Trusted Computing Use Cases and the TCG Software Stack (TSS 2.0)

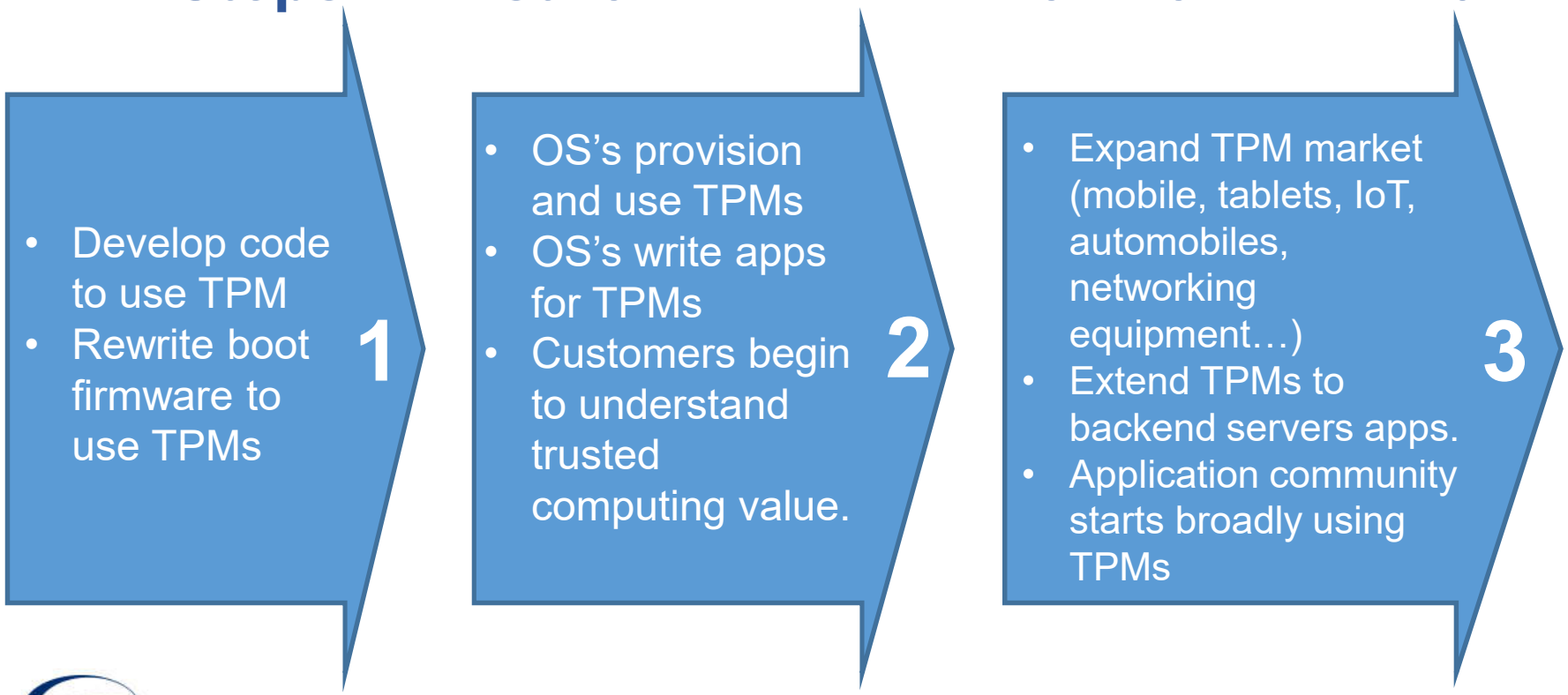
Lee Wilson
TSS WG Chairman
OnBoard Security
November 20, 2017

Trusted Computing:

Where Are We At?

