Enabling the TPM2.0 Ecosystem in Linux

Andreas Fuchs
<andreas.fuchs@sit.fraunhofer.de>

(filling in for Peter Huewe)
Who am I?

- 13 year on/off TPMs
- Fraunhofer SIT: Trustworthy Platforms
- TCG-member: TPM Software Stack WG
- Maintainer
  - tpm2-tss: The libraries
  - tpm2-tss-engine: The OpenSSL engine
  - tpm2-totp: Computer-to-user attestation (mjg’s tpm-totp reimplemented for 2.0)
Putting TPMs into things
Agenda

- Introducing TPMs
- Introducing the TSS
- Existing TSS software
- What’s new?
- Community
Introduction to TPMs

• **Trusted Platform Module (TPM) 2.0**
  - Smartcard-like capabilities but soldered in
  - Remote Attestation capabilities
  - As separate chip (LPC, SPI, I²C)
  - In Southbridge / Firmware
  - Via TEEs/TrustZone, etc
  - Thanks to Windows-Logos in every PC

• **CPU**
  - OS, TSS 2.0, applications
Introduction to TPMs

• Getting started
  – Any PC with a Windows Logo ACPI based discovery
  – A Raspberry-PI with a TPM daughterboard
    On Raspbian:
    /boot/config.txt:
    dtparam=spi=on
dtoverlay=tpm-slb9670

• tpm2-tss/INSTALL.md
Agenda

- Introducing TPMs
- **Introducing the TSS**
- Existing TSS software
- What’s new ?
- Community
The TPM Software Stack 2.0

- Kernel exposes /dev/tpm0 with byte buffers
- TCG specifications:
  - TPM spec for functionality
  - TSS spec for software API
- tpm2-tss is an implementation
- APIs for application integration
- Support in other modules and middlewares for seamless integration
# The TSS APIs

## System API (sys)
- 1:1 to TPM2 cmds
- Cmd / Rsp serialization
- No file I/O
- No crypto
- No heap / malloc

## Enhanced SYS (esys)
- Automate crypto for HMAC / encrypted sessions
- Dynamic TCTI loading
- Memory allocations
- No file I/O

## Feature API (FAPI)
- Spec in draft form
- No custom typedefs
- JSON interfaces
- Provides Policy language
- Provides keystore
- Sec/func separation

## TPM Command Transmission Interface (tss2-tcti)
- Abstract command / response mechanism,
- Decouple APIs from command transport / IPC
- No crypto, heap, file I/O
- Dynamic loading / dlopen API

## TPM Access Broker and Resource Manager (TAB/RM)
- Abstract Storage Limitations
- No crypto
- Power management

## TPM Device Driver
- Device Interface (CRB / polling)
- Pre-boot log handoff
Projects overview

- **tpm2-tss**
  - **tpm2-tools**
    - Commandline Tools
  - **tpm2-abrmd**
    - Access Broker/Resource Manager
  - **tpm2-tss-engine**
    - OpenSSL Engine
  - **tpm2-pkcs11** *
    - PKCS#11 Provider
  - **tpm2-totp**
    - One Time Pads
  - **tpm2-pytss** *
    - Python Bindings

- **Cryptsetup**
  - LUKS *
    - Disk-Encryption

- **Keylime**
  - Remote Attestation

- **OpenConnect**
  - VPN

- **StrongSwan**
  - IPSec

- tpm2-software.github.io
The tpm2-software core projects

- **tpm2-tss (core library)**
  - Autotools, pkg-config, deps: libcrypto OR libgcrypt
    coming deps: libcurl, libjson-c
- **tpm2-abrmd (user space RM)**
  - Autotools, pkg-config, deps: libdbus, libglib
- **tpm2-tools (CLI tools)**
  - Autotools, pkg-config, deps: libcrypto, libcurl
- **tpm2-pytss (python bindings)**
- **tpm2-tss-engine (OpenSSL-engine)**
- **tpm2-tottp (PC-to-human authentication)**
People and community

• **Maintainers:**
  - Bill, Imran, Jonas, Jürgen, John, Phil, Peter, Tadeusz, and me

• >100 contributors

• CI with ~80% coverage targets, scanbuild, coverity, CII best practice, lgtm, ...

• Building multi-distro CI using docker
• Tested in many ways

• Packaged for many distros
Agenda

- Introducing TPMs
- Introducing the TSS
- **Existing TSS software**
- What’s new ?
- Community
Existing TSS things

- Mostly anything runs off of Esys_*(*)
- Provides 1-to-1 mapping of TPM functionality
- Automates
  - marshalling / unmarshaling
  - Object meta-data handling
  - session encryption and authentication
  - memory allocation
  - TPM detection sequence
    (tpm2-abrmd, /dev/tpmrm0, /dev/tpm0, simulator)
- tpm2-tools >= 4.0 use Esys
- tpm2-pytss uses Esys
UC: Shielded key storage and usage

• Keys in RAM are always dangerous
  – “Heartbleed”

• Keys on Disk are always dangerous
  – You can protect them with user passwords but they can be bruteforced
  – Servers have no unlock step
  – Embedded devices have no unlock step

• So how do you prevent ID- cloning ?
  → Use TPM
How do you use the TPM?

