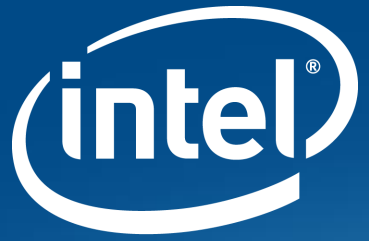


Software

TPM2 SOFTWARE STACK (TSS2)

Philip Tricca philip.b.tricca@intel.com
@flihp
https://github.com/flihp

AGENDA

TSS2: standardization and OSS implementation

- Background
- TSS2 design
  - Goals & Use Cases
  - Components / Architecture
- tpm2-software community
  - Purpose / goals
  - Community building & adoption
- Use-cases & examples
  - Managing the gap between building & *using* the TSS2
  - Highlight TSS2 flexibility

BACKGROUND

Background on TPM, use-cases etc

- See materials & book by Ariel Siegal [1][2]
- Use-case unchanged
  - Protect encryption keys while in use
  - Root of trust for storage & reporting
- TPM 1.2 limited algorithm support
  - Require RSA 1k, 2k & SHA1, no larger key / hash sizes, AES optional
  - Single hierarchy, limited policy
- TPM 2.0 addresses shortcomings of 1.2
  - Flexible to support multiple algorithms & policy
  - Integrity protected and encrypted sessions

TPM2 SOFTWARE STACK (TSS2)

Design

TSS2 DESIGN

Use-case driven [3]

- Layered design
  - Separate transport layer from APIs:
  - Both synchronous and async: 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

TSS2 DESIGN

System API (tss2-sys)
- 1:1 to TPM2 cmds
- Command / Response serialization
- No file I/O
- No crypto
- No heap / malloc

Enhanced SAPI (tss2-esys)
- Automate crypto for HMAC / encrypted sessions
- Dynamic TCTI loading
- Requires heap / does memory allocations
- No file I/O

Feature API (FAPI)
- Spec in draft form
- No implementation yet
- File I/O
- Requires heap
- Automate retries
- Context based state
- Must support static linking

