Mobile trusted computing

Chris Mitchell
Royal Holloway, University of London
c.mitchell@rhul.ac.uk

Contents I

1. **Trusted computing: The story so far**:
   - The Trusted Computing Group (TCG);
   - Fundamentals of trusted computing.
2. **Mobile security**:
   - The need for a trusted mobile platform;
   - The current status of mobile security;
   - Need for standardisation.

Contents II

3. **A Trusted Mobile Platform (TMP)**:
   - Main standardisation bodies;
   - Building a TCG TMP;
   - Use case definition;
   - Requirements analysis (example: OMA DRM v2.0 use case).
4. **Elements of a TCG TMP**:
   - Stakeholders;
   - A TCG TMP;
   - Secure boot;
   - Maintaining integrity after boot;
   - The MRTM and MLTM;
   - Software isolation.

1. Trusted computing: The story so far

- Objectives:
  - Give a brief overview of the history of trusted computing technology;
  - Review the main technological objectives and components of trusted computing.

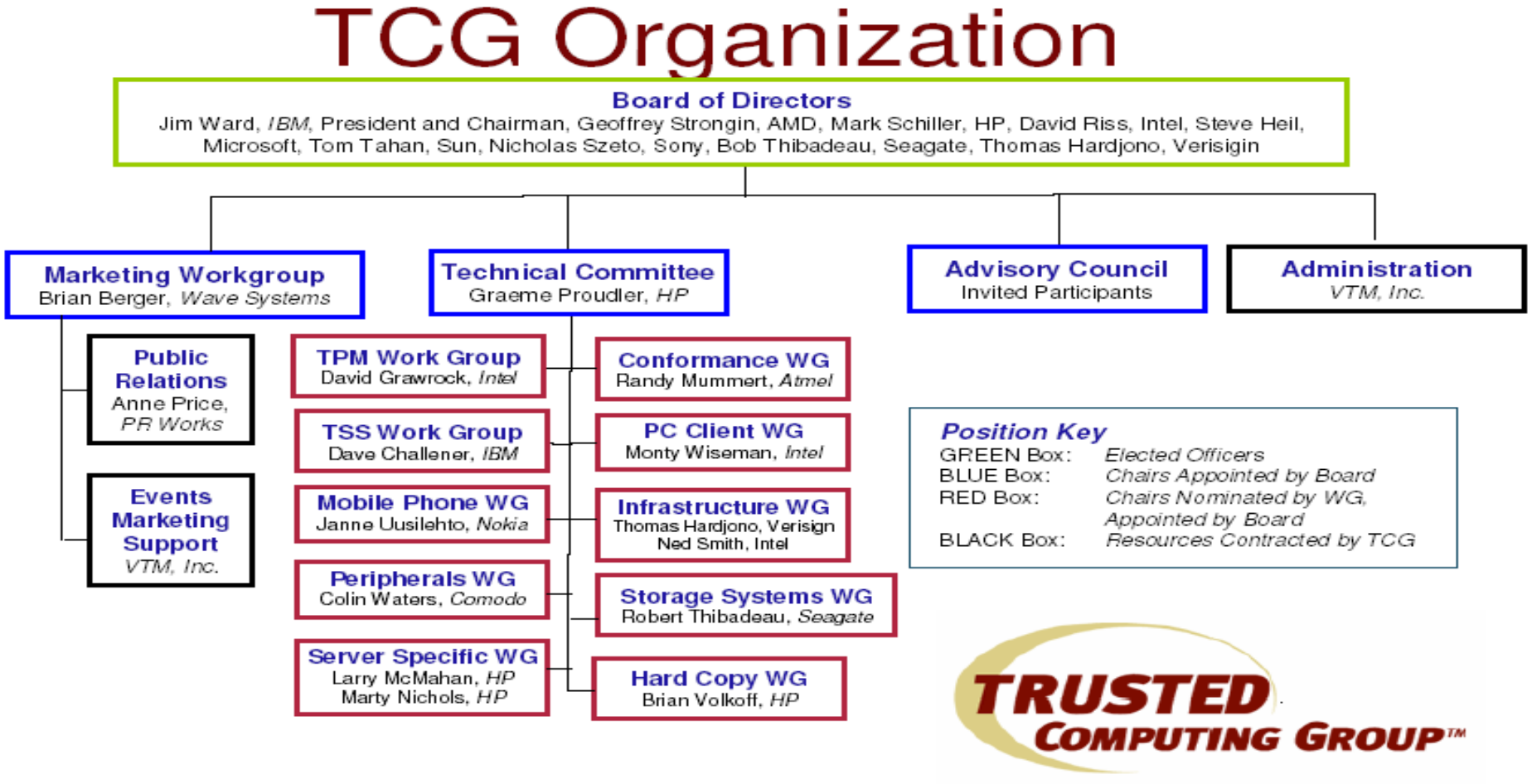
Trusted computing – history I
The TCPA

- **TCPA** (Trusted Computing Platform Alliance): an industry working group.
- Focus: enhancing trust and security in computing platforms.
- Original alliance of promoter companies (HP, IBM, Intel and Microsoft). Founded in 1999.
- Initial draft standard unveiled: late 1999.
- Invitation then extended to other companies to join the alliance.
- TCPA released first specifications in early 2001, defining a fundamental component of a trusted platform, namely the Trusted Platform Module (TPM).
- A TPM is typically implemented as a chip mounted on a PC motherboard, and provides a foundation for all trusted functionality on the PC (in combination with the BIOS).
- By 2002: the TCPA had over 150 member companies.

Trusted computing – history II
The TCG

- **TCG** (Trusted Computing Group): announced April 8, 2003.
- TCPA recognised TCG as its successor organisation for the development of trusted computing specifications.
- The TCG adopted the specifications of the TCPA.
- Aim of the TCG:
  - To extend the specifications for multiple platform types;
  - To complete software interface specifications to facilitate application development and interoperability;
  - To ensure backward compatibility.

Trusted computing – history III
The TCG



Trusted computing – history IV
Operating system architectures

- Operating system components:
  - **Microsoft**: Palladium »
    Next Generation Secure Computing Base (NGSCB) »
    Hyper-V Server 2008;
  - **Academia/open source community**:
    - Terra;
    - Perseus;
    - Open Trusted Computing architecture;
    - European Multilaterally Secure Computing Base.
- Processor extensions:
  - **Intel**: LaGrande Technology (LT) »
    Trusted eXecution Technology (TXT);
  - **AMD**: AMD-V.

Trusted computing – history V
Prior research

- Hardened processor architectures:
  - AEGIS;
  - XOM.
- A secure boot process:
  - AEGIS.

TCG specifications

- The TCG publishes its completed specifications freely on the web.
- Specifications under development are not freely available – they are for ‘members only’.
- However, there is a liaison programme for academic institutions, which gives access to documents (under NDA) without charge.
- The v1.2 TPM specifications (the current version) have recently been adopted as an international standard: ISO/IEC 11889 parts 1-4 (with the title *Information technology - Trusted Platform Module*) – the scheduled publication date is 5/5/09!

TCG trusted computing: Basic components and services

- **Integrity measurement –** a cryptographic hash of a platform component (i.e. software executing on the platform);
- **Authenticated boot –** process by which a platform's state (the sum of its components) is reliably measured and stored;
- **Sealed storage –** process of storing data on a platform in such a way that the data can only be retrieved if the platform is in a particular state;
- **Attestation –** process of reliably reporting the platform's current state;
- **Isolated execution –** enables the unhindered execution of software.

Trusted computing
Platform components

- The TCG has specified platform components required in order to implement:
  - Integrity measurement;
  - Authenticated boot;
  - Sealed storage;
  - Attestation.
- Of fundamental importance are the three **Roots of trust**: “components that must be trusted if the platform is to be trusted”:
  - Root of trust for measurement (RTM);
  - Root of trust for storage (RTS);
  - Root of trust for reporting (RTR).