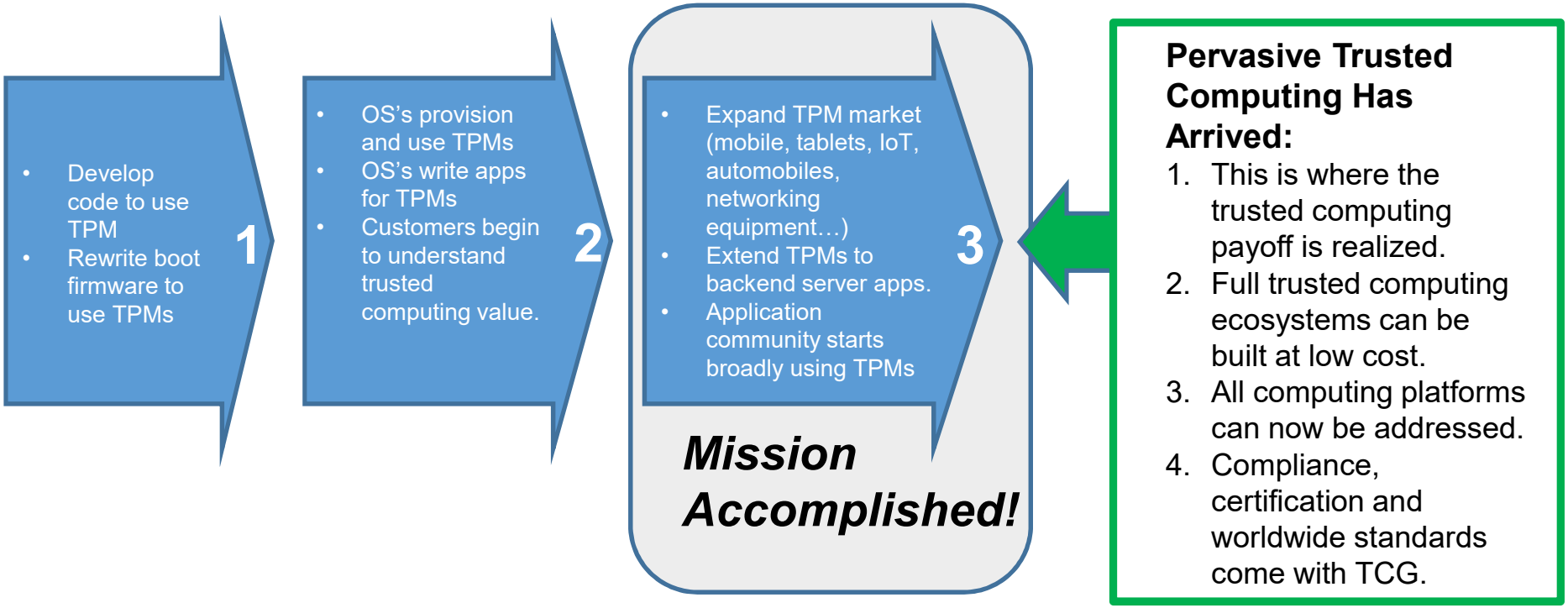
(From the Perspective of Deploying Compelling, Marketable Solutions)

Core Trusted Computing Has Evolved In Three Steps – First for TPM 1.2 – Now for TPM 2.0



TRUSTED COMPUTING GROUP®

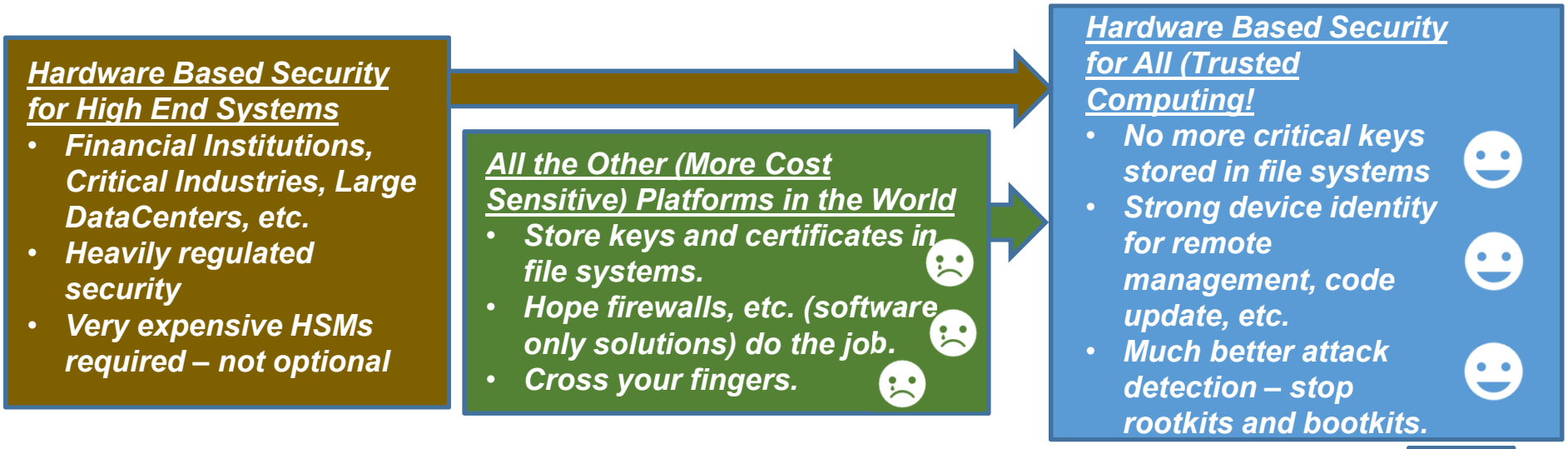
Achieving Trusted Computing's Full Promise with an Ecosystem of Solutions



