Secure96
Joakim Bech
Agenda

- Mezzanine board with security ICs, why?
- ATSHA204A
- ATECC508A
- TPM - Infineon SLB 9670
- What is next?
Mezzanine board with security ICs, why?

- **SFO16 keynote demo**
  - Used IoT devices with different crypto capabilities
  - **Main concern:** how to store keys in a secure manner

- **Found out about**
  - Atmel CryptoAuthentication™ AT88CK590 Demo-evaluation Kit
  - ATECC508A (asymmetric) got my attention, but later on also ATSHA204A
  - Turnkey (?) solution for secure boot MCU’s?

- **Common ground**
  - While working with Zephyr for example, it would be nice to have common ground regardless what MCU you are using while fleshing out initial APIs etc

- **Maker community**
Secure96 - Top

FTDI
TPM
ATSHA204A
ATECC508A
EEPROM
Secure96 - Bottom

- 40-pin (2x20) connector
- Compatible with 96Boards
Similar hardware

- **Atmel CryptoAuthentication™ AT88CK590 Demo-evaluation Kit**

- **CryptoShield** by Josh Datko
  - No longer for sale
  - Nice software called “Hashlet”
ATSHA204A

- **Symmetric authentication** (but still capable of deriving diversified keys)
- I/O: I²C / SWI
- Commands like:
  - CheckMac, DeriveKey, GenDig, HMAC, SHA, Nonce, MAC, Random, Read, Write etc ...
- Not just plain crypto commands
  - HMAC, SHA-256 digest over the data + some other device data (serial number, OTP etc)
- Random Number Generator
- Guaranteed Unique 72-bit Serial Number
- Been on the market for quite some time (launched 2011)
ATSHA204A - Zones - Configuration

- 88 bytes
- Array with information such as
  - Serial number
  - I2C address
  - Various modes
  - Slot configuration
  - Lock configuration etc
- *Read* and *Write* commands

<table>
<thead>
<tr>
<th>Word</th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Default</th>
<th>Write Access</th>
<th>Read Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 23 xx xx</td>
<td>Never</td>
<td>Always</td>
</tr>
<tr>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xx xx xx xx</td>
<td>Never</td>
<td>Always</td>
</tr>
<tr>
<td>0x02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xx xx xx xx</td>
<td>Never</td>
<td>Always</td>
</tr>
<tr>
<td>0x03</td>
<td>SN[8]</td>
<td>Reserved</td>
<td>I2C Enable</td>
<td>Reserved</td>
<td>EE 55 xx 00</td>
<td>Never</td>
<td>Always</td>
</tr>
<tr>
<td>0x04</td>
<td>I2C Address</td>
<td>CheckMacConfig</td>
<td>OTP Mode</td>
<td>Selector Mode</td>
<td>C8 00 55 00</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x05</td>
<td>Slot Configuration 0</td>
<td>Slot Configuration 1</td>
<td>8F 80 80 A1</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x06</td>
<td>Slot Configuration 2</td>
<td>Slot Configuration 3</td>
<td>82 E0 A3 60</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x07</td>
<td>Slot Configuration 4</td>
<td>Slot Configuration 5</td>
<td>94 40 A0 65</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>Slot Configuration 6</td>
<td>Slot Configuration 7</td>
<td>66 40 87 C7</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x09</td>
<td>Slot Configuration 8</td>
<td>Slot Configuration 9</td>
<td>0F 00 89 F2</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0A</td>
<td>Slot Configuration 10</td>
<td>Slot Configuration 11</td>
<td>8A 7A 00 88</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B</td>
<td>Slot Configuration 12</td>
<td>Slot Configuration 13</td>
<td>0C 4C 8D 4D</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>Slot Configuration 14</td>
<td>Slot Configuration 15</td>
<td>C2 42 AF 6F</td>
<td>If Config is unlocked</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0D</td>
<td>Use Flag 0</td>
<td>Update Count 0</td>
<td>Use Flag 1</td>
<td>Update Count 1</td>
<td>FF 00 FF 00</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x0E</td>
<td>Use Flag 2</td>
<td>Update Count 2</td>
<td>Use Flag 3</td>
<td>Update Count 3</td>
<td>FF 00 FF 00</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x0F</td>
<td>Use Flag 4</td>
<td>Update Count 4</td>
<td>Use Flag 5</td>
<td>Update Count 5</td>
<td>FF 00 FF 00</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x10</td>
<td>Use Flag 6</td>
<td>Update Count 6</td>
<td>Use Flag 7</td>
<td>Update Count 7</td>
<td>FF 00 FF 00</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x11</td>
<td>Last Key Use 0</td>
<td>Last Key Use 1</td>
<td>Last Key Use 2</td>
<td>Last Key Use 3</td>
<td>FF FF FF FF</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x12</td>
<td>Last Key Use 4</td>
<td>Last Key Use 5</td>
<td>Last Key Use 6</td>
<td>Last Key Use 7</td>
<td>FF FF FF FF</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x13</td>
<td>Last Key Use 8</td>
<td>Last Key Use 9</td>
<td>Last Key Use 10</td>
<td>Last Key Use 11</td>
<td>FF FF FF FF</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x14</td>
<td>Last Key Use 12</td>
<td>Last Key Use 13</td>
<td>Last Key Use 14</td>
<td>Last Key Use 15</td>
<td>FF FF FF FF</td>
<td>If Config is unlocked</td>
<td>Always</td>
</tr>
<tr>
<td>0x15</td>
<td>User Extra</td>
<td>Selector</td>
<td>Lock Data</td>
<td>Lock Config</td>
<td>00 00 55 55</td>
<td>Via Update Extra Command Only</td>
<td>Always</td>
</tr>
</tbody>
</table>
ATSHA204A - Zones - Data