Roots of trust I

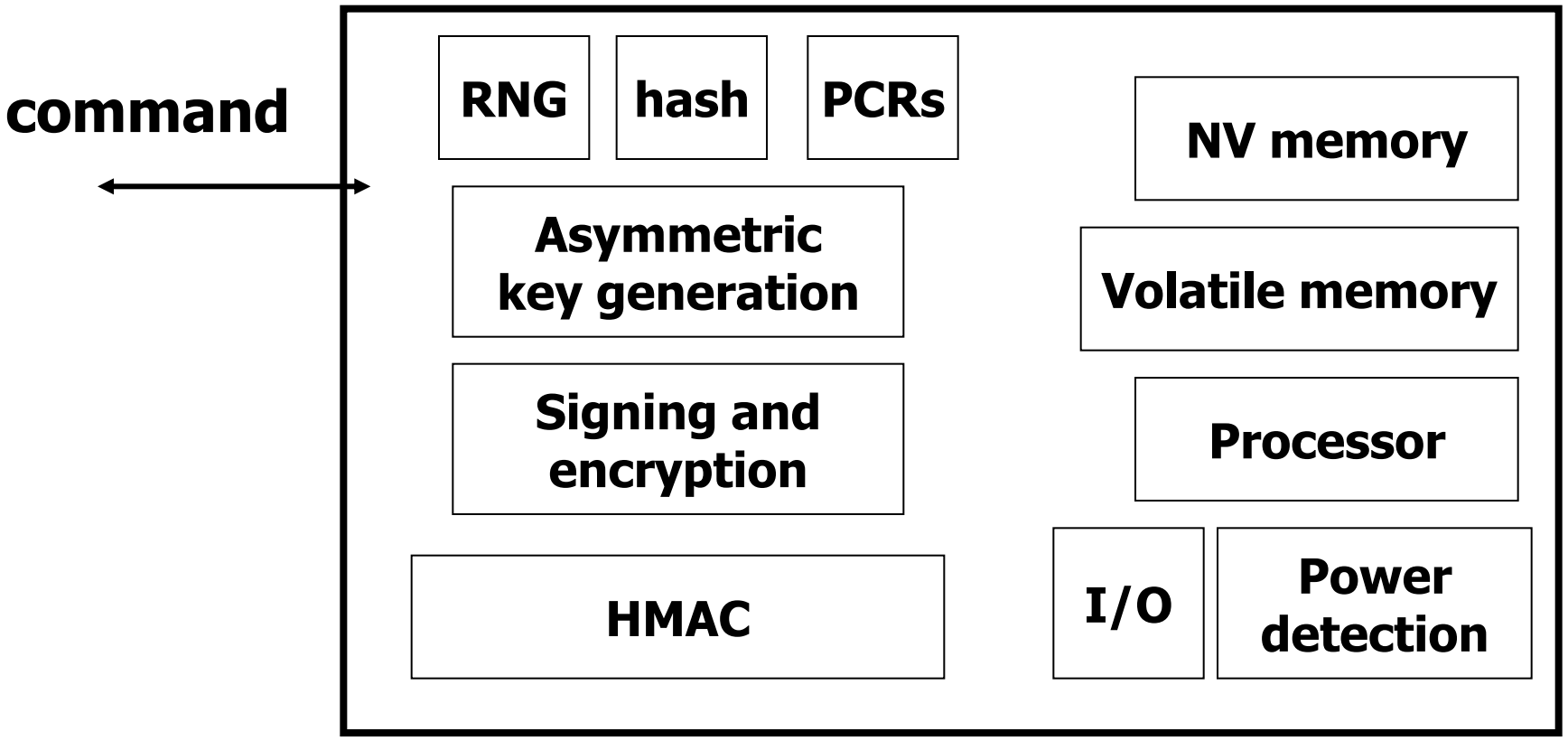
- **The RTM**:
  - The RTM is a computing engine which accurately generates at least one integrity measurement event representing a software component running on the platform;
  - For the foreseeable future, it is envisaged that the RTM will be integrated into the normal computing engine of the platform, where the provision of additional BIOS boot block or BIOS instructions (the **Core RTM** or **CRTM**) cause the main platform processor to function as the RTM.

Roots of trust II

- The **RTS** is a collection of capabilities which must be trusted if storage of data inside a platform is to be trusted:
  - Storing accurate summary of integrity measurements (platform state information);
  - Integrity and confidentiality protection of data;
  - Sealing.
- The **RTR** is a collection of capabilities that must be trusted if reports of integrity measurements which represent the platform state are to be trusted.
- The RTS and RTR constitute the minimum functionality that should be provided by a **Trusted Platform Module (TPM)** – which is typically implemented as a hardware chip bound to the platform.

The TPM

A TPM is typically implemented as a chip mounted on the motherboard of its host platform.



Cryptographic aside

- The cryptographic functions are fixed (‘hard coded’) in the v1.2 TPM specifications.
- This has recently caused major problems, with the discovery of weaknesses in the design of SHA-1, since SHA-1 is one of the functions built into the v1.2 TPM specifications.
- SHA-1 now looks set to be phased out by NIST over the next few years.
- There will thus be a need for a new TPM specification in the next couple of years (TPM.next), which looks likely to use crypto in a more flexible way (e.g. with algorithm identifiers, as in X.509, instead of fixed algorithms).

The TSS

- The **TCG Software Stack (TSS)** is software (running on the host platform) which supports use of the TPM.
- The TSS architecture consists of a number of software modules, which provide resources to support access to the TPM:
  - the TPM Device Driver;
  - TPM Core Services;
  - TPM Service Provider.

Entities in the TCG model

- The **TPM owner** is in complete control of a trusted platform's (TP's) TPM:
  - Some commands are *Owner authorised* (can only be executed by owner).
- **TPM user** (may be different to TPM owner).
- **Challenger** (wishing to verify platform state).
- **Protected object owner** (owner of data/software on a platform, which may be distinct from TPM owner and TPM user).
- **Intermediaries –** used to support migration.

Trusted Third Parties

- The TCG system relies on a number of Trusted Third Parties (TTPs), typically to issue signed certificates asserting certain properties of hardware or software.
- We refer to these as **Certification Entities**.
- A Trusted Platform should be shipped with several certificates created by these entities.

Certification entities I

- A **Trusted Platform Module Entity (TPME)** asserts that the TPM is genuine by signing an endorsement credential containing the public endorsement key for that TPM. The TPME is likely to be the TPM manufacturer.
- A **Conformance Entity (CE)** signs a conformance credential to assert that the design and implementation of the TPM and trusted building blocks (TBBs) in a trusted platform meet established evaluation guidelines.
- A **Platform Entity (PE)** signs a platform credential to assert that a particular platform conforms to a TP design, as described in conformance credentials, and that the platform's TPM is genuine.
- In the future, it is planned that every trusted platform will be shipped with an endorsement credential, conformance credential(s), and a platform credential.

Certification entities II

- Two other certification entity types are defined:
  - A **Validation Entity (VE)** certifies integrity measurements, i.e. measured values and measurement digests, which correspond to correctly functioning or trustworthy platform components, for example embedded data or program code, to create a validation certificate.
  - A **Privacy-CA (P-CA)** creates a certificate to assert that an identity (and an attestation identity public key) belong to a trusted platform.

TCG keys

- To perform the tasks expected of it, a TPM uses a range of different types of key, including secret keys and key pairs for asymmetric algorithms.
- These keys include:
  - **Endorsement Key (EK)**, an asymmetric encryption key pair, unique per TPM, and typically generated at time of manufacture;
  - **Attestation Identity Keys (AIKs)**, i.e. signature key pairs, generated by the TPM during use – a TPM may have many;
  - **Storage Root Key (SRK)**, an asymmetric encryption key pair used to support secure storage of data external to the TPM.