TRUSTED COMPUTING GROUP®

A Major Announcement *(Drum roll, etc.)*

The Two Worlds of Security Have Now Converged

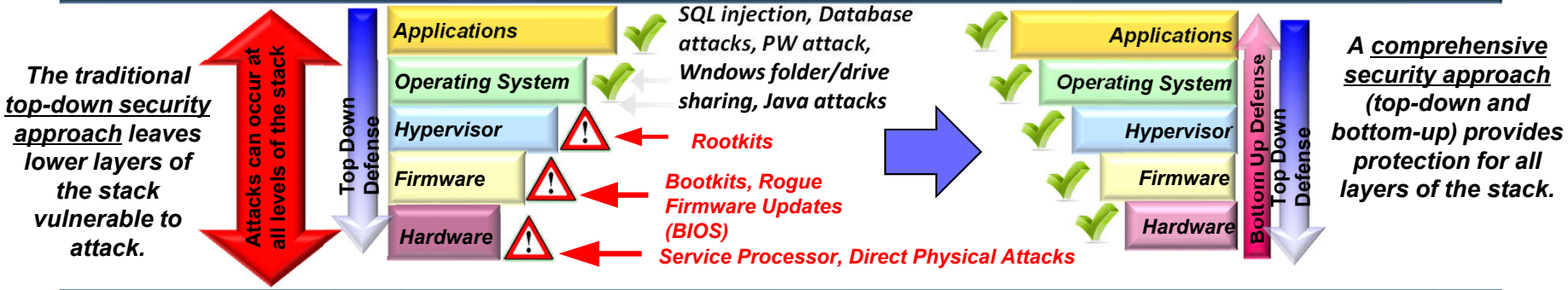


Netting it out: PLEASE get to work on adopting the trusted computing security model!

TRUSTED COMPUTING GROUP®

Security Holes Missed by Software-Only Security – Problem Solved by Trusted Computing

Antivirus programs, Firewalls, Compliance Management, etc. Provide "Top Down" Security.



Trusted Computing Hardware/Firmware/Software Required for Bottom Up Defense

- TPM v2.0/TSS v2.0 provide support infrastructure for trusted computing
- Trusted computing-enabled firmware building a "transitive trust chain" and establishing systems measurements off of a CRTM (Core Root of Trust for Measurement) launch bottom-up security.
- OS'es, RTOS'es,firmware, trusted applications… seal secrets designed to protect overall platform security to the TPM PCRs (Programmable Configuration Registers) and use the TPM as an HSM where appropriate.

TRUSTED COMPUTING GROUP®

Major High Level Use Cases

Building Solutions with Trusted Computing

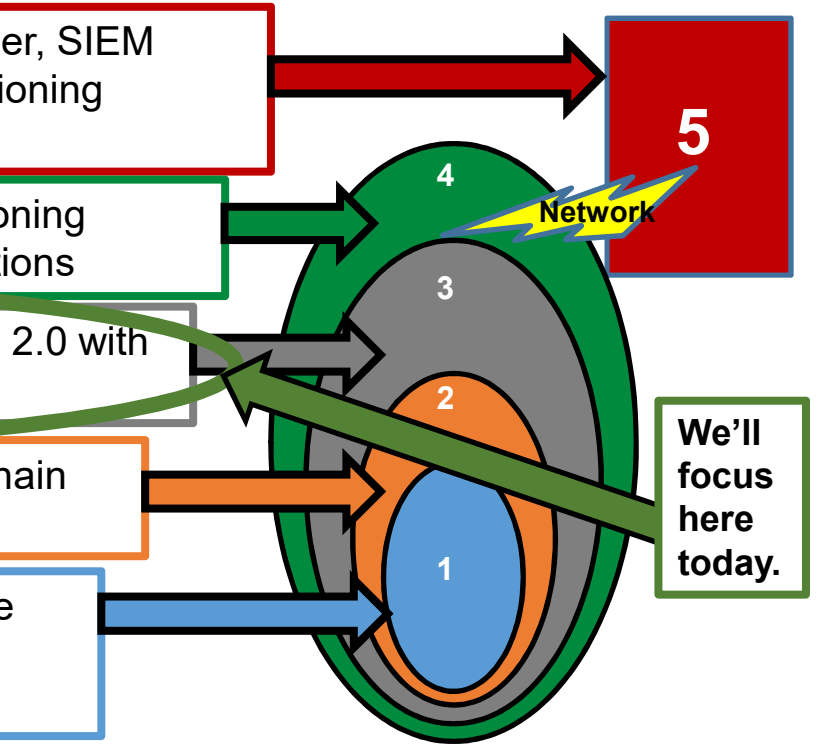
Layer 5 (Backend Server) Attestation Challenger, SIEM tools, Code Update, Endpoint Managers, Provisioning Manager, Key Management Applications

Layer 4 (Applications) Attestation Agent, Provisioning Agent, SSH/SSL/IPSEC. enabled. Signing applications

Layer 3 (Middleware/Services - Extended OS TSS 2.0 with Resource Manager, PKCS#11