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

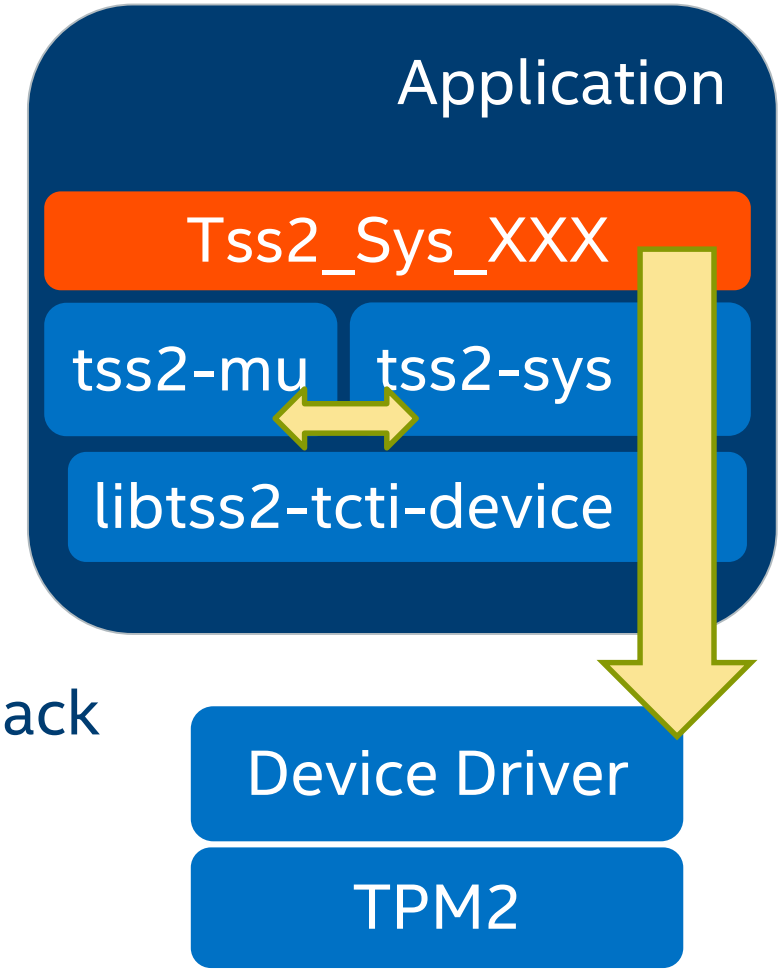
User Space

Kernel

TSS2 APPLICATION

System API, Type Marshaling, & TCTI

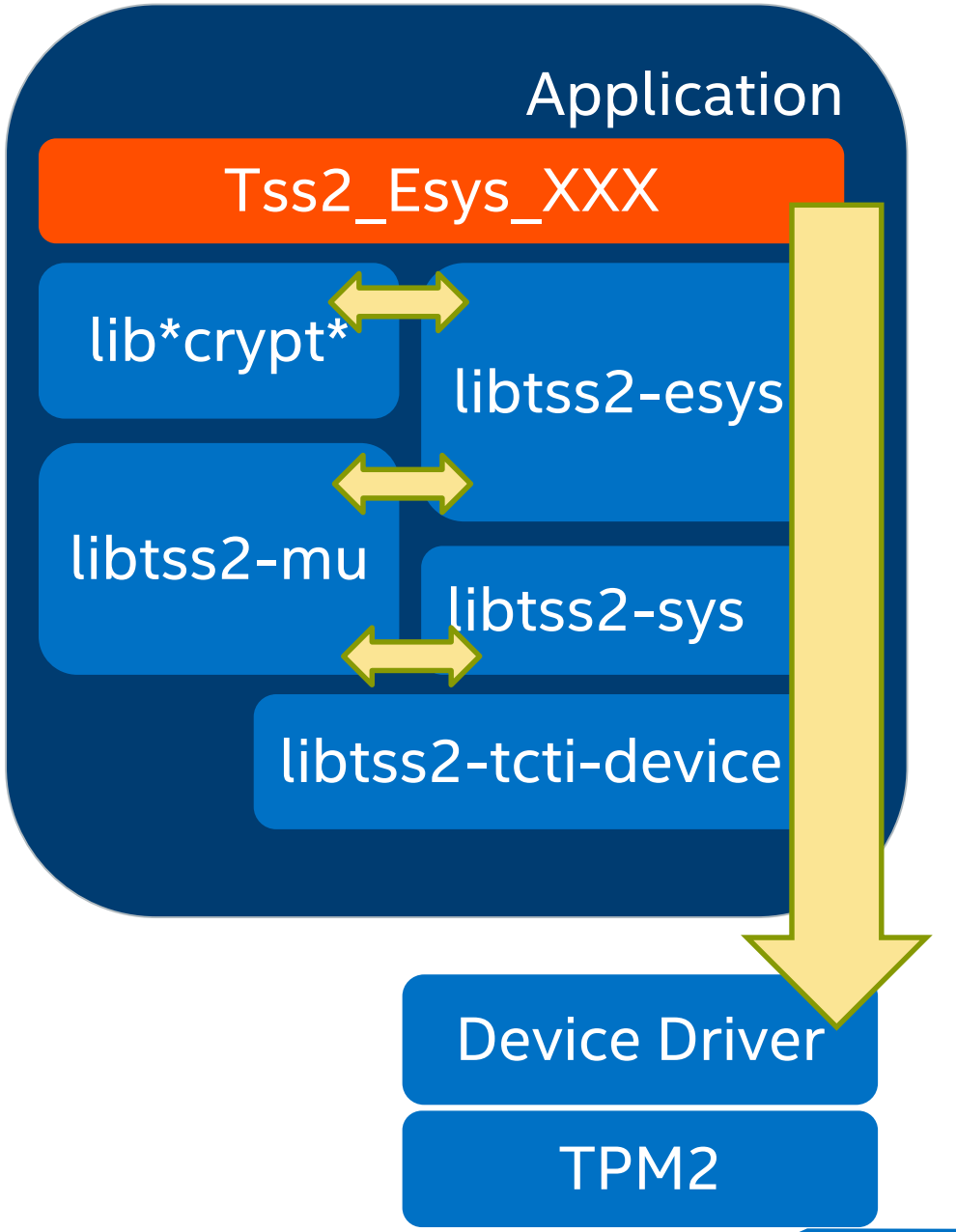
- System API: libtss2-sys
  - Transform C types to TPM command buffer
  - One-to-one mapping to TPM commands
  - Suitable for firmware / embedded applications
- Type Marshaling: libtss2-mu
  - Transform TPM types from C to wire format & back
- TPM2 Command Transmission Interface
  - Abstraction to hide details of IPC mechanism
  - libtss2-tcti-(device|mssim|tbs)