Endorsement Key Pair (EK)

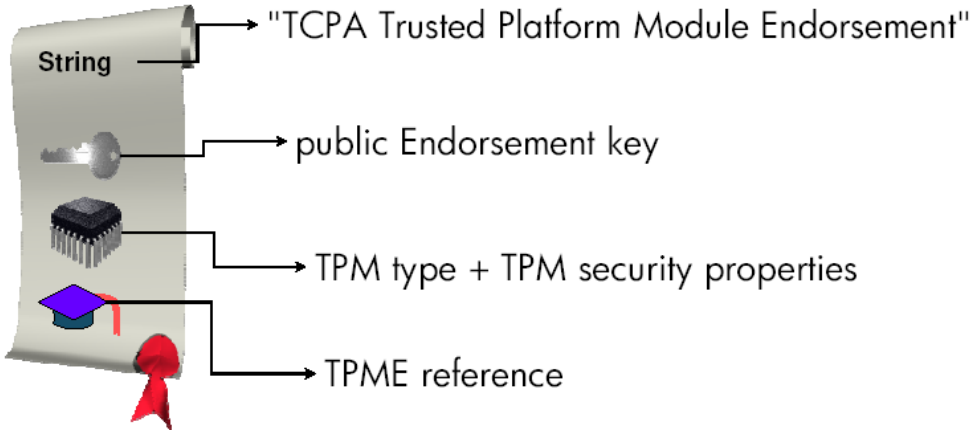
- It is a fundamental requirement that:
  - Each TPM has an endorsement key pair stored in it;
  - The public part of the endorsement key pair is certified by the TPME (e.g. the TPM manufacturer) in the form of the endorsement credential.
- The private part of the EK is used by a TPM to prove that it is a genuine TPM. It is never used for signing.
- It is only ever used in two scenarios:
  - To take ownership of a TPM;
  - To get a public key certificate for a platform attestation identity public key (a 'platform identity').

Platform Credentials

- Prior to use, a trusted platform (and the TPM within the platform) are equipped with a set of signed certificates – generated by some of the TTPs referred to earlier.
- These certificates bind the public part of the EK to the platform, and also attest to properties of the platform.
- We refer to these certificates as the Platform Credentials.

Credentials I

- An **Endorsement credential**:
  - certifies that a public encryption key (the public endorsement key) belongs to a genuine TPM;
  - is signed by a Trusted Platform Management Entity.

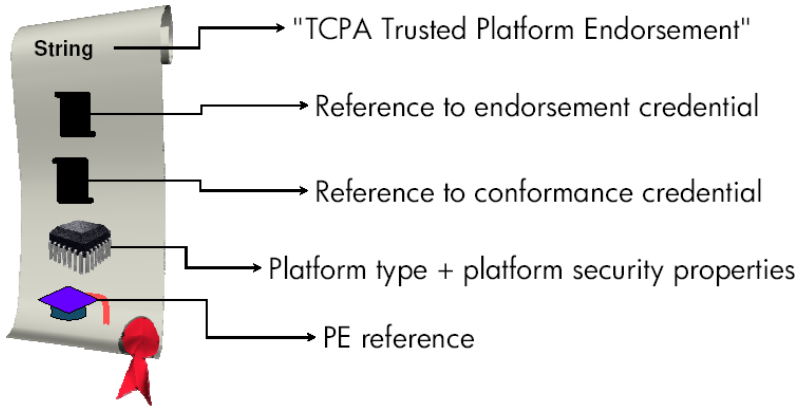


Credentials II

- A **Conformance credential** is:
  - a document that vouches that the design and implementation of the TPM and the trusted building blocks (TBBs) within a trusted platform meet established evaluation guidelines;
  - signed by a Conformance Entity.

Credentials III

- A **Platform credential**:
  - is a document that proves that a TPM has been correctly incorporated into a design which conforms to the specifications;
  - proves the trusted platform is genuine;
  - is signed by a Platform Entity.



Attestation Identity Key Pairs (AIKs)

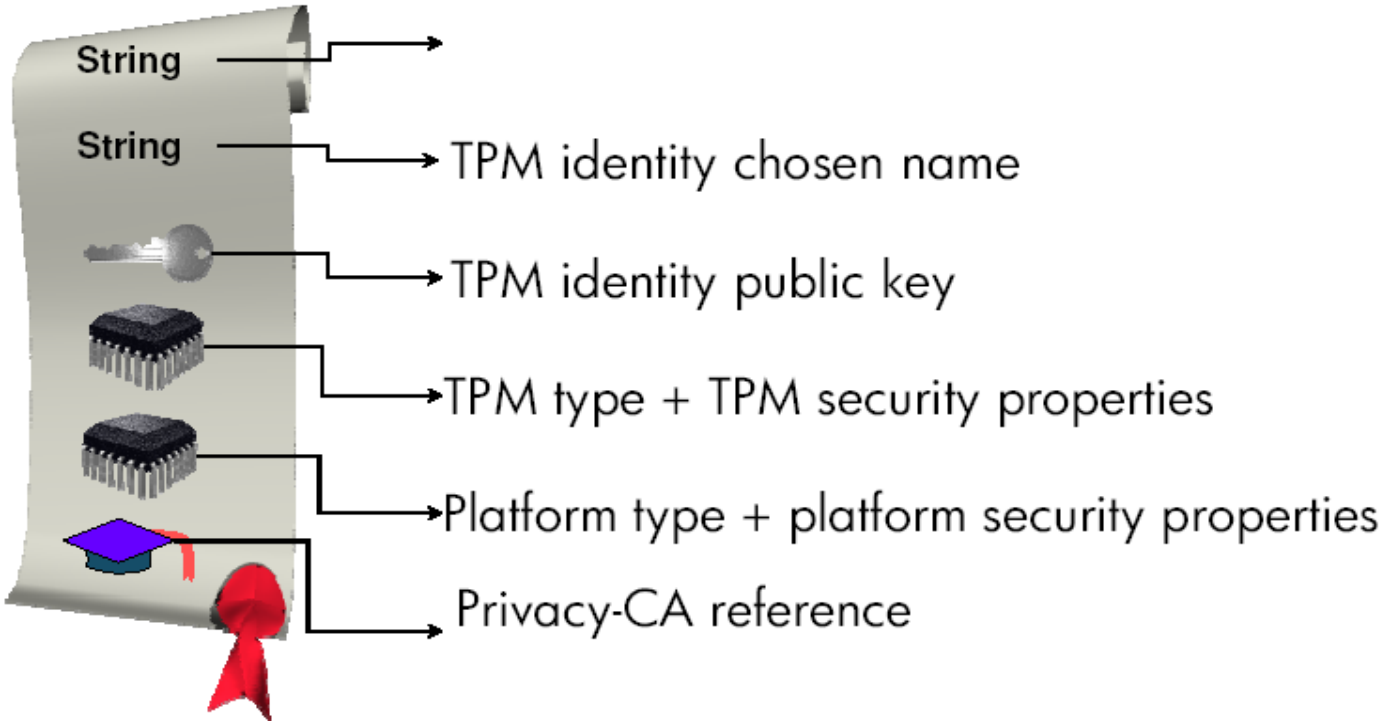
- These signature key pairs are used by a TPM to attest to platform properties to external entities.
- Used by a ‘challenger’ of the platform to verify that a TPM is indeed genuine, without identifying a specific TPM.
- A special trusted third party called a Privacy-Certification Authority (P-CA) supports the use of AIKs.

Generation of AIKs

- TPM chooses a new AIK pair, an ‘identity’, and a P-CA which will be requested to attest to this new identity.
- The TPM signs the public key, the chosen identity, and the identifier of the chosen P-CA, using the newly generated AIK private key.
- The public key, identity, signature and TPM credentials are all encrypted using the P-CA public key and sent to the P-CA.
- The P-CA decrypts the data, verifies the credentials and the signature.
- The P-CA generates the Platform Identity Certificate, a statement that the AIK and the identity being to a genuine trusted platform with the specified properties.

Platform identity certificate

- A Platform identity certificate (as generated by a P-CA) has the following content:



Sending the platform identity certificate to the TPM

- The P-CA generates a random secret encryption key.
- The platform identity certificate is encrypted using this secret key.
- The secret key is encrypted using the TPM's public EK.
- The encrypted certificate and key are then sent back to the requester, thus ensuring that only the appropriate TPM can access the certificate.