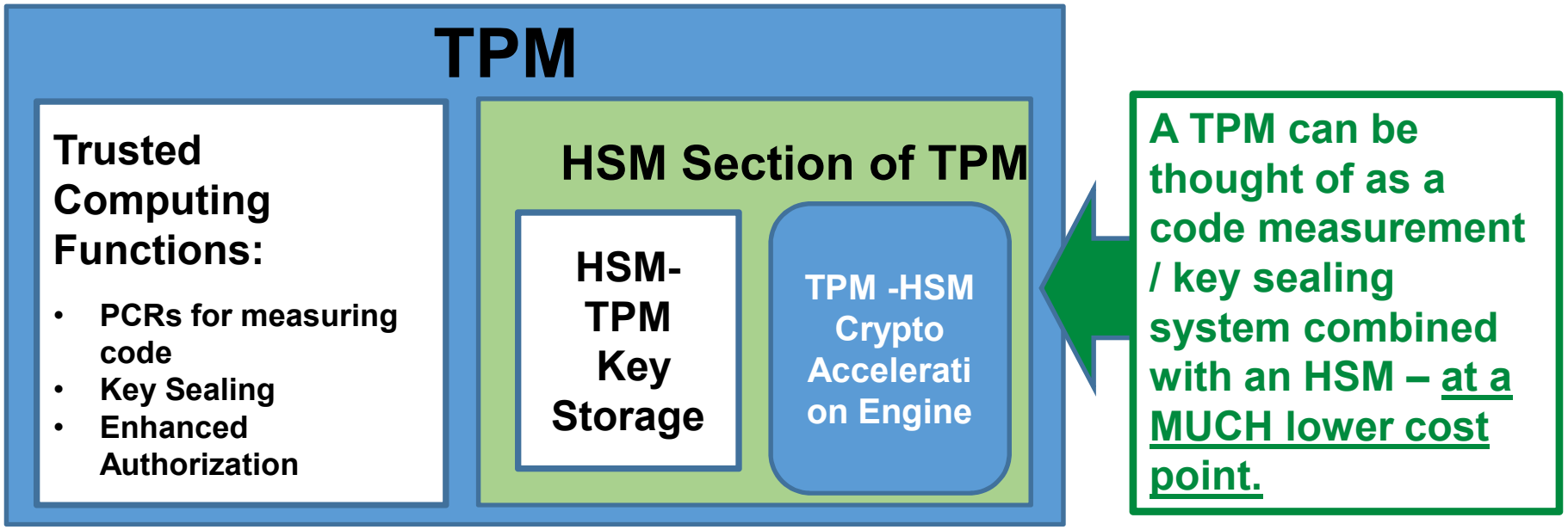
Layer 2 (Core OS) Code extending the transitive trust chain using OS-owned keys, Drive encryption functions

Layer 1 (Firmware/Hardware) CRTM to start the transitive trust chain, Boot firmware to extend beyond the CRTM, initialize the TPM, etc.



TRUSTED COMPUTING GROUP®

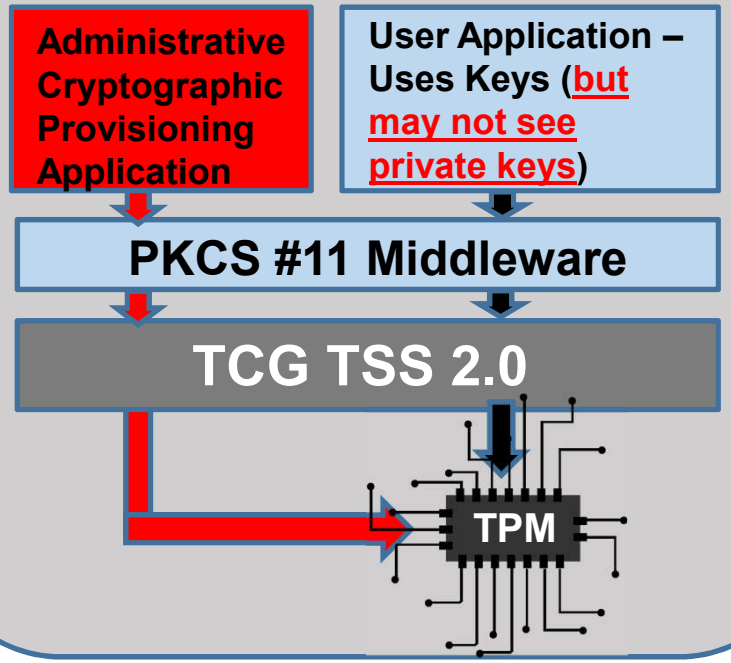
A TPM Is Best Described In Two Pieces



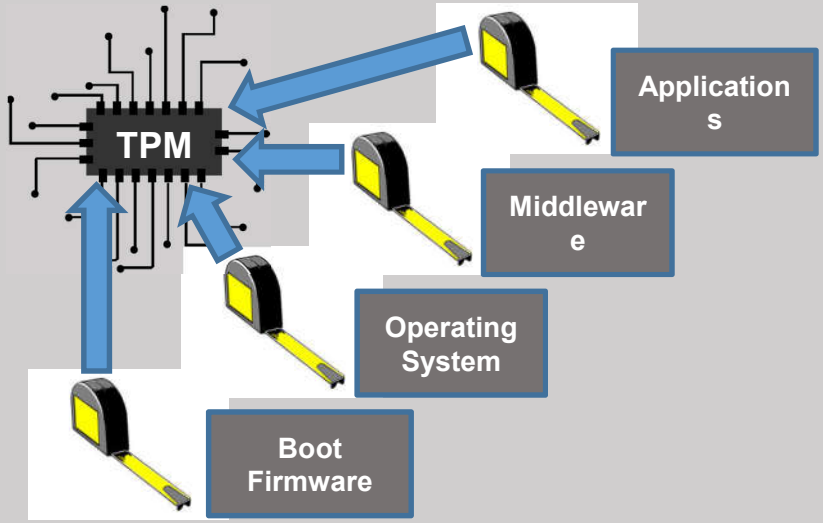
TRUSTED COMPUTING GROUP®

Major Trusted Computing Use Cases - 1&2

Use Case 1: HSM-Style Key Store and Use – But Using a TPM



Use Case 2: Measure your software – Seal keys, detect attacks, endpoint management…