→ easy: tpm2-tss-engine
UC: Disk encryption

• “Bitlocker for Linux”
  - Binding the disk to the machine
  - Short PIN instead of long passwords
  - No more dictionary attacks

• Even more utility in other areas
  - Data Center: People stealing HDDs from the rack
  - Embedded device once more
  - Binding to BIOS integrity status (local attestation)
UC: Disk encryption

- cryptsetup(-tpm) / LUKS2
  - Rearchitecting with Milan
  - Making cryptsetup “module-aware”

```bash
afuchs@pc-fuchslap3:~/Dokumente/oss-tss/cryptsetup-tpm-incubator$ ./cryptsetup luksFormat --type=luks2 --tpm disk.img
WARNING!
--------
Hiermit werden die Daten auf »disk.img« unwiderruflich überschrieben.

Are you sure? (Type uppercase yes): YES
Geben Sie die Passphrase für »disk.img« ein:
Passphrase bestätigen:
afuchs@pc-fuchslap3:~/Dokumente/oss-tss/cryptsetup-tpm-incubator$ ./cryptsetup luksOpen disk.img --test-passphrase
Geben Sie die Passphrase für »disk.img« ein:
afuchs@pc-fuchslap3:~/Dokumente/oss-tss/cryptsetup-tpm-incubator$
```

```json
{  "keyslots": {  "1": {  "type": "tpm2",  "key_size": 32,  "area": {  "type": "tpm2nv",  "nvindex": 29294593,  "pcrselection": 0,  "pcrbanks": 1,  "noda": true  } },
```
UC: (VPN) user authentication

- UserName + Password ? → Machine + UserPassword !

Adding security to network access

- OpenConnect (David Woodhouse)
  - Reuse (copy) of tpm2-tss-engine

- Strongswan
  - Implements Attestation and RIMs as well

- OpenVPN via tpm2-tss-engine ?

- Missing WireGuard, Tinc, ...
Agenda

- Introducing TPMs
- Introducing the TSS
- Existing TSS software
- What’s new?
- Community
What’s new? libtss2-fapi.so!

- **TCG specifications**
  - TSS 2.0 Feature API spec
  - TSS 2.0 JSON and Policy Data spec

- **Features**
  - No TPM-specific data structures:
    Using JSON for all in-/output
  - Decouple functional design from security design:
    Using cryptographic profiles
  - Add a keystore:
    Store TPM’s blobs and meta data on disk
  - Add a policy language:
    Describe policies in JSON and automatically evaluate policies upon use

- **PRs with >25k LoC on tpm2-tss, -tools, -pkcs11**
API Code comparison (Signing)

**FAPI** [2 lines of code]

```c
Fapi_Sign_async(fctx, "name/of/my/key", payload);
Fapi_Sign_finish(fctx, &signature);
```

**ESAPI** [9 lines of code]

```c
Esys_TR_SetAuthValue(ectx, srkTR, authValue);
Esys_StartAuthSession(ectx, srkTR, &sessionTR);
Esys_Load_async(ectx, srkTR, sessionTR, keyblob);
Esys_Load_finish(ectx, &keyTR);
Esys_TR_SetAuthValue(ectx, keyTR, authValue);
Esys_RSA_Sign_async(ectx, srkTR, sessionTR, parameters);
Esys_RSA_Sign_finish(ectx, &params);
Esys_FlushContext(ectx, sessionTR);
Esys_FlushContext(ectx, keyTR);
```

