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, refridgerators
- 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 $\rightarrow$ heat dissipation, cost of communication ...
- Manufacturing cost (cannot be over-estimated). Cheap devices are all identical when they leave the factory,

Constraints (2/2)

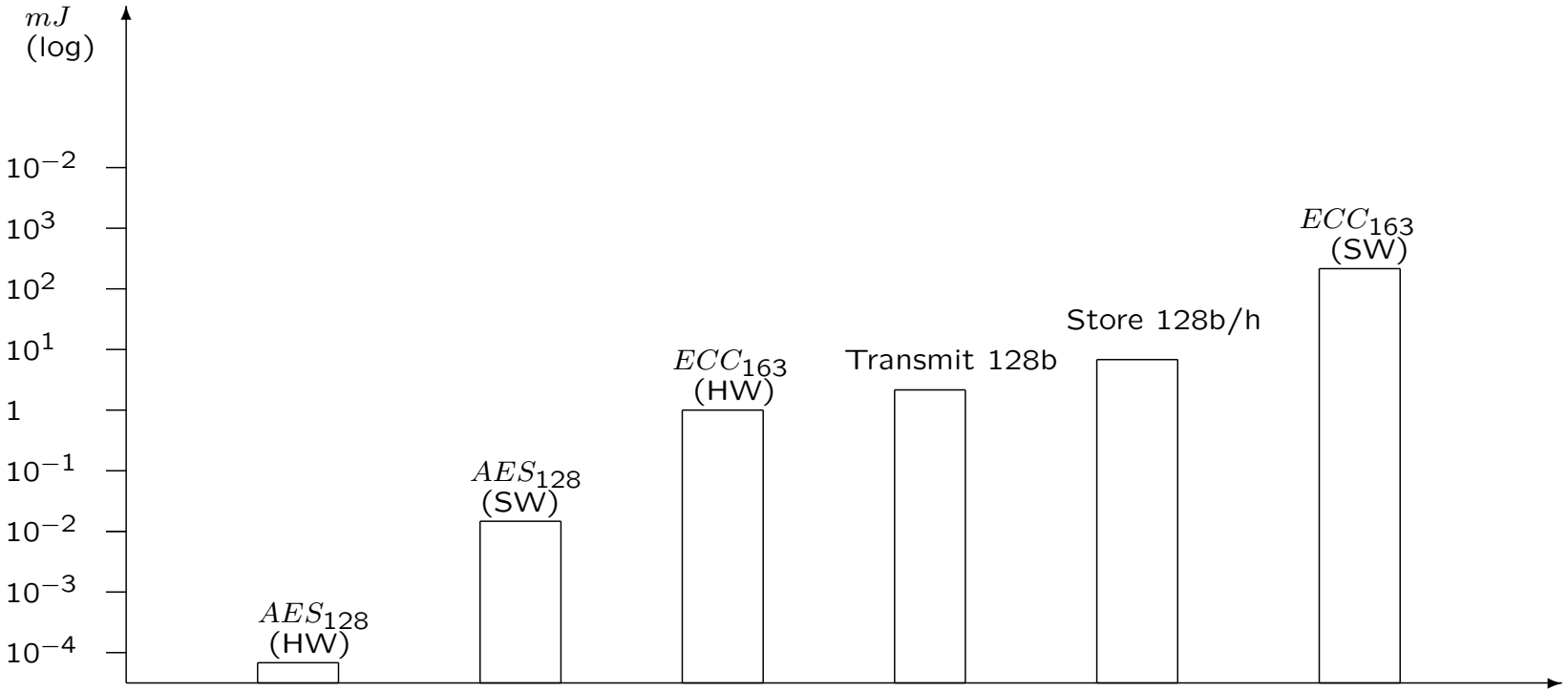
- No global connectivity
- User interface. Sometimes there is none at all.
- The user (for consumer devices). The western society is moving from fix-it-yourself to return-and-complain, i.e. zero-tolerance.