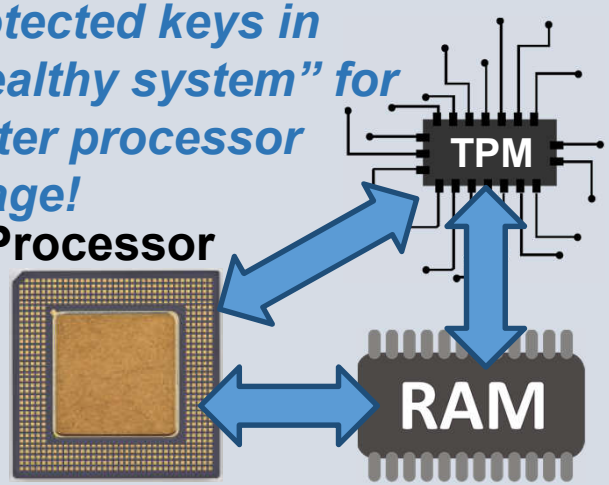


TRUSTED COMPUTING GROUP®

Major Trusted Computing Use Cases - 3&4

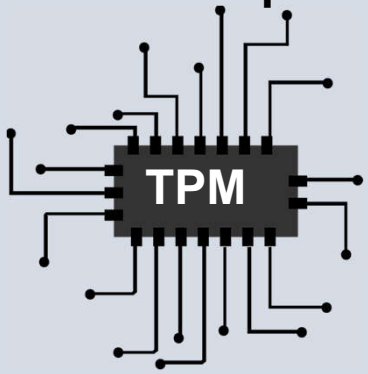
Use Case 3: Trusted Computing Specific Key Use Model: Key

Unseal TPM-protected keys in "healthy system" for faster processor usage!



Use Case 4: "Strong Device Identity and Authentication" Using TPMs

Trusted Computing Enabled Platform (TPM Rigidly Attached As Required)