**SAPI** [32 lines of code]

```c
Application_EncryptSalt(session_salt, &encryptedSalt);
Tss2_Sys_StartAuthSession(sctx, srkHandle, encryptedSalt, &sessionHandle, &tpmNonce);
Application_CalculateSessionKey(session_salt, &session_key);
Tss2_Sys_Load_Prepare(sctx, srkHandle, keyblob);
Application_GetMetaData(session.key, session_nonceTPM, srkName, keyName);
Tss2_Sys_GetCommandCode(sctx, &cc);
Tss2_Sys_GetRspBuffer(sctx, &buffer);
Tss2_Sys_GetDecrptParam();
Application_EncryptParamter();
Tss2_Sys_DecryptParam();
Application_CalculateCpHash(cc, srkName, keyName, buffer, &cpHash);
Application_CalculateHmac(session.key, session_nonceTPM, myNonce, cpHash, authValue, &hmac);
Tss2_Sys_SetCmdAuths(sctx, sessionHandle, hmac);
Tss2_Sys_Execute(sctx);
Tss2_Sys_GetRspAuths(sctx, &rspAuths);
Tss2_Sys_GetRspBuffer(sctx, &buffer);
Application_CalculateRspHash(cc, keyName, rbuffer, &rHash);
Application_VerifyHmac(session.key, myNonce, cc, rHash, rspAuth, &tpmNonce);
Tss2_Sys_Load_Finish(sctx, &keyHandle);
Tss2_Sys_RSA_Sign_Prepare(sctx, keyHandle, parameter);
Tss2_Sys_GetCommandCode(sctx, &cc);
Tss2_Sys_GetRspBuffer(sctx, &buffer);
Application_CalculateCpHash(cc, keyName, buffer, &cpHash);
Application_CalculateHmac(session.key, session_nonceTPM, myNonce, cpHash, authValue, &hmac);
Tss2_Sys_SetCmdAuths(sctx, sessionHandle, hmac);
Tss2_Sys_Execute(sctx);
Tss2_Sys_GetRspAuths(sctx, &rspAuths);
Tss2_Sys_GetRspBuffer(sctx, &buffer);
Application_CalculateRspHash(cc, keyName, rbuffer, &rHash);
Application_VerifyHmac(session.key, myNonce, cc, rHash, rspAuth, &tpmNonce);
Tss2_Sys_RSA_SignFinish(sctx, &params);
Tss2_Sys_FlushContext(sctx, sessionHandle);
Tss2_Sys_FlushContext(sctx, KeyHandle);
```
Policy Comparison

• ESYS:

TPML_PCR_SELECTION pcrSelection = {
  .count = 1, .pcrSelections = {
    .hash = TPM2_ALG_SHA1,
    .sizeofSelect = 3,
    .pcrSelect = {00, 00, 01} }
};

TPM2B_DIGEST pcr_digest_zero = {
  .size = 20, .buffer = {0x67, 0x68, 0x03, 0x3e, 0x21, 0x64, 0x68, 0x24, 0x7b, 0xd0, 0x31, 0xa0, 0xa2, 0xd9, 0x87, 0x6d, 0x79, 0x81, 0x8f, 0x8f}};

r = Esys_StartAuthSession(esys_context, 
ESYS_TR_NONE, ESYS_TR_NONE, 
ESYS_TR_NONE, ESYS_TR_NONE, 
ESYS_TR_NONE, &nonceCallerTrial, 
TPM2_SE_POLICY, &symmetric, 
TPM2_ALG_SHA1, &session);

r = Esys_PolicyPCR(esys_context, session, 
ESYS_TR_NONE, ESYS_TR_NONE, 
ESYS_TR_NONE, &pcr_digest_zero, 
&pcrSelection);

• FAPI:

{
  "description":"PCR 16 value", 
  "policy":[
    {
      "type":"POLICYPCR", 
      "pcrs":[
        {
          "pcr":16, 
          "hashAlg":"SHA1", 
          " digest":"00...00"
        }
      ]
    }
  ]
}
UC: (General) user authentication

- **Typical SmartCard workflow (PKCS11)**
  - Proof of possession (of smartcard)
  - Proof of knowledge (of PIN not password)
  - More secure and convenient than passwords
  → **tpm2-pkcs11 (Virtual SmartCard)**
  - Proof of possession (of TPM-holding device)
  - Proof of knowledge
  - Fully compatible

- **Heavy rework to run off of FAPI**
Agenda

- Introducing TPMs
- Introducing the TSS
- Existing TSS software
- What’s new?
- Community
Welcome to the tpm2-software community page.

This is the home for information related to the use of the TPM2 using the TPM2-tools. The website collects links to software that already contains support for TPM2 as well as ideas and benefits from TPM2 support. We provide links to external / educational resources on how to work with the TPM2.

The core software of this community can be found at:
- [https://github.com/tpm2-software](https://github.com/tpm2-software)

The mailing list for this community can be found at:
- [https://lists.01.org/postorius/lists/tpm2-lists.01.org](https://lists.01.org/postorius/lists/tpm2-lists.01.org)

The gitter can be found at the bottom right or at
- [https://gitter.im/tpm2-software/community](https://gitter.im/tpm2-software/community)

Contributing

Contributions to this site are always wanted and highly welcome – from additional tutorials. Please feel free to fork and send merge requests on GitHub:
- [https://github.com/tpm2-software/tpm2-software.github.io](https://github.com/tpm2-software/tpm2-software.github.io)

Also, each of the core and non-core project has a lot of enhancement issues that started in their issue trackers.
What’s missing?

- **Attestation**
  - Some support by FAPI; protocol bindings
  - Reference value descriptions

- **More core system integration**
  - 802.1X: NetworkManager, systemd-networkd
  - User keyrings: gnome-keyring, kwallet
  - VPNs: Wireguard, Tinc, ...
  - Signing: GnuPG
  - WebCrypto / WebAuthn (Firefox, Chrome, ...)
  - .......

- **2nd maintainer for tpm2-tss-engine :-)**
Questions?

https://tpm2-software.github.io