Linaro connect
San Francisco 2017

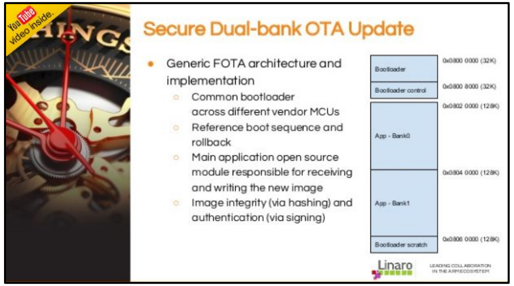
Secure96

Joakim Bech

Agenda

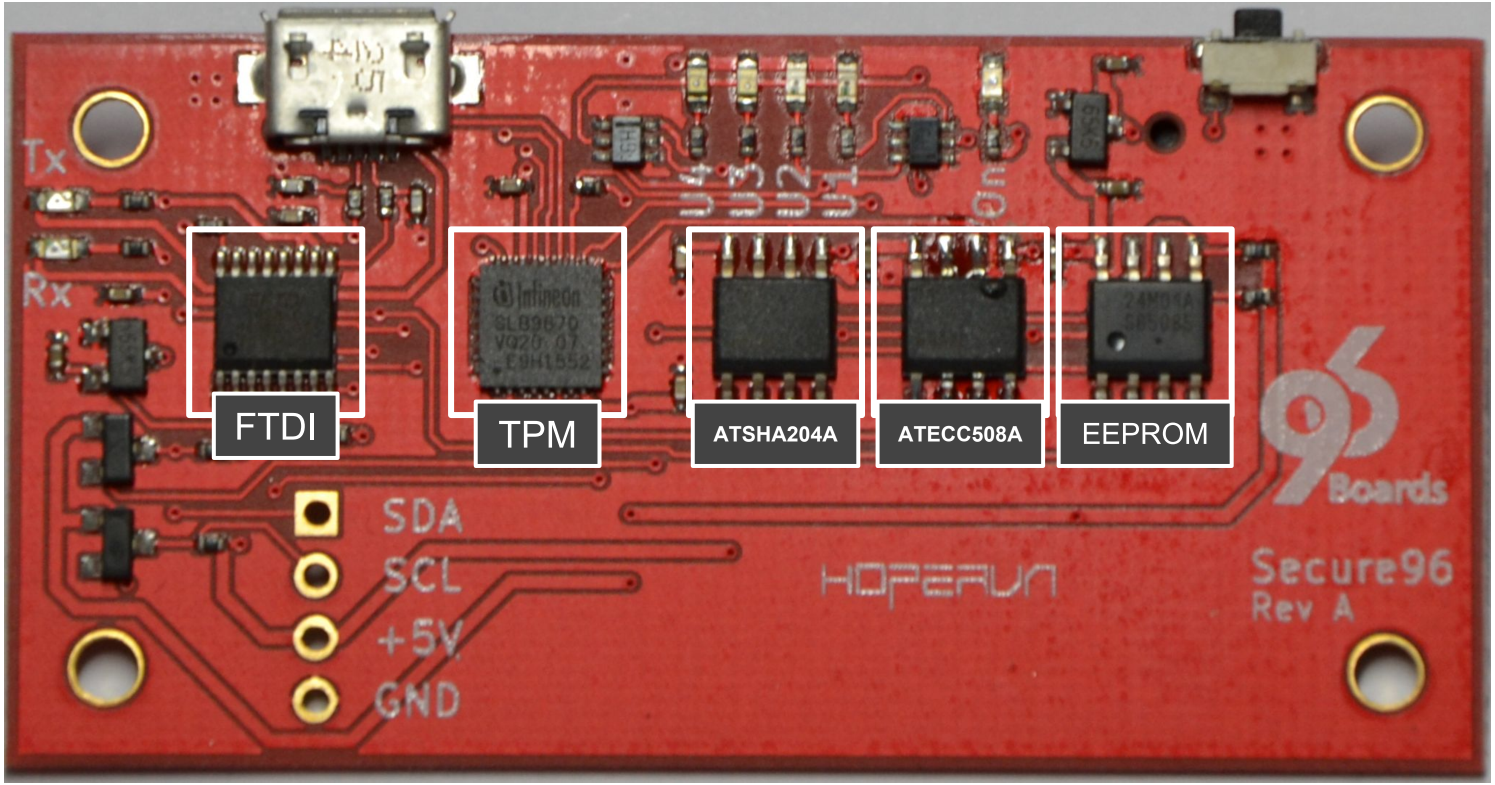
- Mezzanine board with security ICs, why?
- ATSHA204A
- ATECC508A
- TPM - Infineon SLB 9670
- What is next?