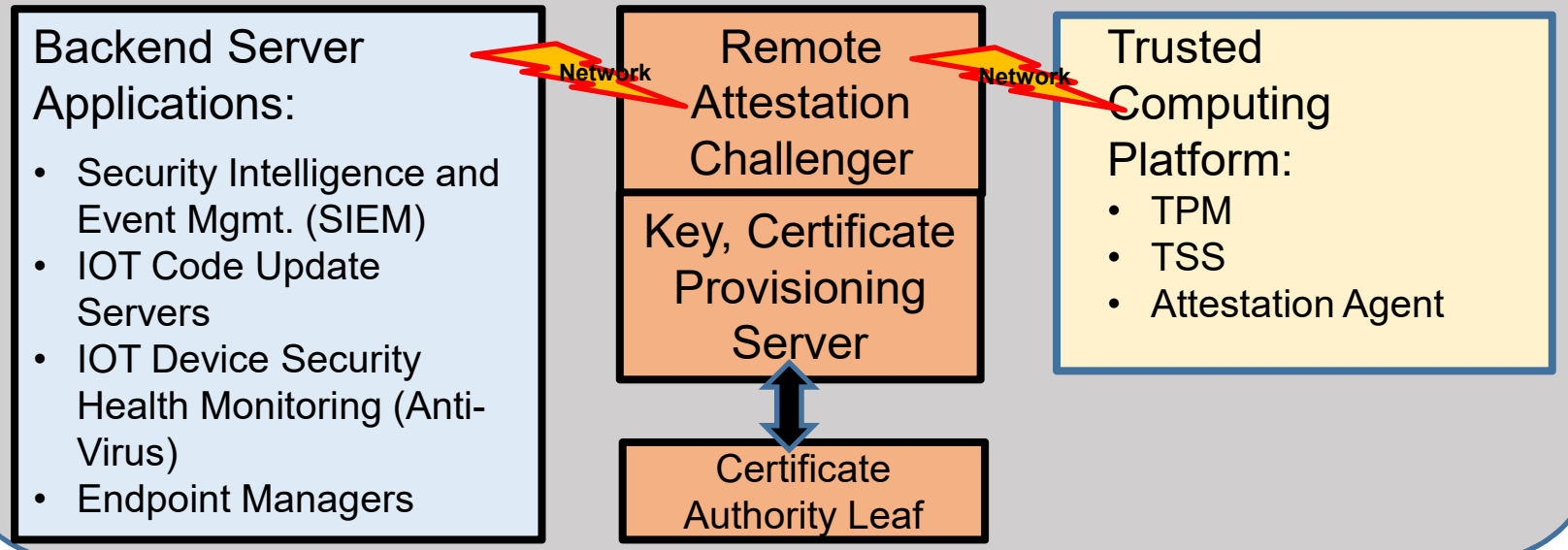


Inexpensive TPMs are required to be permanently melded to their platforms

TRUSTED COMPUTING GROUP®

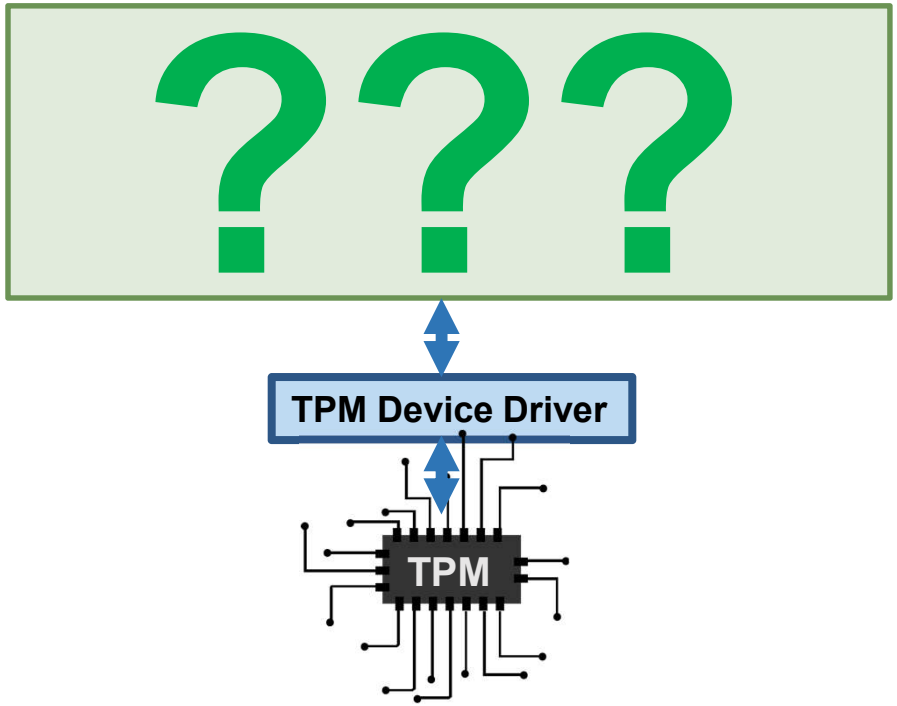
Major Trusted Computing Use Cases - 5

Use Case 5: Trusted Computing Ecosystem Health Monitoring and Management



How Do My Applications Actually Use TPM 2.0 to Achieve These Use Cases?

What Core Software Do You Need to Build Secure Solutions with TPM 2.0?



TRUSTED COMPUTING GROUP®

Why Is TCG TSS 2.0 Needed?

- What does the TCG Software Stack 2.0 (TCG TSS 2.0) do for your programmers:
  - Handles the marshaling/unmarshaling needed when you communicate with a TPM – handles multiple TPM applications.
  - Provides synchronous and asynchronous function call models for communicating with the TPM.
  - Encrypts the data stream from the software to the TPM stopping side-channel (hardware probing) attacks (EAL 4++).
  - Simplifies context and session management needed when applications work with TPMs.,
  - Provides varying levels of abstraction (depending on the TSS layer you use) simplifying the task of using the TPM.
  - Provides “scalable solutions” allowing different code footprints from the smallest IOT device up to server applications.