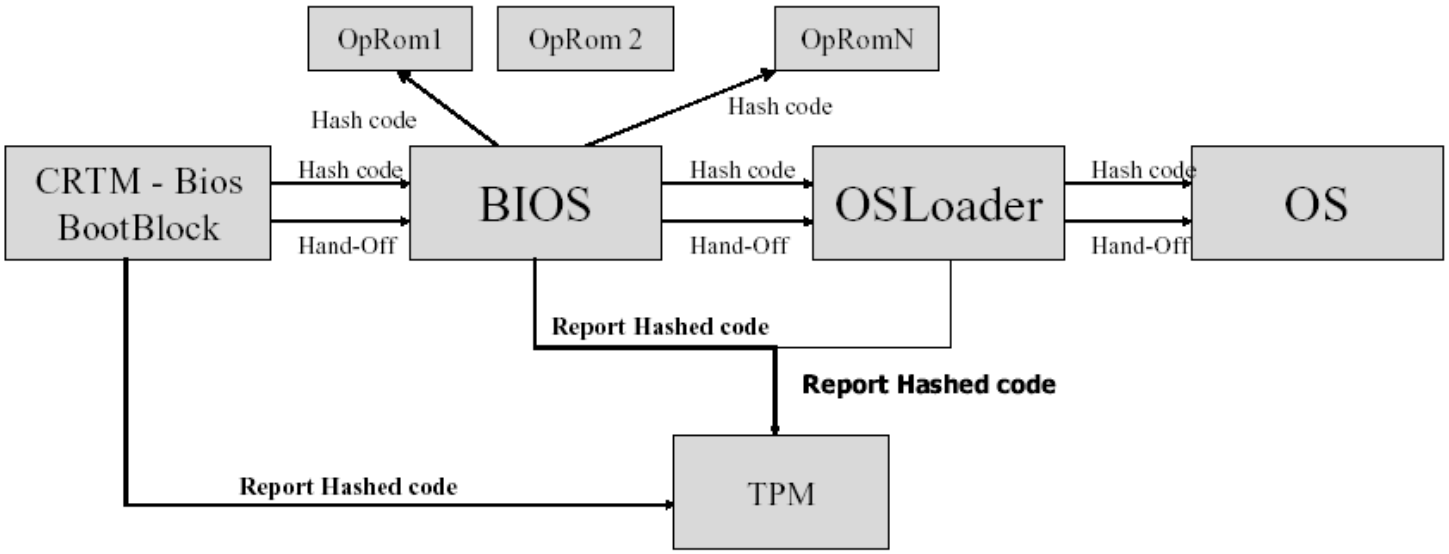
Issues with use of a P-CA

- The P-CA gets to see all the platform credentials, including the endorsement credential (and the public part of the EK).
- A TPM has only one EK, and hence the P-CA can link the AIK (and its associated identity) with a unique trusted platform.
- Hence, although a TPM can have many AIKs/identities, and hence a degree of anonymity/pseudonymity, this depends on the honesty of the P-CA, i.e. the P-CA can compromise this anonymity.
- As a result, an alternative protocol called DAA (Direct Anonymous Attestation) has been devised which avoids this problem.

Direct Anonymous Attestation (DAA)

- A P-CA is a threat to privacy since it is capable of:
  - user/TPM activity tracking; or
  - making unwanted disclosures of platform information.
- The DAA protocol removes the need to disclose the public value of the endorsement key to a P-CA.
- DAA is based on a family of cryptographic techniques known as zero knowledge proofs.
- DAA allows a TPM to convince a remote ‘verifier’ that it is indeed valid without disclosing the TPM public endorsement key, thereby removing the threat of a TTP collating data which may jeopardise the privacy of the TPM user.

Authenticated boot I



Authenticated boot II

- A TPM incorporates a set of Platform Configuration Registers (PCRs).
  - They are used to store platform software integrity metrics.
  - A TPM has several PCRs (a minimum of sixteen) and uses them to record different aspects of the state of the trusted platform.
  - Each PCR has length equal to a SHA-1 digest, i.e. 20 bytes.

Authenticated boot III

- Each PCR holds a value representing a summary of all the measurements presented to it from boot time:
  - This is less expensive than holding all individual measurements in the TPM;
  - This means that an unlimited number of results can be stored.
- A PCR value is defined as:
  - SHA-1(existing PCR value || latest measurement result).
- A PCR must be a TPM shielded location, protected from interference and prying.
  - The fewer sequences/PCRs there are, the more difficult it is to determine the meaning of the sequence;
  - The more sequences/PCRs there are, the more costly it is to store sequences in the TPM.

Reporting on integrity

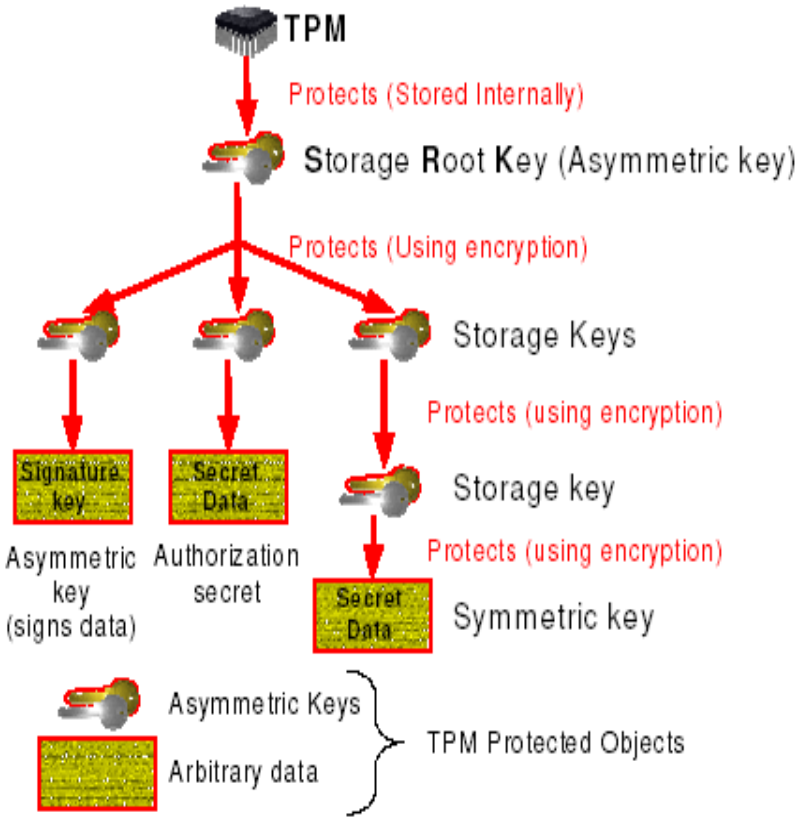
- Measurements reported to the TPM during or after the boot process cannot be removed or deleted until reboot.
- The attestation identity keys are used to sign integrity reports.
- The recipient of a signed integrity report can then evaluate the trustworthiness of the:
  - signed integrity measurements, by examining the platform identity certificate;
  - software configuration of the platform, using the reported measurements.

Authenticated versus secure boot

- The above measures provide authenticated boot, i.e. a means by which a third party can verify that a certain set of software has booted.
- They do not guarantee secure boot, i.e. guarantee that only a particular set of software is able to boot.

Secure storage I

- Each trusted platform contains a key hierarchy.
- At the root is the storage root key, SRK, stored securely in the TPM.
- Data or keys can be encrypted such that they can only be decrypted by the TPM.
- Asymmetric encryption is used.



Secure storage II

- Binding (data):
  - This TPM capability allows for external data to be encrypted using a public TPM parent key such that it can only be decrypted by the TPM.
- Wrapping (keys):
  - **TSS Wrap Key**: This TPM capability allows an externally generated key to be encrypted using a parent key.
- Wrapping variants:
  - **TSS Wrap key to PCR**: Similar to above, but the externally generated key is wrapped to PCR values [the key can only be revealed if the PCR values are correct];
  - **TPM Create wrap key**: Creates a TPM key, which may or may not be locked to PCRs.

