A New Kid on the Block: CLINT - a Cryptographic Library for the INternet of Things

Mike Scott
CertiVox Ltd

Real World Crypto, London, January 2014
A problem

Figure: Communication Problem
Part of the Reason?

Figure: Research Reality
There are Real Problems!

Virtual Real World Crypto
(Here Quantum Computers exist!)

Virtual Real World Crypto
(Here multi-linear maps exist)

Virtual Real World Crypto
(Here FHE is efficient)

Help!!!

Yet Another Virtual Real World Crypto

Figure: These guys need help!
Maybe Part of the Solution

Academic Cryptographers

Virtual Real World Crypto
(Here Quantum Computers exist!)

Virtual Real World Crypto
(Here FHE is efficient)

Virtual Real World Crypto
(Here multi-linear maps exist)

Yet Another Virtual Real World Crypto

Tools

Help!!!

Real Real World Crypto

Figure: Easy to use tools
There are many crypto libraries out there.

Many offer a bewildering variety of cryptographic primitives, at different levels of security.

Many use extensive assembly language in order to be as fast as possible.

Many are very big, even bloated. Some rely on other external libraries.

Most were designed by academics for academics, and so are not really suitable for commercial use.
CLINT – 1

- CLINT is completely self-contained (except for the requirement for an external entropy source).
- CLINT is for use in the pre-quantum era – that is in the here and now.
- CLINT is portable - no assembly language.
- The release version is available in pure C, Java and Javascript using only generic programming constructs.
- New language version can be produced in 3-4 weeks. Next up C# and Swift.
- All versions will be “identical” – all internal calculations are the same.
CLINT – 2

- CLINT is fast, but does not attempt to set speed records (a particular academic obsession).
- CLINT is small – less than 10,000 lines of code.
- CLINT has a very small footprint – important for IoT.
- CLINT supports only one level of security (AES-128)
- CLINT implements only curve based Public Key methods (including Pairings)
Support for SHA256, AES-128, AES modes plus GCM
Raw Entropy processing for random number generation.
Elliptic Curves (Weierstrass, Edwards, Montgomery)
Types of moduli (general, Montgomery friendly, pseudo-mersenne)
BN-curve based optimal pairings
2048-bit RSA (legacy support)
Awareness of modern pipelined architecture
Avoid `if` statements (particularly unpredictable branches)
Side channel attack resistance baked-in.
Example APIs that communicate to the “Real World” using simple byte arrays.
Raspberry pi implementation - space

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Code Size</th>
<th>Maximum Stack Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC protocol -O3</td>
<td>63236</td>
<td>3004</td>
</tr>
<tr>
<td>ECC protocol -Os</td>
<td>30102</td>
<td>2940</td>
</tr>
<tr>
<td>PBC protocol -O3</td>
<td>80493</td>
<td>10124</td>
</tr>
<tr>
<td>PBC protocol -Os</td>
<td>45008</td>
<td>9744</td>
</tr>
</tbody>
</table>

**Table:** Typical Memory Footprint
## Raspberry pi implementation - time

<table>
<thead>
<tr>
<th></th>
<th>Time in milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC point multiplication -O3</td>
<td>11.9</td>
</tr>
<tr>
<td>ECC point multiplication -Os</td>
<td>17.2</td>
</tr>
<tr>
<td>PBC pairing -O3</td>
<td>85</td>
</tr>
<tr>
<td>PBC pairing -Os</td>
<td>122</td>
</tr>
</tbody>
</table>

**Table: C Benchmarks**
Question Time

- Thank you for your attention