Practical Experiences with crypto on 8-bit

draft-ietf-lwig-crypto-sensors-01

Mohit Sethi, Jari Arkko, Ari Keränen, Heidi-Maria Back
Public Key Experiences

- Can we do Public key crypto on (really) small 8-bit devices 2-5 kB of RAM (Class 0/1)
  - What is available off-the-shelf to a developer?
  - How hard it is to run these? (How much time and hacking does it need)
  - How is the performance?
PK Experiences - RSA

- [http://www.emsign.nl/](http://www.emsign.nl/)
- AVRCryptoLib

<table>
<thead>
<tr>
<th>Key Length</th>
<th>Execution Time (ms): Keys in SRAM</th>
<th>Memory footprint (bytes): Keys in SRAM</th>
<th>Execution Time (ms): Keys in ROM</th>
<th>Memory footprint (bytes): Keys in ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>64</td>
<td>40</td>
<td>69</td>
<td>32</td>
</tr>
<tr>
<td>128</td>
<td>434</td>
<td>80</td>
<td>460</td>
<td>64</td>
</tr>
<tr>
<td>256</td>
<td>3516</td>
<td>160</td>
<td>3818</td>
<td>128</td>
</tr>
<tr>
<td>512</td>
<td>25076</td>
<td>320</td>
<td>27348</td>
<td>256</td>
</tr>
<tr>
<td>1024</td>
<td>199688</td>
<td>640</td>
<td>218367</td>
<td>512</td>
</tr>
<tr>
<td>2048</td>
<td>1587567</td>
<td>1280</td>
<td>1740258</td>
<td>1024</td>
</tr>
</tbody>
</table>
ECDSA libraries

Performance Comparison of ECC

- TinyECC
- TinyECC (No assembly)
- Wiselib
- Relic-Binary-fast
- Relic-Binary-low

Curves:
- 160k1
- 160r1
- 160r2
- 160k1
- 160r1
- 160r2
- k163-aem
- k163
- b163
- k163-aem
- k163
- b163

Execution Time (ms)

RAM Used (bytes)
EdDSA libraries

- Edwards-curve Digital Signature Algorithm (EdDSA)
- NaCl and μNaCl high-speed software library
- Public domain
- Signing* = 23,216,241 clock cycles ~ 1,4 sec
- Verification* = 32,634,713 clocks cycles ~ 2 sec

* NaCl on 8-Bit AVR Microcontrollers, Michael Hutter, Peter Schwabe, http://link.springer.com/chapter/10.1007/978-3-642-38553-7_9
Example application

Client

Server

Delegate Work

Register

Proxy

Get Data

RD

Client

draft-ietf-lwig-crypto-sensors-01, Sethi et al.
## Example application

<table>
<thead>
<tr>
<th>Memory Consumption / Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash memory consumption (for the entire prototype (including Relic crypto + CoAP + Arduino UDP etc. libraries))</td>
<td>51 kB</td>
</tr>
<tr>
<td>SRAM consumption (for the entire prototype including client + key generation + signing the hash of message + COAP + UDP)</td>
<td>4678 bytes</td>
</tr>
<tr>
<td>Execution time for creating the key pair + sending registration message + time spent waiting for acknowledgement</td>
<td>2030 ms</td>
</tr>
<tr>
<td>Time for signing the hash of message + sending update</td>
<td>987 ms</td>
</tr>
<tr>
<td>Signature overhead</td>
<td>42 bytes</td>
</tr>
</tbody>
</table>

---

draft-ietf-lwig-crypto-sensors-01, Sethi et al.
What we learnt

• Chosen prototype platform was unnecessarily restrictive in the amount of code space
  – we chose this platform on purpose to demonstrate something that is as small and difficult as possible

• Power requirements necessary to send or receive messages are far bigger than those needed to execute cryptographic operations

• No good reason to choose platforms that do not provide sufficient computing power to run the necessary operations
Discussion

• Feasibility

• Message freshness

• Symmetric vs Asymmetric

• Link vs Network vs Transport vs Application
Changes

• Working group adoption call during last IETF
• Interest in the group and confirmed on the mailing list
• Thanks for feedback: Akbhar, Rahul, Daniel, John, Abhijan, Renzo, Raghavendra
• Remove reference to DTLS group keys
• Fix editorial suggestions
• Update reference to Pub/Sub broker
• Smaller key lengths are for reference only
• Ready for WGLC
• More reviews are welcome