TPM Software Stack: Enabling the TPM2.0 Ecosystem in Linux

Peter Huewe, Infineon Technologies AG, @PeterHuewe
Joshua Lock, Intel
Agenda

› Background
› Design & Architecture
› OSS Implementation, Community and Adoption
› TSS2 Use Cases and Examples
### Speakers’ Bios

<table>
<thead>
<tr>
<th>Peter Hüwe, Infineon Technologies AG</th>
<th>Joshua G. Lock, Intel</th>
</tr>
</thead>
<tbody>
<tr>
<td>› Senior Staff Engineer - Embedded Security Software, TPM Firmware &amp; Linux Security</td>
<td>› Software Engineer – Open Source Technology Center</td>
</tr>
<tr>
<td>› Project Lead for sponsored ESAPI development by FHG SIT</td>
<td>› Co-maintainer of tpm2-tools</td>
</tr>
<tr>
<td>› TPM Subsystem Maintainer (retired?)</td>
<td>› Long-time Yocto Project contributor</td>
</tr>
<tr>
<td>› Contributor to tpm2-software</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Philip Tricca, Intel</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSS2 Software Stack (TSS2) @ Linux Security Summit NA 2018</td>
</tr>
<tr>
<td><strong>Thanks for the Slide Content!</strong></td>
</tr>
</tbody>
</table>
TPM2.0 Background

› See materials & book by Ariel Siegal [1][2]
› TPM-JS Tutorial by Google [3]
› Use-case unchanged
   - Protect encryption keys while in use
   - Root of trust for storage & reporting
› TPM 1.2 limited algorithm support
   - Mandates RSA 1k, 2k & SHA1, no larger key / hash sizes, AES optional
   - Single hierarchy, limited policy
› TPM 2.0 addresses shortcomings of 1.2
   - Adds cryptoagility and support for complex policies
   - Integrity protected and encrypted sessions
TPM2 Software Stack
Design & Architecture
TSS2 Design

› Layered design
  – Separate transport layer from APIs:
  – Both synchronous and asynchronous: event-driven programming
  – Details exposed if you need them, “sane defaults” otherwise
    – Chosen by: TCG / platform / distro / OS?
› Lower layers of stack provide data transport & thin layer over TPM2 commands
  – “Expert” applications in constrained environments
  – Minimal dependencies (libc)
› Upper layers provide convenience functions & abstractions
  – Crypto for sessions, dynamic memory allocation, transport layer configuration
  – More Features -> more dependencies
## Architecture / Design

<table>
<thead>
<tr>
<th>System API (tss2-sys)</th>
<th>Enhanced SAPI (tss2-esys)</th>
<th>Feature API (FAPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1 to TPM2 cmds</td>
<td>Automate crypto for HMAC / encrypted sessions</td>
<td>Spec in draft form</td>
</tr>
<tr>
<td>Command / Response serialization</td>
<td>Dynamic TCTI loading</td>
<td>No implementation yet</td>
</tr>
<tr>
<td>No file I/O</td>
<td>Requires heap / does memory allocations</td>
<td>File I/O</td>
</tr>
<tr>
<td>No crypto</td>
<td>No file I/O</td>
<td>Requires heap</td>
</tr>
<tr>
<td>No heap / malloc</td>
<td></td>
<td>Automate retries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Context based state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Must support static linking</td>
</tr>
</tbody>
</table>

### 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)
- Power management
- Potentially no file IO – depends on power mgmt.
- Abstract Limitations of TPM Storage
- No crypto

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

---

Source: Philip Tricca, Intel, Linux Security Summit 2018 – with Permission
TSS2 Application
System API, Type Marshalling

› **System API**: libtss2-sys
  - Translate C types to TPM command layout
  - One-to-one mapping to TPM commands
  - Suitable for **firmware** / **embedded** applications e.g. available for *Aurix™* and *XMC™* Microcontrollers

› **Type Marshalling**: libtss2-mu
  - Transform TPM types from C to wire format & back
TSS2 Application
Enhanced System API: libtss2-esys

- **Enhanced System API: libtss2-esys**
  - Builds on top of lower-level tss2-* libs
  - Exposes all TPM2 functions + utility functions
    - HMAC calculations for HMAC session
    - Crypto for encrypted session
    - Maintain state for authorizations
  - Adds crypto library dependency
    - Current implementation supports
      - libgcrypt
      - Openssl

Contribution by Fraunhofer SIT, sponsored by Infineon Technologies
TSS2 Application
TPM Command Transmission Interface (TCTI)

› **TPM2 Command Transmission Interface:**
libtss2-tcti-xxx

› Modular, dynamically loaded transport layer. Decouples API from command transport/IPC.
  - device: `/dev/tpmX` (no AB & RM) or `/dev/tpmrmX` (no AB & some RM)
  - `abrmd`: Access Broker and Resource Manager Daemon
  - `tbs`: TPM Base Services on Windows 2008/Vista+
  - `mssim`: IBM's Linux port of Microsoft's TPM2 simulator
  - `efi`: enabling use of tss2-sys API in UEFI (*in development*)
TPM2 Resource Management