Mezzanine board with security ICs, why?

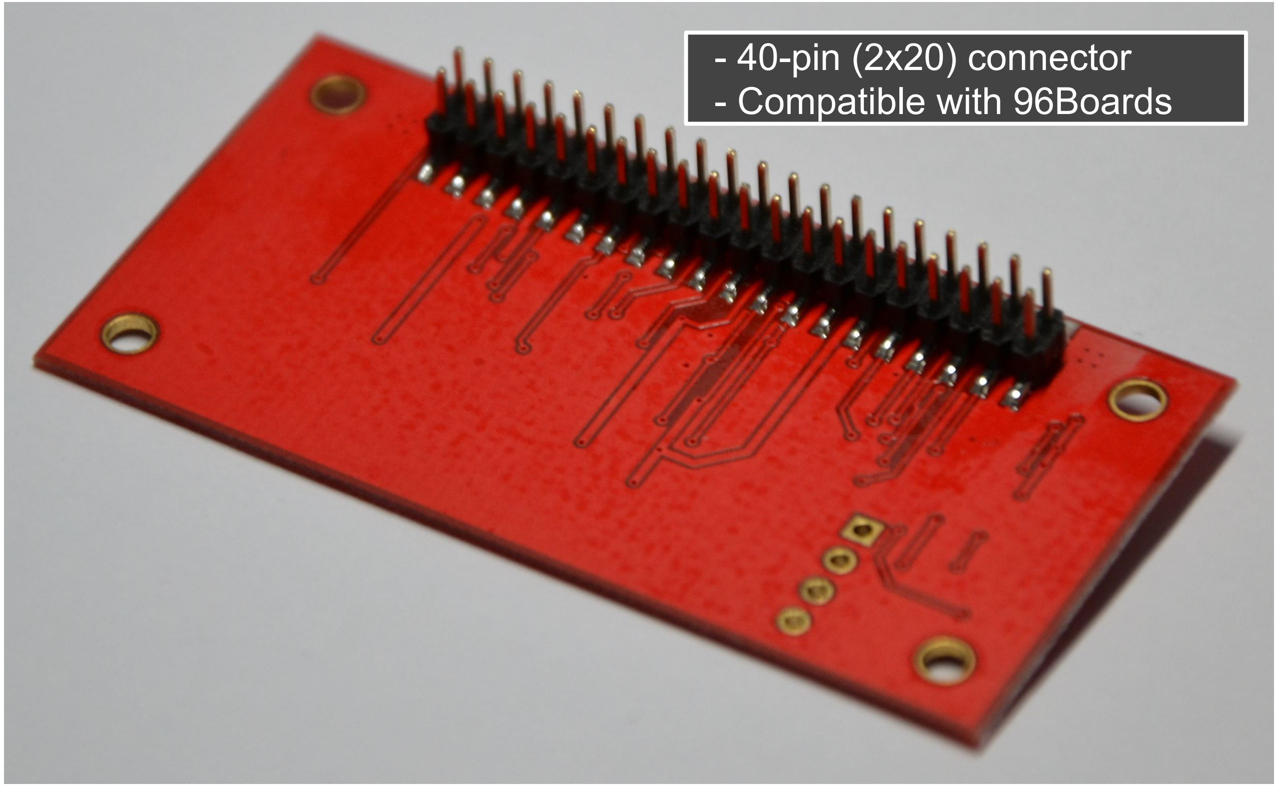


- SFO16 keynote demo
  - Used IoT devices with different crypto capabilities
  - **Main concern:** how to store keys in a secure manner
- Found out about
  - Atmel CryptoAuthentication™ AT88CK590 Demo-evaluation Kit
  - ATECC508A (asymmetric) got my attention, but later on also ATSHA204A
  - Turnkey (?) 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