TSS2 APPLICATION

Enhanced System API: libtss2-esys

- **Suitable for general C applications**
- Builds on top of lower-level tss2-* libs
- Expose all TPM2 functions + utilty functions
  - HMAC calculations for HMAC session
  - Crypto for encrypted session
  - Maintain state for authorizations
- Adds dependency on crypto library
  - Current implementation supports
    - libgcrypt
    - openssl



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 & FlushContnext
  - 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)

TPM2 SOFTWARE STACK (TSS2)

OSS Implementation, Community and Adoption

FROM PROTOTYPE TO OSS PROJECT

Stability & Reliability

- Eliminate liabilities / high priority technical debt
  - Make it debuggable
  - Use 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

TPM2-SOFTWARE GITHUB ORG / PROJECT

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
- Community
  - Maintainers from: Intel, Fraunhofer SIT, RedHat
  - Contributions from: Infineon, Facebook, Alibaba, RedHat, GE, Suse, Debian
- New Projects
  - PKCS#11 module, UEFI TCTI, cryptsetup integration, RC decoding library & spec

DOWNSTREAM ADOPTION

Support and Usage in OpenEmbedded, RHEL & Suse

- Packaging for distros
  - RHEL, Suse, Debian, Ubuntu
  - 2.0 TSS2 release *should* make it into RHEL 8, missed SLES 15 ☹
  - Clevis supporting TPM2 module [4]
- StrongSwan VPN
  - Uses TPM2 / TSS2 for key protection
- OpenEmbedded upstreaming effort underway
  - Maintained as part of meta-measured
  - Planning effort to upstream into OE proper: reduce duplication

CHANGELOG

Major milestones & developments

- 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
- Windows support for core libraries / APIs
  - TCTI for communication with TBS: libtss2-tcti-tbs
  - 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
  - Protected crypto 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
- Often times 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: EXAMPLE

Sign data with TPM2 key / verify signature with OpenSSL

- Refresh example from Davide Guerri @ FOSDEM 2017 [5]
- Create primary key in storage hierarchy
  - tpm2_createprimary --hierarchy o --out-context pri.ctx
- Create subkey for signing
  - tpm2_create --context-parent pri.ctx --pubfile sub.pub --privfile sub.priv
- Load subkey
  - tpm2_load --context-parent file:sub.priv --pubfile sub.pub --privfile sub.priv --out-context sub.ctx
- Calculate hash
  - openssl dgst -sha1 -binary -out hash.bin msg.txt
- Sign the hash
  - tpm2_sign --key-context file:sub.ctx --format plain --digest hash.bin --sig hash.plain
- Create OpenSSL compatible DER encoded public key
  - tpm2_readpublic -c “file:sub.ctx” --format der –out-file sub-pub.der
- Verify the signature
  - openssl dgst -verify sub-pub.der -keyform der –sha1 -signature hash.plain msg.txt