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)

Algorithm	Energy/op (HW)	Energy/op (SW)
AES(128b)	0.045 μJ	17.9 μJ
RSA(1024b)	2.41/0.37 mJ	546/16 mJ
ECC(163b)	0.66/1.1 mJ	134/196 mJ

Energy consumption (2/2)



Sensor networks keying

- Pre-distribution is the foremost means of key distribution
- A commonly shared key is vulnerable to *sensor compromize*
- 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 compromize*
- Pairwise shared keys requires storage of n keys

Random keys (2)

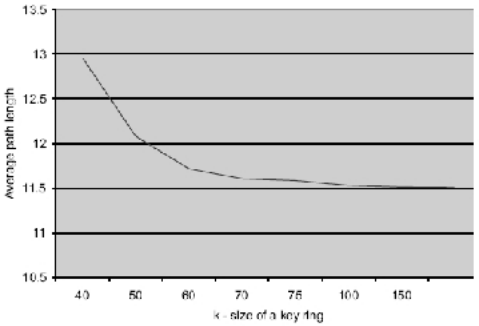
In order to establish pairwise keys from the randomly distributed keys we note that the probability depends on the connectedness of the resulting networks

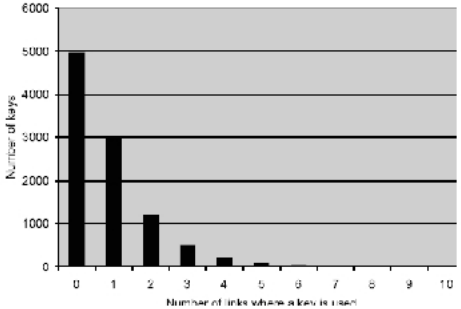
The probability that two nodes share at least one key (are connected)

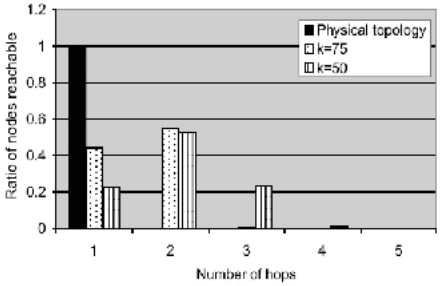
$$p = 1 - \frac{k!(P-k)!(P-k)!}{P!k!(P-2k)!}$$

And the next question is whether this secure overlay is fully connected?

... so we simulate (1000 nodes, 40 nodes in neighborhood)