- 512 bytes - split into 16 general purpose registers (slots)
- One slot = 32 bytes
- There can be different restrictions on each slot (slot configuration)
- Typically used to store
  - Keys
  - Model number
  - Calibration data etc
ATSHA204A - Zones - OTP

- 64 bytes
- Two modes (three)
  - Read-only
    - Impossible to do any further updates.
  - Consumption
    - 0xFF default state, bits can only be changed to zero (after locking config).
Personalize the device

1. Configure and lock the configuration data
   ○ Lots of options here. When just playing with the device, the default configuration is probably a pretty good start

2. Program the slots

3. Lock the data and OTP

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 3</td>
<td>ReadKey</td>
<td>Slot of the key to be used for encrypted reads. If 0x0, then this slot can be used as the source slot for the CheckMac Copy Command.</td>
</tr>
<tr>
<td>4</td>
<td>CheckOnly</td>
<td>0 = This slot can be used for all crypto commands. 1 = This slot can only be used for CheckMac and GenDig followed by CheckMac Commands.</td>
</tr>
<tr>
<td>5</td>
<td>SingleUse</td>
<td>0 = No limit on the number of time the key can be used. 1 = Limit on the number of time the key can be used based on the UseFlag (or LastKeyUse) for the slot.</td>
</tr>
<tr>
<td>6</td>
<td>EncryptRead</td>
<td>0 = Clear reads are permitted. 1 = Requires the slot to be Secret and encrypted read to access.</td>
</tr>
<tr>
<td>7</td>
<td>IsSecret</td>
<td>0 = The slot is not secret and allows clear read, clear write, no MAC check, and no Derivekey Command. 1 = The slot is secret. Reads and writes if allowed, must be encrypted.</td>
</tr>
<tr>
<td>8 – 11</td>
<td>WriteKey</td>
<td>Slot of the key to be used to validate encrypted writes.</td>
</tr>
<tr>
<td>12 – 15</td>
<td>WriteConfig</td>
<td>See detailed function definition for use.</td>
</tr>
</tbody>
</table>
### Personalize - Write Config bits