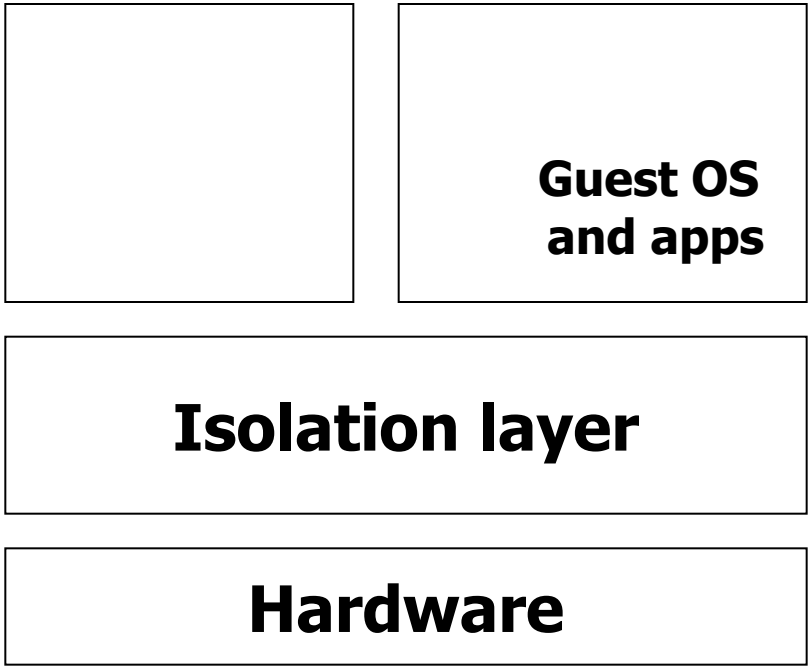
Secure storage III (sealing)

- **Sealing** (data / secret keys):
  - This is an important aspect of protected storage.
  - The seal operation can bind a secret to an individual TPM.
  - External data is concatenated with the value of an integrity metric sequence at the time the seal operation is performed, and then encrypted using the public key of a parent key pair.
  - It provides the capability to store a secret such that it can only be revealed by the TPM when the platform is in an specified software state.
  - The caller of the seal operation may choose not to wrap the secret to any PCR values.

Demonstrating privilege (access control)

- TPM access control functions support:
  - Owner authorised commands;
  - Protected objects;
  - Before a TPM is owned, the TPM is unavailable.
- Owner control is based on ‘Cryptographic authorisation’:
  - 20 bytes, for example a hashed password, or 20 bytes from a smartcard submitted to a hash algorithm, may be used;
  - Separate authorisation data must exist for the TPM owner as well as protected objects;
  - There are a number of authorisation protocols which protect against:
    - Man in the middle attacks;
    - Replay;
    - The exposure of the authorisation data.
- Physical presence:
  - Certain commands require the physical presence of a human, e.g. to push a switch.

Trusted computing fundamentals
Isolated execution environments



Example implementations include: OS-hosted VMM (VMWare workstation), Stand-alone VMM (Terra), Hybrid isolation layer (XEN 2.0), Hardware supported isolation layer (NGSCB).

- Protection from external interference
- Observation of isolated environment activity only by controlled inter-process communication
- Secure communication between isolated environments
- Trusted path between a program running in an isolated environment and I/O devices

2. Mobile security

- Objectives:
  - Review current status of mobile security;
  - Look at motivation for standardisation;
  - Review work of OMTP, in particular the TR1 document.

The need for a trusted mobile platform

- Ubiquitous adoption of mobile technologies.
- Expanding feature set available on mainstream devices:
  - Increasing number of services which require a device to be secure (Internet, DVB, music, video, gaming).
- It is predicted that mobile devices (such as smart phones and PDAs) will increasingly become targets of crimeware in the coming years.
- We are seeing the convergence of fixed and mobile technologies.

Mobile Security – Current status

(U)SIM to 3GPP network authentication	GSM = A3/A8 UMTS = f1-f5
User to SIM/device authentication	SIM/device PINs, biometric recognition schemes
SIM to device authentication	SIMLocking – T6, proprietary
Confidentiality of data in transit over wireless interfaces	GSM/UMTS = A5, f8 Bluetooth = E0 WLAN = WEP, WPA IrDA = None Wireless USB = None

Mobile Security – Current status

End-to-end data confidentiality	GSM/UMTS voice = None GSM/UMTS/WLAN data = VPN/IPsec, SSL/TLS
Access control to broadcast	3GPP MBMS, DVB-H, OMA BCAST = Security frameworks
User identity confidentiality	IMSI/TMSI scheme
Protection of user data	Proprietary
Secure 3rd party software download/installation	Java™ VM – Java MIDP 2.0 Symbian certificate-based schemes
DRM protected content	OMA DRM v2.0

Need for standardisation?

- Industry-centric:
  - Pool resources – top experts, peer review;
  - Broader customer base – lower costs, speedier time to market;
  - Prevents fragmentation – interoperability, reduced R&D costs .
- User-centric:
  - Increases confidence in devices;
  - Lower device cost;
  - Speedier adoption of new systems.

Standardisation bodies – OMTP

- **Open Mobile Terminal Platform**: founded in June 2004 by eight mobile operators.
- Aim: to simplify the customer experience of mobile data services and improve mobile device security.
- As the OMTP has grown, the complete mobile value chain is now represented.
- Security issues are addressed in the OMTP Application Security Working Group, and in the OMTP Hardware Security Requirements Group.
- OMTP TR0 and TR1 – define a threat model and a set of security requirements which enable the development of a trusted mobile environment.

OMTP TR1 – Advanced Trusted Environment

- The development of the **Advanced Trusted Environment** (**TR1**) specification is a continuation of the work completed in the initial **TR0** (**Trusted Environment**) specification.
- TR1 has two parts:
  - In Part I, the TR1 enabling technologies and their requirements are defined;
  - In Part II example requirements for implementations of potential operator use cases using TR1 enablers are considered.

OMTP TR1 – Enablers

- TR1 specifies the following trust-enabling functionalities:
  - Trusted execution environment;
  - Secure storage;
  - Flexible secure boot;
  - Run-time integrity checking;
  - Secure access to user Input/Output facility;
  - Secure interaction of UICC with mobile equipment.

Need for TCG standardisation revisited

- Industry-centric:
  - With the convergence of fixed and mobile technologies:
    - Achieve TCG interoperability;
  - Security assessment:
    - Assess each device or assess a standard?
- User-centric:
  - Prevention of unauthorised use and theft of content and data.
- Possible benefits:
  - Short-term – Industry/Operators;
  - Long-term – End users.

3. A trusted mobile platform

- Objectives:
  - Review role of the TCG MPWG;
  - Main standardisation bodies;
  - Building a TCG TMP;
  - Use case definition;
  - Requirements analysis (example: OMA DRM v2.0 use case).

The MPWG

- The **Mobile Phone Working Group** (**MPWG**), one of a number of platform-specific working groups within the TCG, works on the extension and adoption of trusted computing concepts for the mobile device.
- The group builds on existing specifications and concepts to address specific characteristics of mobile devices, such as:
  - connectivity; and
  - limited capability.

MPWG work programme

- Use case definition;
- Requirements analysis;

} Similar to work completed by the OMTP

- Trusted Mobile Platform (TPM) specification.

Use case definition

- The use case document was written to guide subsequent technical specification work within the TCG MPWG.
- The 11 use cases outline possible scenarios for use of devices meeting the TCG TMP specifications.
- The use case document laid a foundation for the ways in which the MPWG:
  - derived requirements that address situations described in the use cases;
  - specified an architecture based on the TCG architecture that meet these requirements;
  - specified the functions and interfaces that meet the requirements in the specified architecture.

Use cases I

1. Platform integrity:
   - **Goal**: to ensure that a device possesses and runs only authorised operating systems and hardware.
2. Device authentication:
   - **Goal**: to ensure the device can store and protect identification information, e.g. keys, such that a device can be securely authenticated to a service provider or a network provider (the device identity may or may not be bound to the user).
3. Robust DRM implementation:
   - **Goal**: to assure both service and content providers that device implementations of DRM specifications (e.g. OMA DRM V2) are robust and can be trusted to protect their digital content (robust in this context indicates resistance to a specified level of attack).

Use cases II

4. SIMLock/Device personalisation:
   - **Goal**: to ensure that a mobile device remains locked to a particular network, network subset, service provider, corporation or (U)SIM until the device is unlocked in an authorised manner.
5. Secure software download:
   - **Goal**: to enable the secure download of application software or updates and/or firmware updates or patches.
6. Secure channel between device and UICC:
   - **Goals**: to enable the UICC to be aware of the trust status of the device it is inserted into, and vice versa;

     to enable establishment of a secure channel between the device and the UICC.