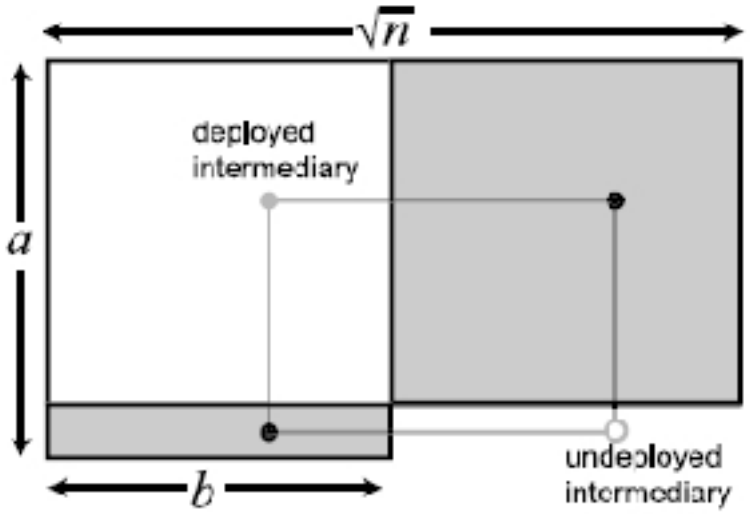




PIKE: Peer Intermediaries for KE

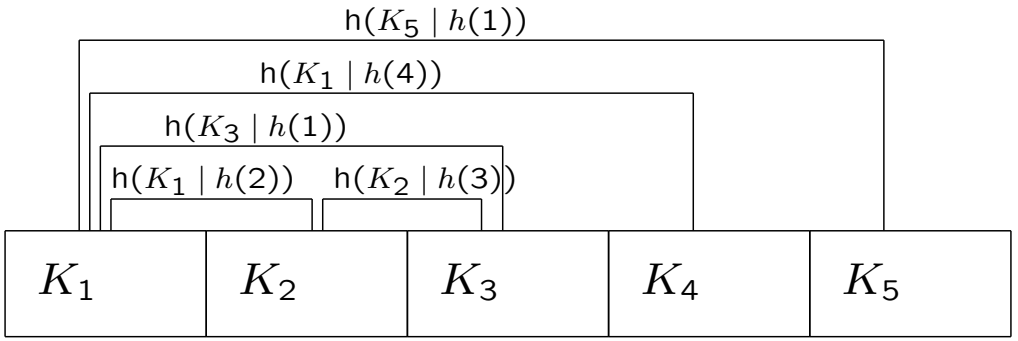
- Pairwise intermediaries in a perfect square.
- Keys always exist. Key count $= O(\sqrt{n}), notlinear$.
- More resilient against node compromize.
- Deployment can be done in phases
- Can be extended to 3D and higher dimensions, and additional axes (more intermediaries) can be added.

00	01	02	03	04	...	09
10	11	12	13	14	...	19
20	21	22	23	24	...	29
30	31	32	33	34	...	39
.	.	.	.	.		.
.	.	.	.	.		.
.	.	.	.	.		.
90	91	92	93	94	...	99



$$
\begin{aligned}
A \rightarrow C: &\ E_{K_{AC}}\{A, B, K_{AB}\}, \mathrm{MAC}_{K_{AC}}(E_{K_{AC}}\{A, B, K_{AB}\}) \\
C \rightarrow B: &\ E_{K_{BC}}\{A, B, K_{AB}\}, \mathrm{MAC}_{K_{BC}}(E_{K_{BC}}\{A, B, K_{AB}\}) \\
B \rightarrow A: &\ E_{K_{AB}}\{A, B, N_B\}, \mathrm{MAC}_{K_{AB}}(E_{K_{AB}}\{A, B, N_B\})
\end{aligned}
$$

Saving some memory in PIKE



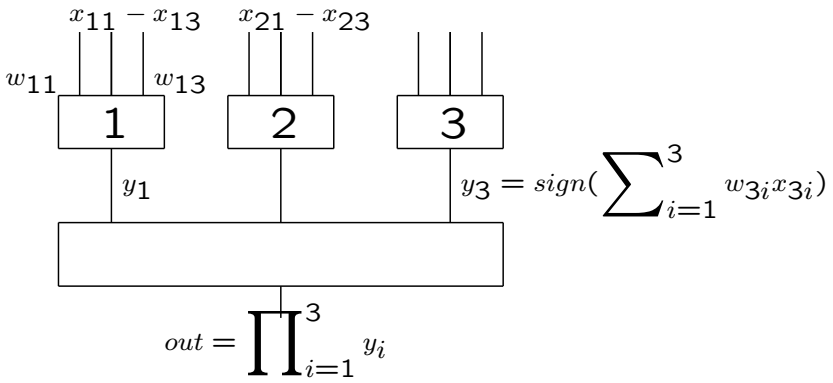
- Attributed to Shih-I-Huang

- 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: $w'_{kj} = w_{kj} + out \cdot x_{kj}$

Smart Trust for Smart Dust

Alternative attacker model

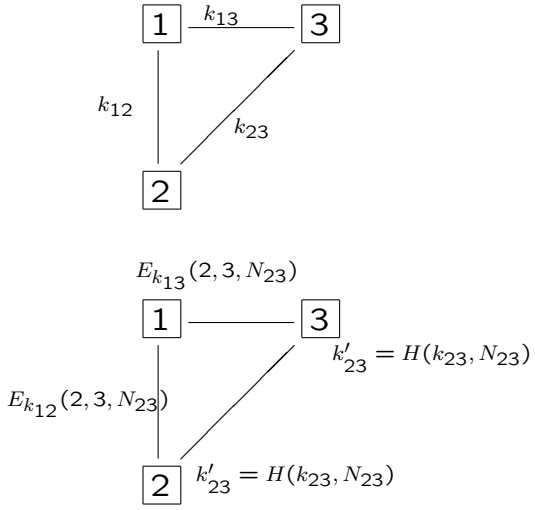
- No physical access to deployment site
- Only a small proportion of communication during deployment is monitored
- No active attacks can be executed during deployment

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
- $\rightarrow$ 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 $E_{k_{13}}(D_1, D_2, N_1)$