<table>
<thead>
<tr>
<th>Bit15</th>
<th>Bit14</th>
<th>Bit13</th>
<th>Bit12</th>
<th>Write Command</th>
<th>Description</th>
<th>DeriveKey</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ALWAYS</td>
<td>Clear text writes ALWAYS permitted</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ALWAYS</td>
<td>Clear text writes ALWAYS permitted</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>NEVER</td>
<td>Writes NEVER permitted</td>
<td>TARGET</td>
<td>DeriveKey command can be run without authorizing MAC (Roll).</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>NEVER</td>
<td>Writes NEVER permitted</td>
<td>PARENT</td>
<td>DeriveKey command can be run without authorizing MAC (Create).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>TARGET</td>
<td>DeriveKey command can be run without authorizing MAC (Roll).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>PARENT</td>
<td>DeriveKey command can be run without authorizing MAC (Create).</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NEVER</td>
<td>Writes NEVER permitted</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>NEVER</td>
<td>Writes NEVER permitted</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>NEVER</td>
<td>Writes NEVER permitted</td>
<td>TARGET</td>
<td>Authorizing MAC required for DeriveKey command (Roll).</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>NEVER</td>
<td>Writes NEVER permitted</td>
<td>PARENT</td>
<td>Authorizing MAC required for DeriveKey command (Create).</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>TARGET</td>
<td>Authorizing MAC required for DeriveKey command (Roll).</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ENCRYPT</td>
<td>Writes permitted if MAC is OK</td>
<td>PARENT</td>
<td>Authorizing MAC required for DeriveKey command (Create).</td>
</tr>
</tbody>
</table>
Use cases

- **Accessory authentication**
  - Mobile device wants to authenticate a (genuine) battery
  - Consumables, ink cartridges etc
  - Technical support

- Secure boot
- Data integrity verification
- Session key exchange

Yes, I know it smells DRM ...
I²C bus and still secure?

- How do I protect the bus between the crypto device and microprocessor?
  - Atmel’s answer (from the FAQ):
    - “Perform an authentication on the result of crypto calculation (CheckMAC command)”
    - “It involves using the second key that is both stored in the CryptoAuthentication device and compiled into the code. After the successful completion of the “CheckMAC” operation, the second secret is copied into the TempKey register. Then the MCU sends over a unique number (for example, time of day), which is then combined with that second secret using SHA and returned to the MCU.”
    - “The software on the MCU does the same combination using the compiled secret to see if it agrees with the result from the authentication device.”
  - The “compiled into the code” part sounds scary and feels like it defeats the purpose of the device, but what if you have that part in a TrustZone protected environment for example?
Software for ATSHA204

- **Secure96** → [https://github.com/jbech-linaro/secure96](https://github.com/jbech-linaro/secure96)
  - In the future this should be moved to a “Linaro” page.
  - Still very much Work In Progress, but basic communication, read, write, getting config data, generate random numbers, nonce, HMAC etc has been implemented.
  - Personalization “works”, but still a bit crude.
  - Works with any I²C enabled device, but currently configured so it works with DragonBoard 410c by default.

- **Hashlet** → [https://github.com/cryptotronix/hashlet](https://github.com/cryptotronix/hashlet)
  - Has been around for a while, seems pretty mature
  - GPLv3

  - Atmel’s own reference implementation
ATECC508A

- Shares lots of functionality with ATSHA204A
- Major difference is that it is working with **asymmetric key pairs** instead of symmetric key pairs
- Supports ECDH and ECDSA
- Requires NDA (why?) to get the datasheet / TRM!
  - The CryptoAuthLib supports this IC, so you can study the code without the reference manual, but ….
- No work done so far, but in the end this is the device that we would like to use
TPM - Infineon SLB 9670

- I/O: SPI
- Compliant to TCG TPM 1.2 and 2.0
- Have not done any work with this device more than sanity test it using the
  - Intel TSS TPM2.0 resource manager
    https://github.com/01org/TPM2.0-TSS.git
  - And the tpm2.0 tools
    https://github.com/01org/tpm2.0-tools.git
- For more details see the official page for the IC
What is next?

- No real roadmap for now, but ... some ideas
  - Finalize the ATSHA204A implementation
  - Create a library for the ATSHA204A implementation
  - Offline implementation to mimic device behavior (in a Trusted Application in a TEE)
  - Use IC(s) for secure boot on a 96Boards IoT device
  - Get the specification and implement support for ATECC508A
  - TPM chip
    - Try it out using IMA in Linux
    - Use it to store SSH credentials
Thank You

#SFO17

SFO17 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org