TPM2-ENGINE: EXAMPLE

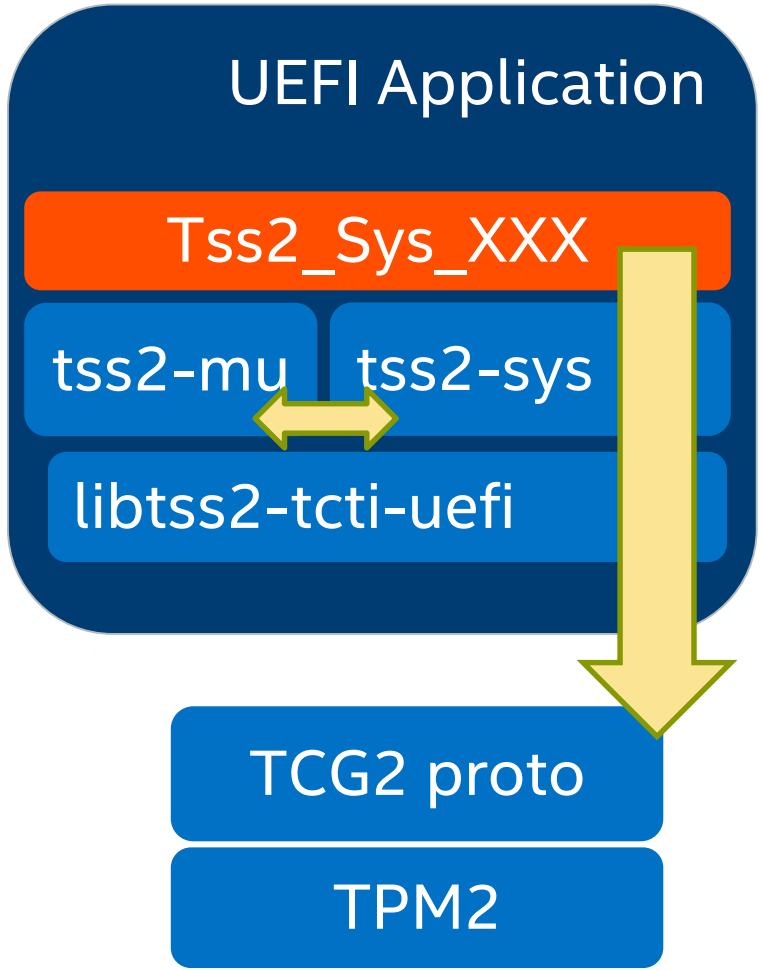
Sign data with OpenSSL (using TPM2 engine) / verify signature with OpenSSL

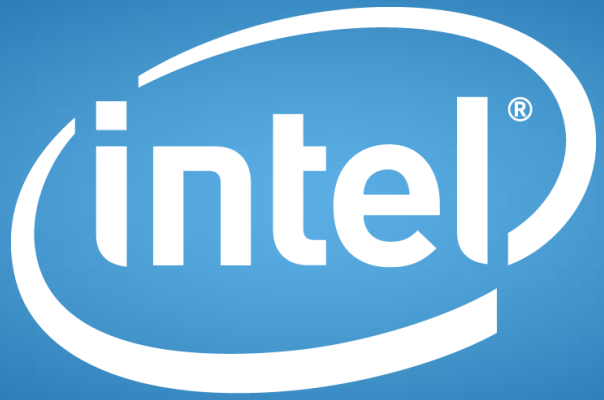
- Same use-case as previous example using tpm2-tools
- Create an RSA key
  - tpm2tss-genkey -a rsa -s ${KEY_SIZE} ${KEY_FILE}
- Export public key in PEM format
  - openssl rsa -engine tpm2tss -inform engine -in key.bin -pubout -outform pem -out key.pem
- Hash the document
  - openssl dgst –sha256 –out hash.txt message.txt
- Sign the hash
  - openssl pkeyutl -engine tpm2tss -keyform engine -inkey key.bin -sign -in hash.txt -out sig.bin
- Verify the signature
  - openssl dgst -verify –pubin key.pem -sigfile sign.bin –in hash.txt

TPM2-UEFI

TCTI enabling use of tss2-sys API in UEFI

- TPM2 support in UEFI: TCG2 protocol
  - Query UEFI protocol & PCR bank settings (5 functions)
  - Measure stuff: HashLogExtendEvent
  - Send command buffer: SubmitCommand
- TCTI built on TCG2 UEFI protocol: libtss2-tcti-uefi
- Enables use of all TPM2 commands via tss2-sys API
  - System manufacturing & provisioning
  - Encrypted boot partition with TPM protected keys
- TCTI for TIS interface for non-UEFI firmwares possible
- Example UEFI applications in source tree
  - https://github.com/flihp/tpm2-uefi





Software

REFERENCES

1. Introduction To Trusted Computing: http://opensecuritytraining.info/IntroToTrustedComputing.html
2. Trusted Platform Modules: https://www.theiet.org/resources/books/computing/tpmwhy.cfm
3. How To Design A Good API and Why it Matters: https://www.youtube.com/watch?v=heh4OeB9A-c
4. Clevis TPM2: https://blog.dowhile0.org/2017/10/18/automatic-luks-volumes-unlocking-using-a-tpm2-chip/
5. FOSDEM TPM2-TOOLS: https://archive.fosdem.org/2017/schedule/event/tpm2/