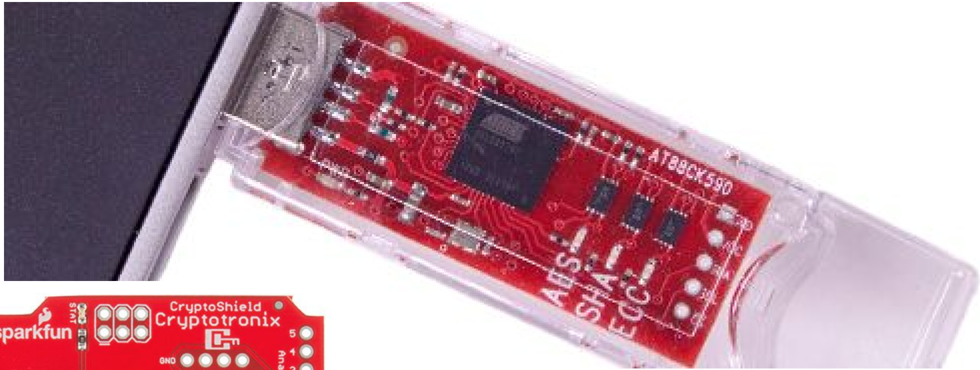


Secure96 - Bottom



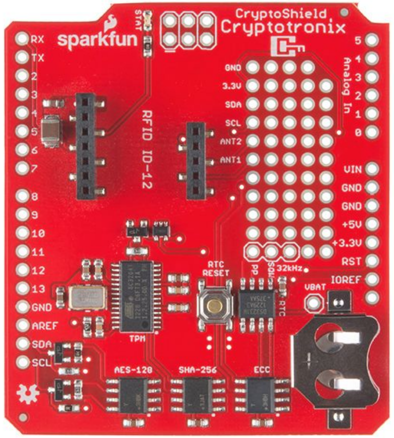
Similar hardware

- Atmel CryptoAuthentication™ [AT88CK590 Demo-evaluation Kit]



[Image courtesy of Microchip at http://www.atmel.com/tools/AT88CK590.aspx]

- [CryptoShield] by Josh Datko
  - No longer for sale
  - Nice software called “Hashlet”



[Image courtesy of SparkFun at https://www.sparkfun.com/products/retired/13183]

ATSHA204A

- **Symmetric authentication** (but still capable of deriving diversified keys)
- I/O: I^2C / 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)
- http://www.microchip.com/wwwproducts/en/ATSHA204A