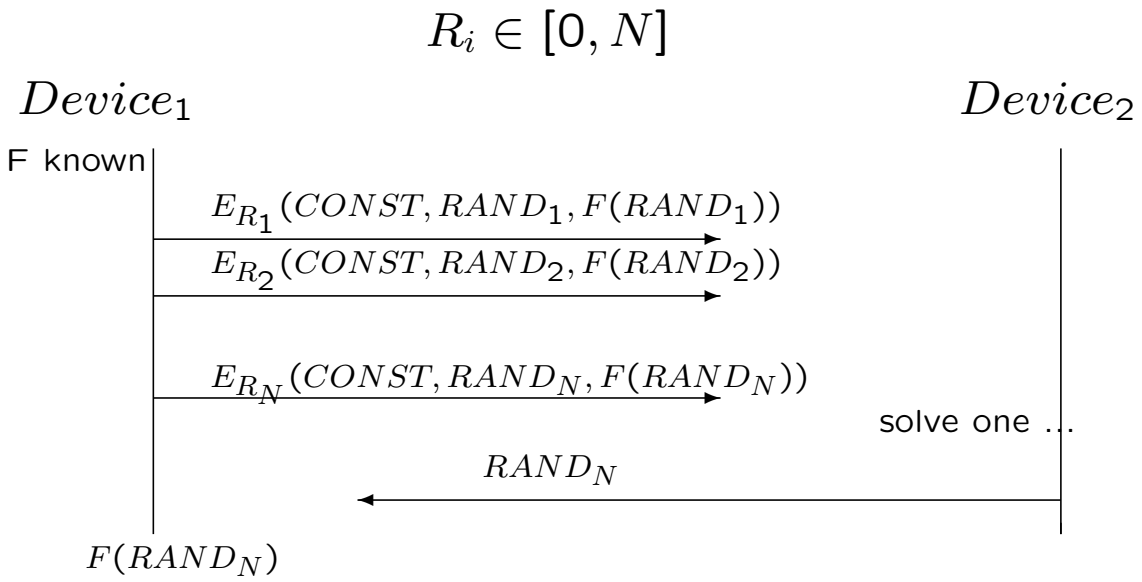
Then, if state of k_{13} is OK, then $k'_{12} = E_{k_{12}}(N_1)$ for example.

Environment monitoring

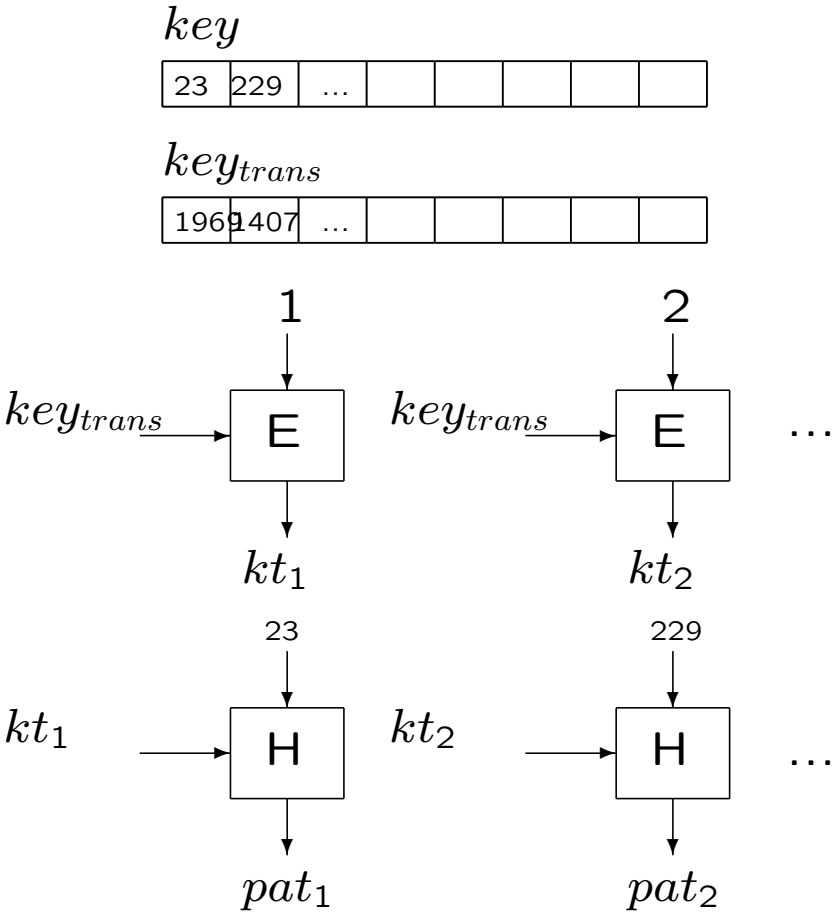
If there are randomly looking occurences in the envirommment, these can be used to construct a key through amplification. E.g. monitor visible PAN addresses during a day. In the end, run the protocol:

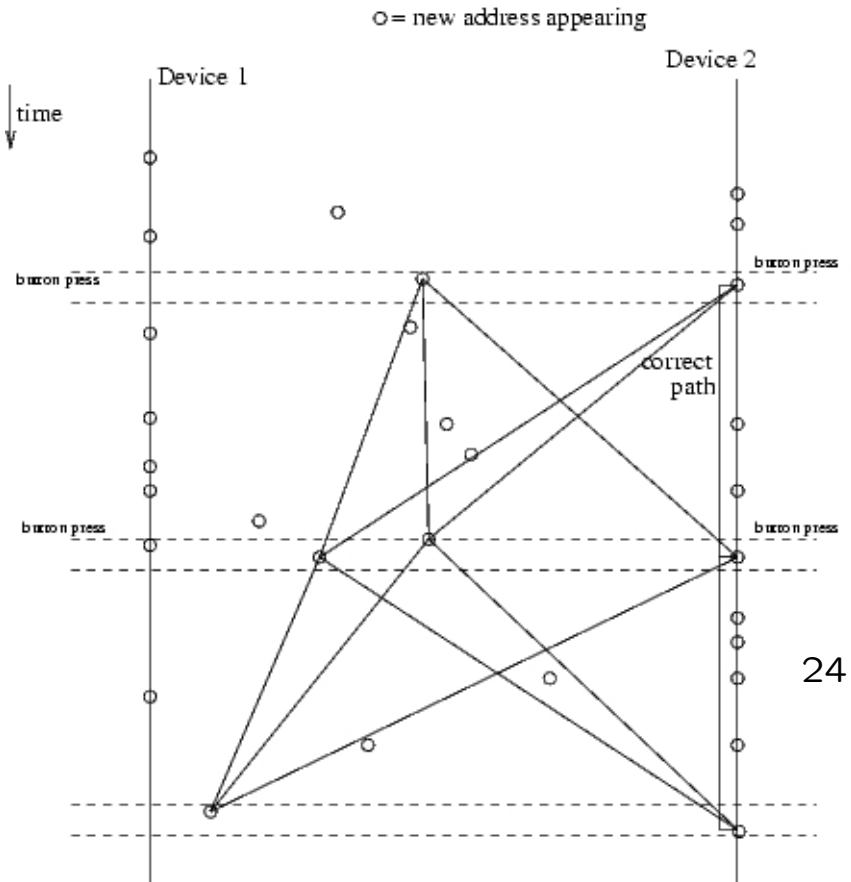
```
      D1                          D2
            ADDR_1/LOW_BITS
      ------------------------->
                                if ADDR_1 found
                                   k' = E(k, ADDR_1)
                                else k' = k
           ADDR_2/LOW_BITS,
         E(k',ADDR_2/LOW_BITS)
      <-------------------------
determine k'
if ADDR_2 found
 k'' = E(k', ADDR_2)
else k'' = k'
```

Merkle's puzzles



Induces $O(N)$ work for the attacker


key
23
229
...
key_trans
1969
1407
...
1
2
key_trans
E
key_trans
E
...
kt_1
kt_2
23
229
kt_1
H
kt_2
H
...
pat_1
pat_2
trans_1
key_trans[1]
(pat_1)_0...39
1969
trans_2
key_trans[2]
(pat_2)_0...39
1407


= new address appearing
Device 1
Device 2
time
button press
button press
button press
button press
correct
path

Piecewise obfuscation

- the devices sends 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