Use cases III

7. Mobile ticketing:
   - A ticket is a proof of access/usage rights to a particular service.
   - Tickets can be purchased or redeemed using a variety of mechanisms, including phone call, internet, as well as store front .
   - A ticket may be downloaded via any channel to a mobile device at the time it is purchased.
   - **Goal**: to ensure that downloaded tickets are not duplicated or modified in any way, e.g. to change their rights as purchased
   - Ticket presentation is a procedure to prove the possession and validation of such a data object in a mobile device.
   - A ticket is consumed if it is presented and accepted.
   - **Goal**: to ensure that, after ticket consumption, the rights represented by the data object are “expired”.

Use cases IV

8. Mobile payment:
   - Incorporates a payment application and user account information.
   - **Goals**: to execute a payment protocol between a device application and a point of sale;

     to enable an account owner to authorise the precise payment amount.
9. Software use:
   - **Goal**: to ensure a user can securely use an application, i.e. that a platform enforces predefined software use policies (covering: subject access, rights definition/enforcement and revocation).

Use cases V

10. Proving platform and/or application integrity to end user:
    - **Goal**: to ensure an end user can reliably learn whether a device or application can be trusted.
11. User data protection and privacy:
    - This information may include, but is not limited tom the following types of personally identifiable information: contact /address books, wallets, credentials, identity tokens.
    - **Goal**: to ensure a device user can protect his/her information from being accessed, viewed, and/or copied by unauthorised entities.

Requirements analysis example: OMA DRM v2

- The Open Mobile Alliance (OMA) was founded in June 2002.
- One of the original objectives of the OMA was to define a DRM specification set for use in the mobile environment.
- OMA DRM v1 was approved as an OMA enabler specification after full interoperability testing had been completed in 2004.
- Following this, in 2004, work on OMA DRM v2 was completed.
- OMA DRM v2 builds upon the version 1 specifications to provide higher security and a more extensive feature set.

OMA DRM v1 – goals/functions

- Main goals:
  - Timely and inexpensive to deploy;
  - Easy to implement on mass market mobile devices;
  - Avoid the need for the roll-out of a costly infrastructure.
- Three classes of DRM functionality:
  - Forward lock;
  - Combined delivery;
  - Separate delivery:
    - encrypted content;
    - rights object and decryption key for the associated content delivered via SMS.

OMA DRM v1 – weaknesses

- A rights issuer has no way in which to determine whether the requesting device supports DRM.
- In the separate delivery DRM class, where the content is encrypted, the content encrypting key is not protected.
- The device has no way of authenticating the rights issuer and therefore may be sent bogus rights objects from an entity claiming to be the legitimate rights issuer.

OMA DRM v2 – overview

- Intended to address weaknesses in v1.
- Both device authentication and rights issuer authentication are provided.
- Mechanisms are deployed in order to protect the confidentiality of media objects.
- Mechanisms are also deployed so that the OMA DRM v2 agent can determine whether a media object received from a RI has been modified in an unauthorised way.
- Also support for an extended feature set: subscription, streaming content, reward schemes, domains, unconnected devices.

OMA DRM v2 – security features

- The rights object acquisition protocol suite:
  - The 4-pass registration protocol;
  - The 2-pass rights acquisition protocol;
  - The 1-pass rights acquisition protocol;
  - The 2-pass join domain protocol;
  - The 2-pass leave domain protocol.
- A trust model enables an RI to obtain assurances about DRM agent behaviour and the robustness of the DRM agent implementation:
  - It is the responsibility of the Content Management Licensing Administrator (CMLA), or a similar organisation, to provide a trust model, i.e. robustness rules, and to define actions which can be taken against a manufacturer which builds devices which are not sufficiently robust.

Use case analysis

- Analyse:
  - OMA DRM v2 agent installation;
  - the Rights Object Acquisition Protocol suite.
- In each case perform:
  - Threat analysis;
  - Security requirements extraction;
  - Mapping of requirements to TPM specifications.
- We briefly review two examples of this process.

OMA DRM v2 agent installation I

- The OMA DRM application must be installed, stored and executed on the device
- **Threats**:
  - Unauthorised modification of the OMA DRM v2 agent code on installation onto the device;
  - Unauthorised modification of the OMA DRM v2 agent code while in storage on, or while executing on, the device.

OMA DRM v2 agent installation II

- **TMP security requirement**:
  - the TMP SHALL provide a mechanism so “an OMA DRM v2 agent can perform self-checking of the integrity of its component parts such that unauthorised modifications will be expected to result in a failure of the implementation to provide the authorised authentication and/or decryption function” [CMLA].

OMA DRM v2 private key protection I

- Every OMA DRM v2 agent is provisioned with a unique key pair:
  - The private key from this key pair is used by a OMA DRM v2 agent to generate digital signatures so that a rights issuer can authenticate a particular DRM agent;
  - The public key from this pair is also used by rights issuers in order to distribute rights object encryption keys which protect content encryption keys used to encrypt content.
- **Threats**:
  - Unauthorised reading/copying of the OMA DRM v2 agent private key on installation into the device;
  - Unauthorised reading/copying of the OMA DRM v2 agent private key while in storage and in use on the device.

OMA DRM v2 private key protection II

- **TMP security requirements**:
  - The TMP SHALL provide a mechanism such that the OMA DRM v2 agent private key can be confidentiality-protected during its installation;
  - The TMP SHALL provide a mechanism such that the OMA DRM v2 agent private key can be confidentiality-protected while in storage or in use on the device.

4. Elements of a TCG TMP

- Objectives:
  - Specify the *stakeholders* in a TMP;
  - Describe the TCG TMP;
  - Outline the operation of secure boot;
  - Maintaining integrity after boot;
  - Describe the MRTM and MLTM;
  - Consider the role of software isolation.

TMP stakeholders I

- **Users**, who store their data in the platform:
  - There may be multiple user stakeholders in a platform;
  - For example, an employee or a consumer may be a user stakeholder.
- **Service providers**, who provide services consumed in a platform:
  - There may be multiple service provider stakeholders in a platform;
  - Examples of services include: corporate services for employees; content distribution services for consumers; an address book; a diary.

TMP stakeholders II

- **Communications carriers**, who are specialist service providers providing cellular radio access for the platform:
  - There may be multiple communications carrier stakeholders in a platform.
- The **device manufacturer**, who provides the internal communications within a platform and typically provides all the hardware resources within a platform:
  - There is a single device manufacturer stakeholder in a platform.

Cellular-radio conformance

- Cellular-radio enabled platforms must:
  - conform to regulations that govern network protocols;
  - allow unrestricted access to certain network parameters.
- Regulations also dictate that certain network parameters (such as the IMEI) must be unique to individual platforms.
- Otherwise, a cellular-radio enabled platform cannot operate (and should not operate, since there is the danger that it may damage/disable the network).

A TCG TMP

- Conventional TCG-enabled platforms enforce:
  - the rights of a single platform Owner (who has exclusive control over the data protection mechanisms in the platform); and
  - the rights of multiple data owners (who use the data protection mechanisms, with permission from the platform Owner).
- If a cellular-radio enabled platform was just a conventional TCG-enabled platform, it follows that an Owner or User who turned off the platform TPM would prevent the radio from operating.