TPMs are resource constrained: small & inexpensive

› RAM on the order of “a few kilobytes”
› Scarce resources must be shared
  – TPM commands specific to object and session management:
    – ContextLoad, ContextSave & FlushContext
    – Resource Management: Saving & Loading “contexts”
› Isolation through Resource Management
  – Associate objects (keys, session) with connection
  – Prevent access by other connections (with exceptions)
› Components of resource mgmt. tasks moving into kernel driver
  – /dev/tpmrm0: performs simple object / session isolation & load / save
  – Aligning user-space daemon w/ in-kernel resource mgmt. (ongoing work)
OSS Implementation, Community and Adoption
From Prototype to OSS Project

Stability & Reliability

› Eliminate liabilities / high priority technical debt
  - Make it debuggable
  - Use the right tools for the task
  - Complete re-write of resource mgmt. daemon

› Model a healthy OSS project
  - Friendly to packaging for distros
  - Semantic versioning scheme: https://semver.org
  - Testing: unit & integration, make adding new tests easy
  - Continuous Integration (CI): travis-ci, coveralls, coverity & scan-build
Community forming around development and use of TSS2 APIs

- TPM2 Software Github Org: https://github.com/tpm2-software
- Mailing list: https://lists.01.org/mailman/listinfo/tpm2
- Core libraries: https://github.com/tpm2-software/tpm2-tss
- Command line tools: https://github.com/tpm2-software/tpm2-tools
- OpenSSL Engine: https://github.com/tpm2-software/tpm2-openssl-engine
- Resource Mgmt: https://github.com/tpm2-software/tpm2-abrmd
- PKCS#11 Engine: https://github.com/tpm2-software/tpm2-pkcs11 (*NEW*BETA*)

Community

- Maintainers from: Intel, Fraunhofer SIT, RedHat
- Contributions from: Infineon, Facebook, Alibaba, RedHat, GE, Suse, Debian

New Projects

- UEFI TCTI, cryptsetup integration, RC decoding library & spec
TPM2-TSS Commit activities

Addition of ESAPI by Fraunhofer SIT sponsored by Infineon Technologies
Downstream adoption

- Packaging for distros
  - RHEL, Suse, Fedora, Debian, Ubuntu
  - 2.0 TSS2 release *should* make it into RHEL 8, missed SLES 15 😞
- OpenEmbedded upstreaming effort underway
  - Maintained as part of meta-measured
  - Planning effort to upstream into OE proper: reduce duplication
Changelog

› Version 2.0.0 released on 2018-06-20
  - Compatibility with TPM 2.0 v1.38 spec
  - Support for some commands from 1.46 (Attached Component)
› New libraries / APIs
  - Type marshalling library: libtss2-mu
  - Enhanced System API: libtss2-esys (by FHG SIT, sponsored by IFX)
› Version 2.1.0 released on 2018-10-01 added Windows support for core
  - TCTI for communication with TBS: libtss2-tcti-tbs (by Facebook)
  - CI using AppVeyor
TSS2 use cases
Bootstrapping & Expanding Community
TPM Use cases / Examples

TSS2 built & installed ... “Now what?”

› Reduce learning curve

› What TPMs are good for:
  – Data protection: root of trust for storage
  – Attestation: root of trust for reporting
  – Cryptography: protected keys & operations

› Start with basic crypto operations
  – No code required (maybe a little scripting)
  – Key creation & use
  – Interface to more familiar tools
tpm2-tools

- Command line tools for TPM2 operations
  - [https://github.com/tpm2-software/tpm2-tools](https://github.com/tpm2-software/tpm2-tools)
  - Oftentimes a user’s first experience with the TSS2
  - Started as a clone of the IBM command line tools from TSS for TPM 1.2
  - Has evolved into a near 1:1 mapping to TPM2 commands
  - Individual tool execs can be strung together to achieve a higher level task
    - Create policy assertion
    - Create object bound by policy
    - Save object to disk
    - ...
tpm2-tools

Next major release of the tools features:

- port to new Enhanced System API
  - can implement HMAC/encrypted sessions without providing own crypto implementation
- improved ease of use:
  - sane defaults
  - unified options
  - import/export objects in standard formats: DER & PEM
tpm2-tools example:
Sign data with TPM2 key / verify signature with OpenSSL

Refresh example from Davide Guerri @ FOSDEM 2017 [4]

<table>
<thead>
<tr>
<th>1) Create primary key in storage hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpm2_createprimary --hierarchy o --out-context pri.ctx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2) Create subkey for signing</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpm2_create --context-parent pri.ctx --pubfile sub.pub --privfile sub.priv</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3) Load subkey</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpm2_load --context-parent pri.ctx --pubfile sub.pub --privfile sub.priv --out-context sub.ctx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4) Calculate the hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>openssl dgst -sha1 -binary -out hash.bin msg.txt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5) Sign the hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpm2_sign --key-context sub.ctx --format plain --digest hash.bin --sig hash.plain</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6) Create OpenSSL compatible DER encoded public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpm2_readpublic -c sub.ctx --format der -out-file sub-pub.der</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7) Verify the signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>openssl dgst -verify sub-pub.der -keyform der -sha1 -signature hash.plain msg.txt</td>
</tr>
</tbody>
</table>
tpm2-tss-engine

