this secure overlay is fully connected?
... so we simulate (1000 nodes, 40 nodes in neighborhood)
PIKE: Peer Intermediaries for KE

- Pairwise intermediaries in a perfect square.
- Keys always exist. Key count $= O(\sqrt{n})$, not linear.
- More resilient against node compromise.
- Deployment can be done in phases
- Can be extended to 3D and higher dimensions, and additional axes (more intermediaries) can be added.
\[
\begin{array}{cccccccc}
00 & 01 & 02 & 03 & 04 & \ldots & 09 \\
10 & 11 & 12 & 13 & 14 & \ldots & 19 \\
20 & 21 & 22 & 23 & 24 & \ldots & 29 \\
30 & 31 & 32 & 33 & 34 & \ldots & 39 \\
& \vdots & \vdots & \vdots & \vdots & \lfloor \sqrt{n} \rfloor \\
90 & 91 & 92 & 93 & 94 & \ldots & 99
\end{array}
\]

\[
A \rightarrow C : \text{MAC}_{K_{AC}}(E_{K_{AC}}(A, B, K_{AB}))
\]

\[
C \rightarrow B : \text{MAC}_{K_{BC}}(E_{K_{BC}}(A, B, K_{AB}))
\]

\[
B \rightarrow A : \text{MAC}_{K_{AB}}(E_{K_{AB}}(A, B, N_B))
\]
Saving some memory in PIKE

- Attributed to Shih-I-Huang

- We save almost 50% of the key material with this simple scheme
Tree parity and neural networks

- Key establishment by hebbian learning
- The decision function calculates parity
- Most known attacks can be made hard by enlarging the network
- Hardware solution feasible, and implemented
- Closed-form mathematical proof of equivalence of end state missing
Tree parity and neural networks

If output parities match on a common input, and $y_x = out$ then adjust input weights to $y_x$: $w'_{k,j} = w_{k,j} + out \cdot x_{k,j}$
Smart Trust for Smart Dust

Alternative attacker model

- No physical access to deployment site
- Only a small proportion of communication during deployment is monitored
- No active attacks can be executed during deployment
Key infection

$Device_1$ broadcasts its key $k_1$. Anybody that hears it, say $Device_2$, responds with $E_{k_1}(2, k_{12})$, sent with a minimum of transmission power needed (measured from the broadcast).

The latter communication is called *whispering*. 
Secrecy amplification

Simulations, with compromised links at around 3% gives around 20% improvement with three-party secrecy amplification (combined with routing setup).
Secrecy amplification in time

- SA relies on locality in space (geography)
- For mobile, pairing devices, the locality changes over time
- Amplification puts additional requirements on the eavesdropper
Secrecy amplification in time (2)

D1  D2

--------------------
RAND_A
--------------------
RAND_B
<--------------------

\[ k_1 = E(RAND_A, RAND_B) \]

... time ...

\[ E(k_1, RAND_A2), E(k_1, D1, D2, RAND_A2) \]

--------------------
\[ E(k_1, RAND_B2), E(k_1, D2, D2, RAND_B2) \]

<--------------------

\[ k_2 = E(k_1, RAND_A2 \ XOR \ RAND_B2) \]

... time ...
Amplification for moving devices

The same idea can further be extended to, say home networks, where some devices are moving, and some are not ($D_1$ fixed, $D_2$, $D_3$ mobile):

$D_2$ meets $D_3$, tells that $k_{12}$ needs amplification: $E_{k_{23}}(D_1, D_2, N_1) \rightarrow E_{k_{13}}(D_1, D_2, N_1)$. $D_2$ remembers $N_1$ and the response, and when meeting $D_1$, submits $E_{k_{13}}(D_1, D_2, N_1)$

Then, if state of $k_{13}$ is OK, then $k'_{12} = E_{k_{12}}(N_1)$ for example.
Environment monitoring

If there are randomly looking occurrences in the environment, these can be used to construct a key through amplification. E.g. monitor visible PAN addresses during a day. In the end, run the protocol:

```
D1                  D2
ADDR_1/LOW_BITS
---------------------------->
if ADDR_1 found
  k' = E(k, ADDR_1)
else k' = k

ADDR_2/LOW_BITS,
E(k', ADDR_2/LOW_BITS)
<------------------------

determine k'
if ADDR_2 found
  k'' = E(k', ADDR_2)
else k'' = k'
```
Merkle’s puzzles

\[ R_i \in [0, N] \]

\[ Device_1 \]

\[ Device_2 \]

Induces \( O(N) \) work for the attacker
key
23 229 ...

key_{\text{trans}}
1969 1407 ...

\begin{align*}
key_{\text{trans}} & \rightarrow E & key_{\text{trans}} & \rightarrow E \\
kt_1 & \rightarrow H & kt_2 & \rightarrow H \\
23 & & 229 & \\
pat_1 & & pat_2 & \\
\end{align*}

trans_1 & & \text{key}_{\text{trans}}[1] \\
(pat_1)_{0\ldots39} & 1969 \\

trans_2 & & \text{key}_{\text{trans}}[2] \\
(pat_2)_{0\ldots39} & 1407 \\

Device 1 & & Device 2 \\
\text{bacy pass} & & \text{bacy pass} \\
\text{correct path} & & \\

\text{o - new address appearing}
Piecewise obfuscation

- The devices send a message on average every two seconds,

- 16 button-presses during a 5-minute “coffee break”.

- Both devices will transmit 150 messages ($\rightarrow$ 300 messages in total)

The work for the attacker is the key space of $C = \binom{300}{16} \approx 10^{22}$ possible keys.
Conclusions

• Non-powerful devices is a reality, now and in the future

• Cryptographers must take engineering insight, cost/benefit realities as well as users into account

• Pre-distribution of keys works with some network types

• Pairing is possible with just a symmetric encryption block

• ... but sometimes a little trickery is needed