TPM2.0 practical usage
Using a firmware TPM 2.0 on an embedded device

Davide Guerri - dguerri@fb.com
Production Engineer - Facebook London
Agenda

Trusted Platform Module 2.0: a practical example

- what is a TPM?
- using TPM2.0 (on a Minnowboard Max/Turbot)
- a practical example
  - generating a signing key on a TPM2.0
  - signing a document
  - verify a signature
What is a TPM?
What is a TPM

Overview

• TPM stands for **Trusted Platform Module**
• specs written by the **TCG**
  • AMD, Hewlett-Packard, IBM, Intel and Microsoft
  • standardised in **ISO/IEC 11889** (2009, TPM1.2)
• present in most computers, including embedded platforms
  • e.g. Microsoft mandated a TPM 2.0 for WM10
What is a TPM

Overview

- cryptographic processor
- **not an accelerator!**

believe it or not, **TPMs are slow “by design”**

because of import/export restriction on cryptographic technologies that some countries have
What is a TPM

Building blocks

- I/O
  - Cryptographic processing
    - (cs)RNG
    - Key generator
    - Hash Engine
    - Encryption Decryption Signature Engine

- Non-volatile storage
- General-purpose memory
What is a TPM

TPM1.2 vs TPM2.0

Key generator

Encryption Decryption
Signature Engine

Hash generator
digest + HMAC

TPM1.2
RSA 1024/2048
SHA-1

TPM2.0
RSA 1024/2048
ECC P256/BN256
SHA-1
SHA-256
*
What is a TPM

TPM typical usage

- **platform integrity (secure boot, trusted boot)**
  - is a computer platform in a trusted condition?
  - incrementally, from *power-on* to OS is *up and running*

- **disk encryption**
  - TPM stores and control access to the key

- **DRM**
  - *e.g.* verify code signature
What is a TPM

Types of TPM

- **hardware (discrete) TPM**
  - physical component
- **firmware TPM (fTPM)**
  - emulated TPM using an isolated HW environment named Trusted Execution Engine (TXE)
- **simulator**
  - software TPM in user space
Using TPM2.0
Software (x86)

Intel vs IBM TPM2.0-TSS (TPM software stack)- highlights

• IBM
  • TPM simulator running on Linux (can be used with Intel TSS)
  • source available on source forge
  • no Resource Manager
  • lots of tools

• Intel (undergoing some important improvements)
  • developed on Github (more “open”: PRs, etc...)
  • TCP implementation of the RM (in-kernel aimed for 4.11)
  • fewer tools
Hardware!

MinnowBoard Max / MinnowBoard Turbot

- dual Core Atom E3800 family Valleyview SoC
  - 1.33 GHz / 1.46 GHz
- 2 GB DDR3 RAM
- Intel HD Graphics (up to 1920x1080)
- UEFI system firmware
- fTPM 2.0 (not enabled in the OEM firmware)
- ~150 € (used to be sold on Amazon)
A practical example
Using TPM2.0 Tools

Foreword

• using TPM2.0 tools for “real world” applications is not easy
  • they don’t use widely supported formats like PEM or DER

• but the TSSes provide an API (SAPI) that can be used in your C/C++ apps, although the TCG spec is quite hard to digest

• let’s see how to use the Intel tooling to do something useful with a TPM2.0
Intel TPM2.0 Tools
What’s needed

- enable fTPM in UEFI configuration settings (PTT for MBM/T)
- set up Linux (> 4.4 preferred) any recent distro will do
  - flash it on a micro SD card
- install Intel TPM2.0-TSS (packages available for some distro)
  - this includes the Resource Manager daemon
- install Intel TPM2.0-Tools
Create a signing key
Endorsement Key

• Intel Tools won’t allow creating a primary signing key
• we need to create an EK and use that to generate a AIK

```bash
~# tpm2_getpubek -H 0x81010000 -g 0x01 -f ek.pub
```

• this will:
  • generate a 2048 RSA (0x01) key pair
  • store it in the NVM with handle 0x81010000
  • export the public part in `ek.pub`
Create a signing key

Attestation Identity Key

• create an AIK with the EK just created

```
~# tpm2_getpubak -E 0x81010000 -k 0x81010010 \ 
   -f aik.pub -n aik.name
```

• generates a 2048 RSA key pair using the EK with handle 0x81010000
• stores it in the NVM with handle 0x81010010
• exports the public part in ak.pub
• ak.pub is in a format described by the TGC standard
Create a signing key

OpenSSL conversion

• extract RSA modulus (skip TPMT_PUBLIC header)

```bash
~# dd if=aik.pub of=modulus.bin bs=1 skip=102 count=256
```

• create the DER fixed header and mid-header

```bash
~# echo 'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA' | openssl base64 -a -d > header.bin
~# echo -en '\x02\x03' > mid-header.bin
```
Create a signing key

OpenSSL conversion

• create the exponent (always 65537)

```
~# echo -ne '\x01\x00\x01' > exponent.bin
```

• compose the DER key!

```
~# cat header.bin modulus.bin mid-header.bin exponent.bin > aik-pub.der
```
Signing a document

OpenSSL conversion

• create an hash from the document
  • ticket.bin is used as a proof that the hash has been created by this TPM

```
~# tpm2_hash -H e -g 0x0B -I message.txt \ 
   -o hash.bin -t ticket.bin
```

• sign the hash

```
~# tpm2_sign -k 0x81010010 -g 0x0B -m message.txt \ 
   -s sign.bin -t ticket.bin
```
Verify a signature

OpenSSL conversion

• extract the "raw" signature

```bash
~# dd if=sign.bin of=sign.raw bs=1 skip=6 count=256
```

• verify the signature

```bash
~# openssl dgst -verify aik-pub.der -keyform der \ 
   -sha256 -signature sign.raw message.txt
```

Verified OK
Thanks!
References

**TPM2.0 Library specification**
https://fb.me/tpm2-spec

**Intel TPM2.0-TSS and Tools**
https://fb.me/intel-tpm2-tss
https://fb.me/intel-tpm2-tools

**enabling fTPM on MinnowBoard Max/Turbot**
https://fb.me/ftpm-on-mbm

**RSA signatures with TPM2.0 and OpenSSL**
https://fb.me/tpm2-openssl