TRUSTED COMPUTING GROUP®

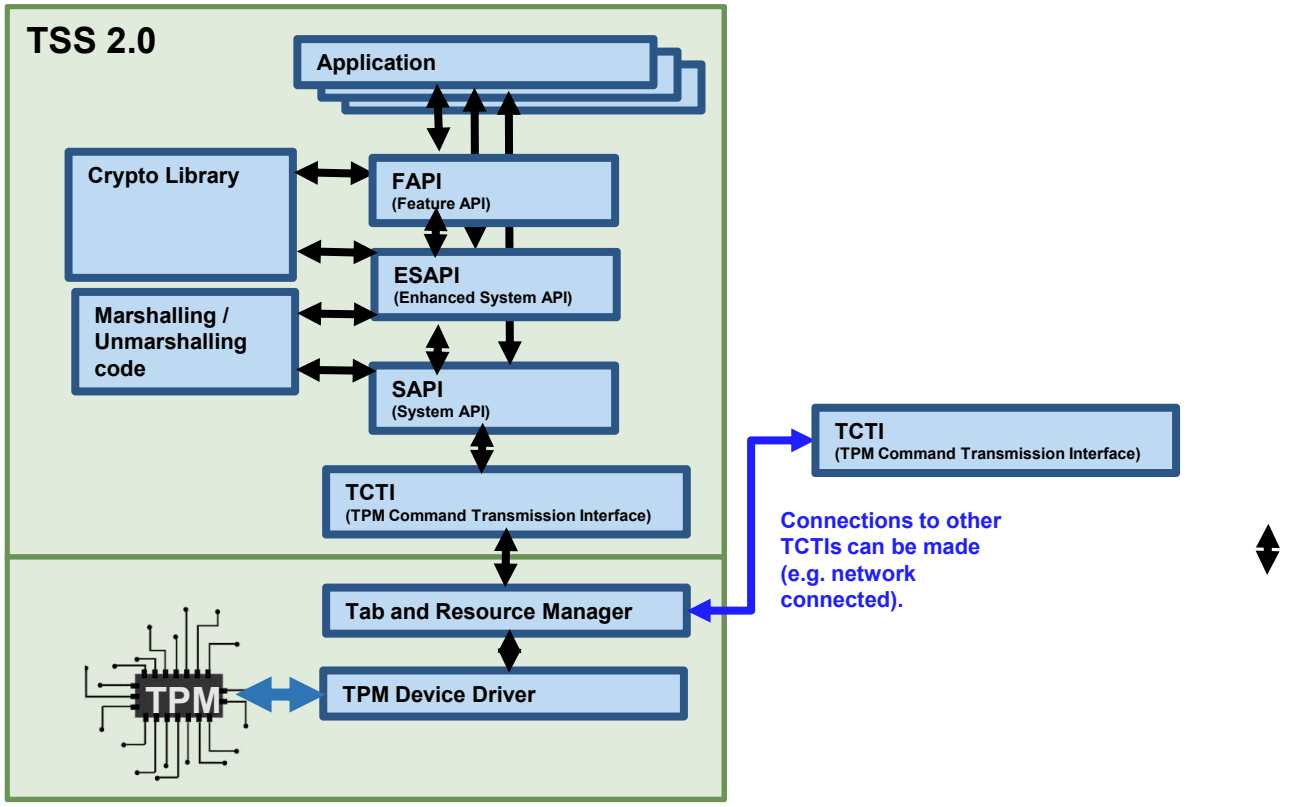
Why TCG TSS 2.0 Specifically (1)?

- Why do you need the TCG TSS 2.0 Specifically:
  - It is a standardized API which will permit applications to use the same programming model cross platform (no need for completely different APIs on each platform).
    - **Note:** Many governments and critical industries will call it out specifically in their RFPs for these reasons.
  - Complies with modern "clean programming" techniques making your code more maintainable and more secure
    - No function overloading ***High Semantic Content!***
    - Strong type checking ***No variadic variables!***
    - High semantic content (Others – including yourself – will be able to read your code, understand it and maintain it over the lengthy product lifecycle we must support.
    - No global variables, etc.
  - Provides both synchronous and asynchronous call support.
  - Easy to write language bindings for TCG TSS 2.0 (implementations are in C99).

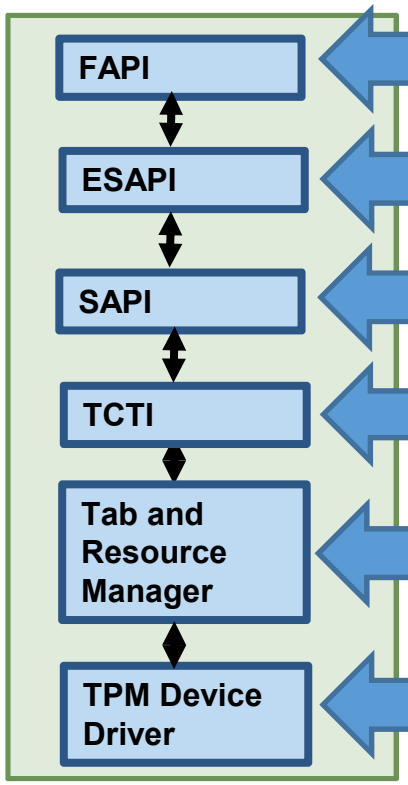
Why TCG TSS 2.0 Specifically (2)?

- Why do you need the TCG TSS 2.0 Specifically (continued):
  - The TSS 2.0 API Compatible with MISRA coding standards (required in embedded and IOT where safety is an issue).
    - Note: The implementation underlying the TSS 2.0 may or may not be MISRA compliant.
  - Developed and scrutinized by the TCG community at large.
  - Strong versioning and revision control insured by the design of the TSS 2.0 API
  - Candidate (under development) application code written to TCG TSS 2.0 will tend to fail at compile time – not run time – stopping errors from reaching the field. If you have to fail "fail early, fail fast." – not at the customer.

The TCG Software Stack (TSS 2.0)



Descriptions of TSS 2.0 Layers



FAPI provides new ease of use not available for TSS 1.2. **It allow programmers to interface to the TPM without having ot be TPM experts.**

ESAPI has easier context management and provides the ability to encrypt the data stream to the TPM **stopping sidechannel attacks (essential to EAL4++)**

SAPI doesn't need a file system or a heap. **It can be integrated with your boot firmware** or used in the smallest IOT devices.