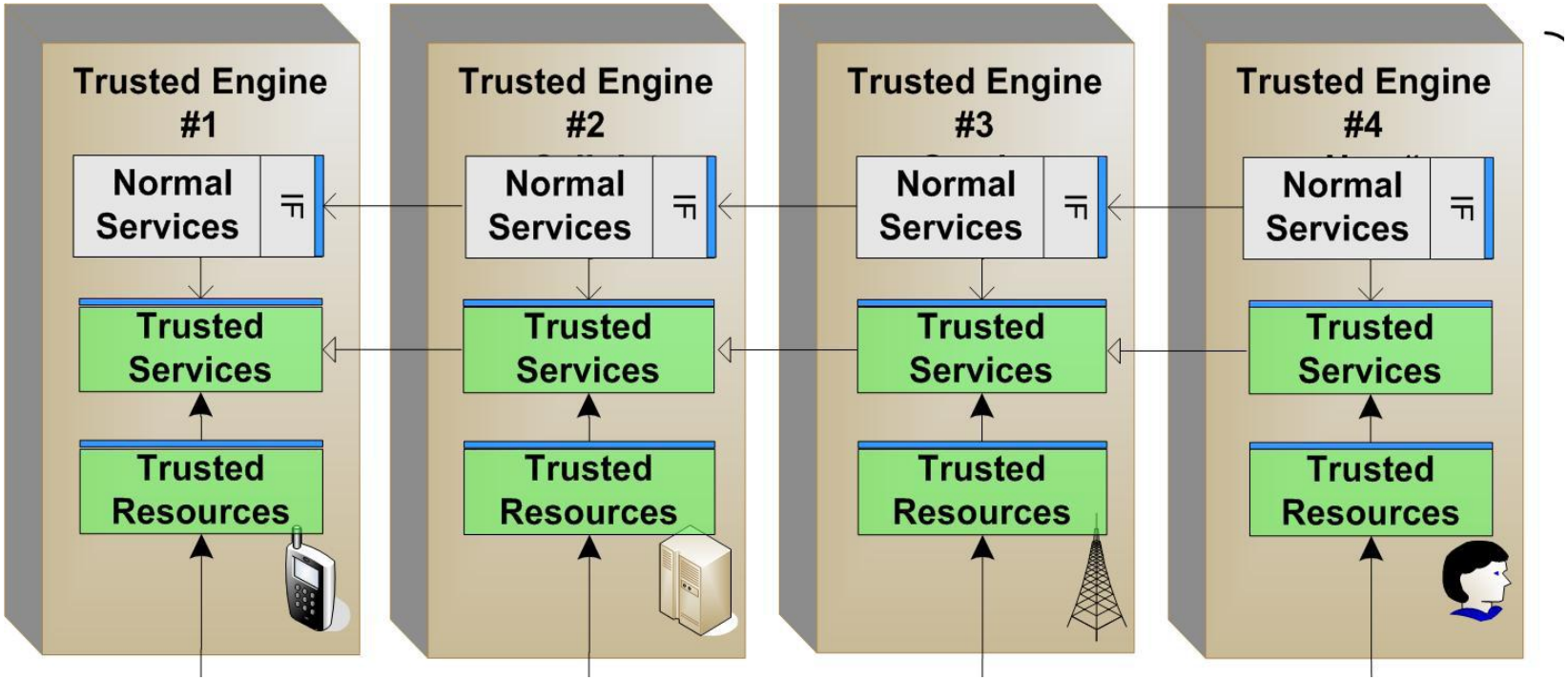
A TCG TMP

- To maintain the right of an Owner or user to turn off his TPM, the TMP specification generalises the concept of a platform to mean a set of trusted “engines”.
- A TMP, as defined by the TCG, is made up of a set of such engines.
- An engine is defined as a construct capable of:
  - manipulating data;
  - providing evidence that it can be trusted to report the current state of the host platform; and
  - providing evidence about the host platform's current state.
- Each stakeholder on a trusted mobile platform has its own engine.

A TMP engine

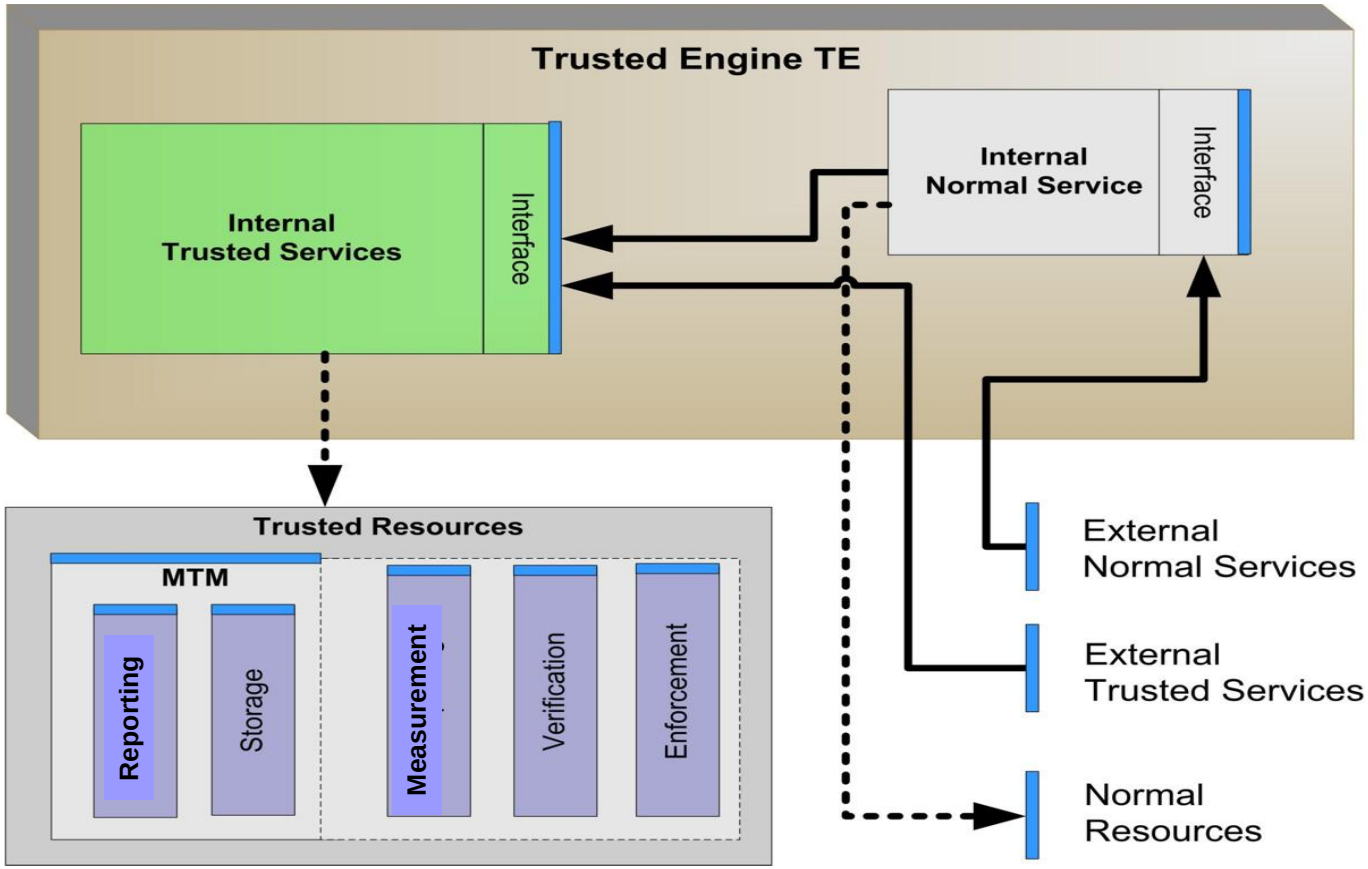
- Each engine provides platform services on behalf of its stakeholder, and also incorporates functionality similar in many ways to a ‘traditional’ TCG trusted platform.
- An engine can:
  - access a set of trusted resources;
  - obtain and use an endorsement key and/or attestation identity keys;
  - provide evidence of its trustworthiness as a trusted platform;
  - report evidence regarding its current state;
  - import and/or export services, shielded capabilities and protected functionality;
  - implement arbitrary software functionalities such as trusted and/or normal services.

A TCG TMP



[Schmidt, Kuntze and Kasper]

A TMP engine – overview



[Schmidt, Kuntze and Kasper]

[Just as a TPM provides trusted resources (reporting and storage) to a PC platform, a Mobile Trusted Module (MTM) provides trusted resources to a mobile platform]

A TMP engine – components

- An engine is made up of trusted resources:
  - The following Roots of Trust are defined for the mobile domain:
    - Root of Trust for Storage (RTS);
    - Root of Trust for Reporting (RTR);
    - Root of Trust for Measurement (RTM);
    - **Root of Trust for Verification (RTV);**
    - **Root of Trust for Enforcement (RTE)**.
  - Each root of trust Provides evidence of its trustworthiness:
    - directly, by proving knowledge of secrets (EK, AIK) and associated credentials that can only be accessed by authenticated subjects of the stakeholder; or
    - indirectly by providing measurements.