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

Word	Byte 0	Byte 1	Byte 2	Byte 3	Default	Write Access	Read Access
0x00	Serial Number[0:3]				01 23 xx xx	Never	Always
0x01	Revision Number				xx xx xx xx	Never	Always
0x02	Serial Number[4:7]				xx xx xx xx	Never	Always
0x03	SN[8]	Reserved	I2C Enable	Reserved	EE 55 xx 00	Never	Always
0x04	I2C Address	CheckMacConfig	OTP Mode	Selector Mode	C8 00 55 00	If Config Is unlocked	Always
0x05	Slot Configuration 0		Slot Configuration 1		8F 80 80 A1	If Config Is unlocked	Always
0x06	Slot Configuration 2		Slot Configuration 3		82 E0 A3 60	If Config Is unlocked	Always
0x07	Slot Configuration 4		Slot Configuration 5		94 40 A0 85	If Config Is unlocked	Always
0x08	Slot Configuration 6		Slot Configuration 7		86 40 87 07	If Config Is unlocked	Always
0x09	Slot Configuration 8		Slot Configuration 9		0F 00 89 F2	If Config Is unlocked	Always
0x0A	Slot Configuration 10		Slot Configuration 11		8A 7A 0B 8B	If Config Is unlocked	Always
0x0B	Slot Configuration 12		Slot Configuration 13		0C 4C DD 4D	If Config Is unlocked	Always
0x0C	Slot Configuration 14		Slot Configuration 15		C2 42 AF 8F	If Config Is unlocked	Always
0x0D	Use Flag 0	Update Count 0	Use Flag 1	Update Count 1	FF 00 FF 00	If Config Is unlocked	Always
0x0E	Use Flag 2	Update Count 2	Use Flag 3	Update Count 3	FF 00 FF 00	If Config Is unlocked	Always
0x0F	Use Flag 4	Update Count 4	Use Flag 5	Update Count 5	FF 00 FF 00	If Config Is unlocked	Always
0x10	Use Flag 6	Update Count 6	Use Flag 7	Update Count 7	FF 00 FF 00	If Config Is unlocked	Always
0x11	Last Key Use 0	Last Key Use 1	Last Key Use 2	Last Key Use 3	FF FF FF FF	If Config Is unlocked	Always
0x12	Last Key Use 4	Last Key Use 5	Last Key Use 6	Last Key Use 7	FF FF FF FF	If Config Is unlocked	Always
0x13	Last Key Use 8	Last Key Use 9	Last Key Use 10	Last Key Use 11	FF FF FF FF	If Config Is unlocked	Always
0x14	Last Key Use 12	Last Key Use 13	Last Key Use 14	Last Key Use 15	FF FF FF FF	If Config Is unlocked	Always
0x15	User Extra	Selector	Lock Data	Lock Config	00 00 55 55	Via Update Extra Command Only	Always

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

Bit	Name	Description
0 – 3	ReadKey	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.
4	CheckOnly	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.
5	SingleUse	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.
6	EncryptRead	0 = Clear reads are permitted. 1 = Requires the slot to be Secret and encrypted read to access.
7	IsSecret	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.
8 – 11	WriteKey	Slot of the key to be used to validate encrypted writes.
12 – 15	Write Config	See detailed function definition for use.