- Cryptographic engine for OpenSSL backed by the TPM.
  - [https://github.com/tpm2-software/tpm2-tss-engine](https://github.com/tpm2-software/tpm2-tss-engine)
  - Still no v1.0 release

- Currently supports:
  - RSA decryption
  - RSA signatures
  - ECDSA signatures
  - more to come…
tpm2-engine example:
Sign data and verify signature with OpenSSL (using TPM2 engine)

Same use-case as previous example using tpm2-tools

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Create an RSA key</td>
<td>tpm2tss-genkey -a rsa -s 3072 key.bin</td>
</tr>
<tr>
<td>2) Export public key in PEM format</td>
<td>openssl rsa -engine tpm2tss -inform engine -in key.bin -pubout -outform pem -out key.pem</td>
</tr>
<tr>
<td>4) Calculate the hash</td>
<td>openssl dgst –sha256 –out hash.txt message.txt</td>
</tr>
<tr>
<td>5) Sign the hash</td>
<td>openssl pkeyutl -engine tpm2tss -keyform engine -inkey key.bin -sign -in hash.txt -out sig.bin</td>
</tr>
<tr>
<td>7) Verify the signature</td>
<td>openssl dgst -verify -pubin key.pem -sigfile sign.bin -in hash.txt</td>
</tr>
</tbody>
</table>
tpm2-pkcs11

- PKCS#11 Provider/Wrapper for the TPM
  - [https://github.com/tpm2-software/tpm2-pkcs11](https://github.com/tpm2-software/tpm2-pkcs11)
  - *BETA* - not ready for production
  - Based on PKCS#11 TPM2.0 work by irtimmer

- Currently status:
  - Works with OpenSSL and P11kit, more or less
  - Still a lot of bugs and unimplemented features
  - Not 100% spec conforming yet
  - Internal data format not yet fixed.
  - SAPI based

⇒ HELP Wanted!
tpm2-pkcs11 example: Use a TPM backed key for SSH authentication

1) Initialise a tpm2-pkcs11 store with a primary object slot
   tpm2_ptool.py init --pobj-pin myobjpin

2) Add a token with the unique name “ssh”
   tpm2_ptool.py addtoken --pid=1 --pobj-pin=myobjpin --sopin=mysopin --userpin=myuserpin --label=ssh

3) Add a key to the token for SSH to use
   tpm2_ptool.py addkey --algorithm=rsa2048 --label=ssh --userpin=myuserpin

4) Generate the SSH key public portion
   ssh-keygen -D /usr/lib64/pkcs11/libtpm2_pkcs11.so | tee my.pub

5) Configure SSH to accept the key (on the server)
   ssh-copy-id my.pub user@host

6) Start a secure shell session
   ssh -I /usr/lib64/pkcs11/libtpm2_pkcs11.so user@host
   Enter PIN for 'ssh': myuserpin
   Last Login: Fri Sep 21 13:28:31 2018 from somehost
Further Downstream Projects

- clevis: auto unlock of LUKS encrypted disks [5]
- strongswan: read X.509 certificates stored in the TPM [6]
- openconnect: TPM wrapped private keys locked to the TPM device, via OpenSSL engine or GnuTLS (in-development, coming in 8.0 release) [7]
- cryptsetup/LUKS: store LUKS key in the TPM and use password/PCR authorization (in-development) [8]
Your Turn – Help Wanted
Working with TSS

› Develop against the simulator
› Prototype with tpm2-tools
› Use tpm2-software projects (BSD-3-Clause) for reference
› Use the debugger
  – Can put both the application and the simulator in the debugger and compare the expected state to try and reverse engineer where things have gone wrong

› Once everything works try with a real TPM
  – SLB9670 Iridium Board [9], Infineon
  – Letstrust TPM [10], Buyzero (with SLB 9670)
  – Intel discrete TPM
  – Intel PTT fTPM
Help wanted

More software could enable TPM2.0 support via our stack, e.g.:

- OpenVPN
- WireGuard
- Tinc
- NetworkManager/wpa_supplicant 802.1X
- gnome-keyring
- KDE wallet
- GNU TLS
- mbedTLS
- Language Bindings!

/* insert your project here */
References

2. https://www.theiet.org/resources/books/computing/tpmwhy.cfm
5. https://blog.dowhile0.org/2017/10/18/automatic-luks-volumes-unlocking-using-a-tpm2-chip/
8. https://github.com/AndreasFuchsSIT/cryptsetup-tpm-incubator
Part of your life. Part of tomorrow.