The Challenges of Securing and Authenticating Embedded Devices and a Suggested Approach for RISC-V

RISC-V Workshop

Derek Atkins
Chief Technology Officer
617-623-3745
datkins@securerf.com
What do you think when...

Think about your small embedded device.

Imagine trying to secure it with public key technology.

Can it be done?
What About Symmetric Solutions?

Why not just use Symmetric Encryption (e.g. AES)?

- It fits
- It’s fast
  - but.....
- It’s hard to deploy
- It doesn’t scale
Public Key’s Bad Rap

Why do people think Public Key won’t work?

▶ Too big
▶ Too slow
▶ Too much power/energy
The Bad News

Many of these beliefs are true.

- **Speed**: ARM reports ECC execution times in the 233-1089 ms range on the Cortex-M (0-4)
- **Size**: Implementations of ECC range in 8-30KB of ROM and require 800-3000B of RAM
- **Hardware implementations** are faster but take a lot of gates (or LUTs)
- **RSA and Diffie-Hellman** are larger and take longer

Why? Multiplying 256-4096-bit numbers takes a lot of effort!
Some Good News
Enter Group Theoretic Cryptography (GTC)

- Studied over 100 years
- GTC dates to the 1970s
- Complexity scales linearly with security (instead of quadratically like RSA, ECC, and DH)
More on GTC

GTC is small, fast, secure, and quantum resistant.

GTC doesn’t require extended math like ECC/RSA/DH (GTC uses 6-8 bit math)

GTC implementations are small, efficient, and fast

SecureRF supplies a GTC-based key agreement and digital signature method for RISC-V (and other platforms)
GTC Methods

▶ Ironwood Key Agreement Protocol

   Enables two endpoints to generate a shared secret over an open channel.

▶ Walnut Digital Signature Algorithm

   Allows one device to generate a document that is verified by another.
   Very fast verification
Test Platform

**RISC-V Core**  SiFive E3 Coreplex with single-issue in-order, 32-bit execution pipeline, RV32IM.

**Toolchain**  SiFive RISC-V GNU toolchain. Hardware multiply and divide instructions disabled.

**Debugger**  Olimex ARM-USB-TINY-H with JTAG support.

**Dev board**  Digilent Arty with Xilinx Artix-7 FPGA, no hard processor system, core clock speed: 62.5MHz.
WalnutDSA vs. ECDSA

<table>
<thead>
<tr>
<th></th>
<th>WalnutDSA</th>
<th>microECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Run time</td>
<td>4.9 ms</td>
<td>* 2,110 ms</td>
</tr>
<tr>
<td>Cycle count</td>
<td>307,300</td>
<td>131,874,900</td>
</tr>
<tr>
<td>ROM used</td>
<td>4,780</td>
<td>31,920</td>
</tr>
<tr>
<td>RAM used</td>
<td>272</td>
<td>940</td>
</tr>
</tbody>
</table>

* 458 ms with hardware multiple/divide instructions enabled
# C vs. RISC-V Assembly Language

<table>
<thead>
<tr>
<th>Method</th>
<th>C</th>
<th>RISC-V Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>WalnutDSA, 128bit</td>
<td>4.9 ms</td>
<td>3.0 ms</td>
</tr>
<tr>
<td>Cycle count</td>
<td>307,300</td>
<td>188,700</td>
</tr>
<tr>
<td>ROM used</td>
<td>4,780</td>
<td>4,580</td>
</tr>
<tr>
<td>RAM used</td>
<td>272</td>
<td>272</td>
</tr>
</tbody>
</table>
## RISC-V vs. ARM Cortex-M3

<table>
<thead>
<tr>
<th></th>
<th>RISC-V</th>
<th>ARM Cortex-M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>WalnutDSA, 128bit</td>
<td>WalnutDSA, 128bit</td>
</tr>
<tr>
<td>Implementation</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Core clock freq</td>
<td>62.5 MHz</td>
<td>48 MHz</td>
</tr>
<tr>
<td>Run time</td>
<td>4.9 ms</td>
<td>5.7 ms</td>
</tr>
<tr>
<td>Cycle count</td>
<td>307,300</td>
<td>275,563</td>
</tr>
<tr>
<td>ROM used</td>
<td>4,780</td>
<td>2,952</td>
</tr>
<tr>
<td>RAM used</td>
<td>272</td>
<td>272</td>
</tr>
</tbody>
</table>
Thank You!

Any Questions?

SecureRF Corporation
100 Beard Sawmill Rd, Suite 350
Shelton, CT 06484
(203) 227-3151

Derek Atkins (datkins@securerf.com)