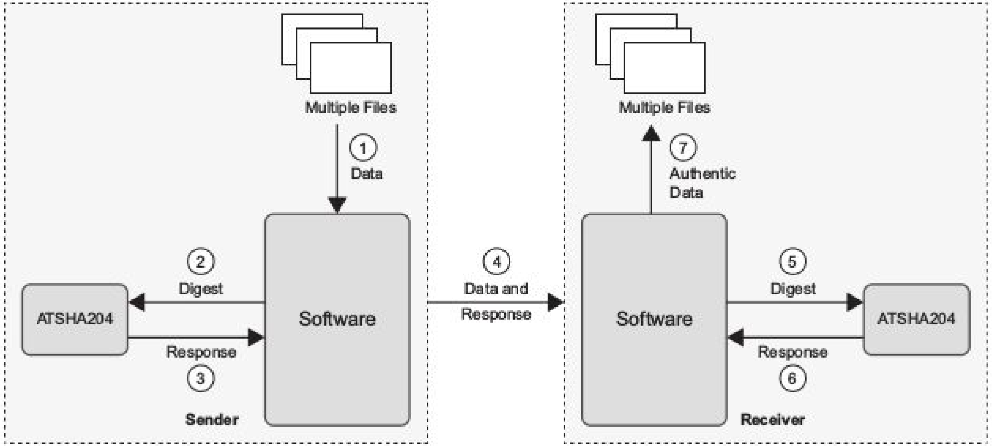
Personalize - Write Config bits

Bit15	Bit14	Bit13	Bit12	Write Command	Description	DeriveKey	Description
~~0~~	0	0	~~0~~	ALWAYS	Clear text writes ALWAYS permitted	-	-
0	0	0	~~1~~	ALWAYS	Clear text writes ALWAYS permitted	-	-
0	0	1	0	NEVER	Writes NEVER permitted	TARGET	DeriveKey command can be run without authorizing MAC (Roll).
0	0	1	1	NEVER	Writes NEVER permitted	PARENT	DeriveKey command can be run without authorizing MAC (Create).
0	1	0	~~0~~	ENCRYPT	Writes permitted if MAC is OK	-	-
0	1	0	~~1~~	ENCRYPT	Writes permitted if MAC is OK	-	-
0	1	1	0	ENCRYPT	Writes permitted if MAC is OK	TARGET	DeriveKey command can be run without authorizing MAC (Roll).
0	1	1	1	ENCRYPT	Writes permitted if MAC is OK	PARENT	DeriveKey command can be run without authorizing MAC (Create).
1	0	0	~~0~~	NEVER	Writes NEVER permitted	-	-
1	0	0	~~1~~	NEVER	Writes NEVER permitted	-	-
1	0	1	0	NEVER	Writes NEVER permitted	TARGET	Authorizing MAC required for DeriveKey command (Roll).
1	0	1	1	NEVER	Writes NEVER permitted	PARENT	Authorizing MAC required for DeriveKey command (Create).
1	1	0	~~0~~	ENCRYPT	Writes permitted if MAC is OK	-	-
1	1	0	~~1~~	ENCRYPT	Writes permitted if MAC is OK	-	-
1	1	1	0	ENCRYPT	Writes permitted if MAC is OK	TARGET	Authorizing MAC required for DeriveKey command (Roll).
1	1	1	1	ENCRYPT	Writes permitted if MAC is OK	PARENT	Authorizing MAC required for DeriveKey command (Create).

Use cases

Yes, I know it smells DRM ...

- Accessory authentication
  - Mobile device wants to authenticate a (genuine) battery
  - Consumables, ink cartridges etc
  - Technical support
- Secure boot
- Data integrity verification
- Session key exchange

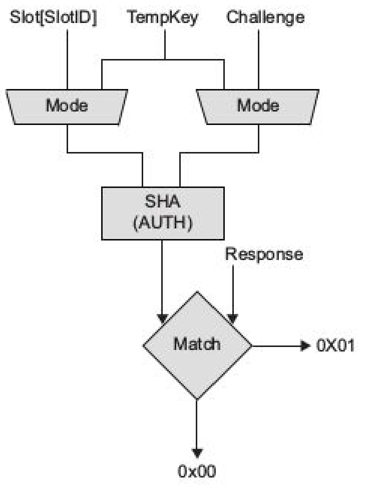


Data integrity verification example [Image courtesy of Atmel: Atmel-8794-CryptoAuth-ATSHA204-Product-Uses-Application-Note.pdf]

I^{2}C 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 CryptoAuthetication 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?

Figure 8-1. Data Flow for CheckMAC Command



Software for ATSHA204

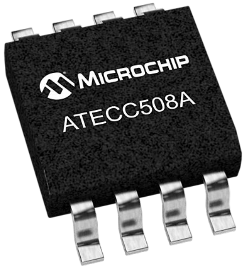
- Secure96→ https://github.com/jbech-linaro/secure96
  - In the future this should be moved to a “Linaro” page.
  - Still very much Work In Progress, but basic communication, read, write, getting config data, generate random numbers, nonce, HMAC etc has been implemented.
  - Personalization “works”, but still a bit crude.
  - Works with any I^{2}C enabled device, but currently configured so it works with DragonBoard 410c by default.

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

- CryptoAuthLib → http://www.atmel.com/tools/CryptoAuthLib.aspx
  - Atmel’s own reference implementation

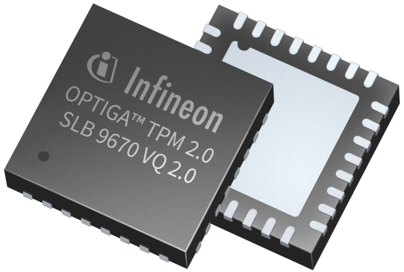
ATECC508A

- 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
- http://www.microchip.com/wwwproducts/en/ATECC508A



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



Linaro connect
San Francisco 2017

ENGINEERS AND DEVICES WORKING TOGETHER

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

Linaro connect
San Francisco 2017

Thank You

#SFO17

SFO17 keynotes and videos on: connect.linaro.org

For further information: www.linaro.org