The TCTI is an enormous help to development programmers. It allows you to target TPMs other than the hardware TPM on your platform (eg. soft TPMs)

The Tab and Resource Manager will vary depending on the operating system. It allows multiple applications and the kernel to share TPM resources.

Drivers are available today in Linux, Windows – your particular IOT platform may need a modified or custom driver.

TRUSTED COMPUTING GROUP®

Code Requirements for TSS Layers

SAPI and TCTI

- No file IO
- No crypto
- No heap
- No external library dependency
- Context based state

ESAPI

- Cryptographic function
- No file IO
- Requires heap
- Context based state

FAPI

- File IO
- Requires heap
- Must be able to do retries
- Context based state
- Must support the possibility of reduced application code size by offering static libraries
- Abstracts TPM details from programmers

TAB and Resource Manager

- Power management
- Potentially no file IO – depends on power mgmt.
- No crypto
- Requires heap

TRUSTED COMPUTING GROUP®

Details on the FAPI API

Objectives of the TCG FAPI Specification:

- The TSS 2.0 Feature API is meant to be a very high level API which allows programmers to use the TPM 2.0 without having a deep knowledge of how TPMs work.
  - It is aimed at having commands in it that will allow 80% of the programmers who write a program using the TPM to find everything they want in the specification.
  - The remaining 20% of programmers will have to supplement this set of APIs with the Extended System API (ESAPI) or System API (SAPI).
- The specification is meant to making programming with the TPM as simple as possible. The cognitive load for a new programmer using this API is kept as low as possible.

TRUSTED COMPUTING GROUP®

Details on the FAPI API (cont.)

The following decisions have been made for the FAPI specification:

- A Profile is used by a programmer that makes many of the complicated decisions for the programmer. It decides such things as the default algorithm sets that are used when creating keys, where they are stored and found.
- Key template names have been created for the dozen or so keys that are expected to be used by most programmers
- Key names will be based on path descriptors, much as files are today.
- All entities used by the feature API will be authenticated by use of a policy. (The policy may point to an authorization done using the authorization data, however.) This means that no entity will be created with a NULL policy. It probably also means that bits will be set to disable use of the authorization data in objects.
- All authorizations done using authorization data will use salted HMAC sessions. Decrypt and encrypt sessions will also be used.
- Policy instances and forms are described in an XML representation which may be found in the Policy XML format document.
- PCR log files will be in the format described by the PC Client Specification
- Commands syntax looks like Tss2_<EntityName>_<Command> :
    - Tss2_Key_Sign
    - Tss2_Nv_Write
    - Tss2_Entity_ChangeAuth
    - Tss2_TPM_GetRandom

Details on the FAPI API (cont.)

The following decisions have been made for the FAPI specification (cont.):

•The Feature API doesn't include two other things which are necessary to get it to work, which are expected to be needed, namely:

1. A utility used to create a policy in the correct XML format

   For example:

   ```
   <PolicyAce type="PolicySigned">
      <Name>Company SmartCard</Name>
      <Driver>MySmartCard</Driver>
      <DriverInfo>DN=CompandSmartcard.com</DriverInfo>
      <etc>...</etc>
   </PolicyAce>
   ```

2. Callback functions will be used to obtain decisions from the user and interfaces related to policy commands that require input. The FAPI will read the policy associated with an entity when it is used, create a Policy session to satisfy it, and walk through the command necessary to satisfy the command. It will use the callback functions to determine:
   - Which branches of an OR (or PolicyAuthorize) policy to follow
   - How to obtain passwords or signatures necessary to satisfy the policies.

- The default TPM this will work with is assumed to be the local one, but another one can be specified when a context is created.

Details on FAPI XML Policies

- TPM 2.0 introduces a flexible design for authorizing actions on keys and other TPM objects. The FAPI XML Policies Document describes:
  - Interoperable XML-based scheme for describing policies and interpreting them to authorize TPM actions
  - A standardized interoperable form for policies will allow different participants to author and consume policy expressions. (e.g. An enterprise management utility might author policies for key migration and recovery. These policies may then be consumed by any TPM client library.)
- The FAPI XML Policies Document contains the following sections:
  - An overview of the authorization capabilities of TPM 2.0
  - An XML-based scheme for expressing policies
  - A proposed simple “normal form” for policies
  - An algorithm for policy evaluation
- The main text of this document uses an abbreviated XML schema. The complete XML schema is included in an appendix to this document.

The TCG TSS 2.0 Is Now Public – Full Scale Deployment Can Begin

- The TCG TSS 2.0 Specifications can be seen on the TCG website at: https://trustedcomputinggroup.org/specifications-public-review/
- Implementations are available:
  - Intel has provided an open source version that includes the TSS 2.0 components up to the SAPI layer.
  - Commercial implementations of the full TCG TSS 2.0 stack are available (the FAPI layer is undergoing final revisions – implementations will follow on publication).
- When you look at TSS 2.0 offerings, please be sure the TSS 2.0 is TCG compliant (others have used the term TSS 2.0 that do not conform to the TCG specifications).

TRUSTED COMPUTING GROUP®