A TMP engine – services

- An engine provide a variety of services:
  - Trusted services:
    - A trusted service customises trusted resources;
    - Trusted services are intended to provide reliable measurements of their current state and to provide evidence of the state of other normal services or resources;
  - Normal services:
    - Normal services customise normal resources and implement functionality.

Mandatory TMP engines

- An engine may be categorised as **mandatory** or **discretionary**.
- A mandatory engine provides the mandated functionality of a TMP; e.g. functions required to comply with regulations governing the operation of mobiles in cellular radio systems.
- Mandatory engines must be supported by a **Mobile Remote-owner Trusted Module (MRTM)**, which supports secure boot and does not permit a local operator to remove the stakeholder from the engine.
  - EK/AIK: EK may not be defined – AIK in permanent data;
  - Ownership: TPM_TakeOwnership command not required to be present on a MRTM;
  - SRK: May be pre-installed;
  - Secure boot support.

Discretionary TMP engines

- A discretionary engine provides services that may be added, removed and turned on/off without the consent of any external service provider.
- Discretionary engines must be supported by a **Mobile Local owner Trusted Module (MLTM)**, which is not required to support secure boot, and which permits a local operator to remove the stakeholder from the engine.
- The device manufacturer and device owner define the mandatory engines that can exist on their platforms.
- A device owner can specify which discretionary engines are permitted on his/her platform.

The endorsement key pair

- The EK is an optional element for the mobile environment.
- If it is used, it must be:
  - created;
  - bound to the device;
  - non-migratable.
- The EK for the device manufacturer's engine will be generated by the device manufacturer and installed in the engine's RTS:
  - generated off-chip and inserted; or
  - generated using on-chip commands.
- An EK credential must also be generated and shipped with the device.
- Replacement of a the EK pair must be prevented before the device is shipped.

Taking ownership – MRTMs

- In the case of a remotely owned engine, the engine's MRTM is typically already enabled, activated and owned by the time a user takes possession.
- This MUST be true of a device manufacturer's engine.
- A remote owner may, however, be able to take and clear ownership at a later date using the TPM_TakeOwnership and TPM_OwnerClear commands (both optional for the MRTM).
- In all cases, however, the remote owner must be protected from a user attempting to remove the remote owner's ownership or attempting to disable or deactivate the remote engine's MRTM.
- In the case that the user is the device owner and the engine is on a DO controlled list, the user can always remove the engine entirely.

Taking ownership – MLTMs

- In the case of a locally owned engine, the engine's MLTM will typically not yet have a owner when the user takes possession
- It should be possible to establish the owner through physical presence.

Storage root key generation

- For the device manufacturer engine, the SRK pair may be generated externally and inserted into the engine during manufacture.
- If an AIK is pre-generated and installed during manufacture, the SRK must be generated and installed at the same time.
- The SRK may be implemented as a symmetric key or an asymmetric key pair.

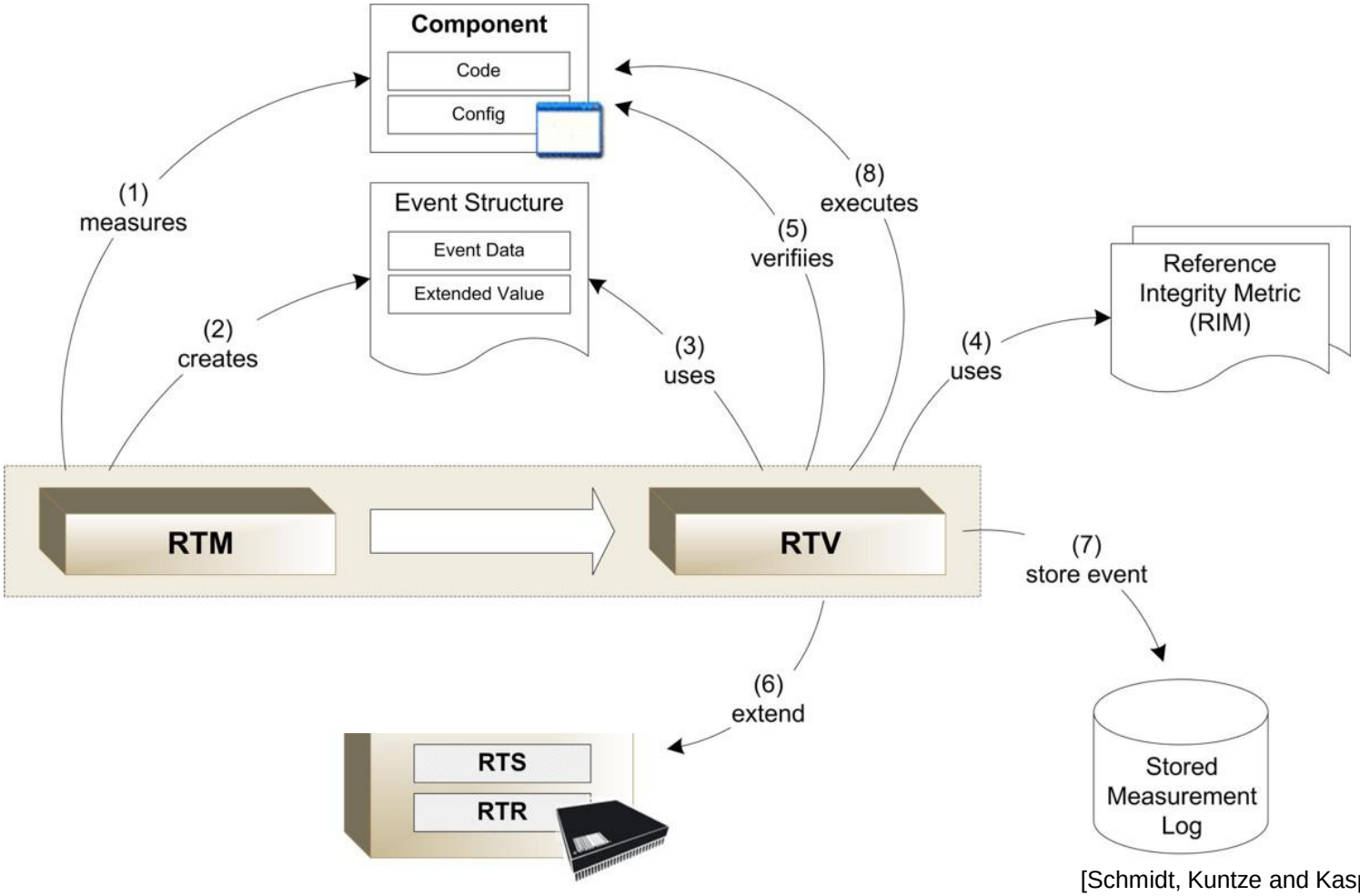
AIK pair generation

- AIK pairs and credentials can be generated and installed in several ways.
  - AIK generation with an EK pair:
    - A AIK pair is generated in the traditional way (using TPM commands) and the public AIK from a pair is certified as belonging to an engine by a P-CA;
  - AIK generation without an EK:
    - If no EK pair exists on a device AIK pairs and the associated credentials may be generated and installed during the manufacture process.

Secure boot

- The TCG MPWG has defined a secure boot process.
- Rather than measuring and recording – as is the case with authenticated boot – a secure boot process allows each platform component to be:
  - **Measured**,
  - **Verified** (by RTV), in which the measured value is compared against a reference value (which indicates what the measurement 'ought to be'), and
  - **Acted upon**, where, if it is discovered that a platform's component measurement is not what is 'ought to be', then the boot process can be aborted (by the RTE).

Secure boot
The process (mandatory engines)



[Schmidt, Kuntze and Kasper]

Secure boot
RIMs and TIMs

- A Target Integrity Metric (TIM) is the actual measurement of a software component taken by the RTM or measurement agent
- A Reference Integrity Metric (RIM) is a reference value used to compare with a TIM.
- A RIM provisioning method needs to:
  - authenticate source;
  - verify authorisation of source to provide RIMs;
  - verify integrity, freshness and validity of RIMs.
- A RIM_Cert is an authenticated and integrity protected structure containing a RIM and some auxiliary information.

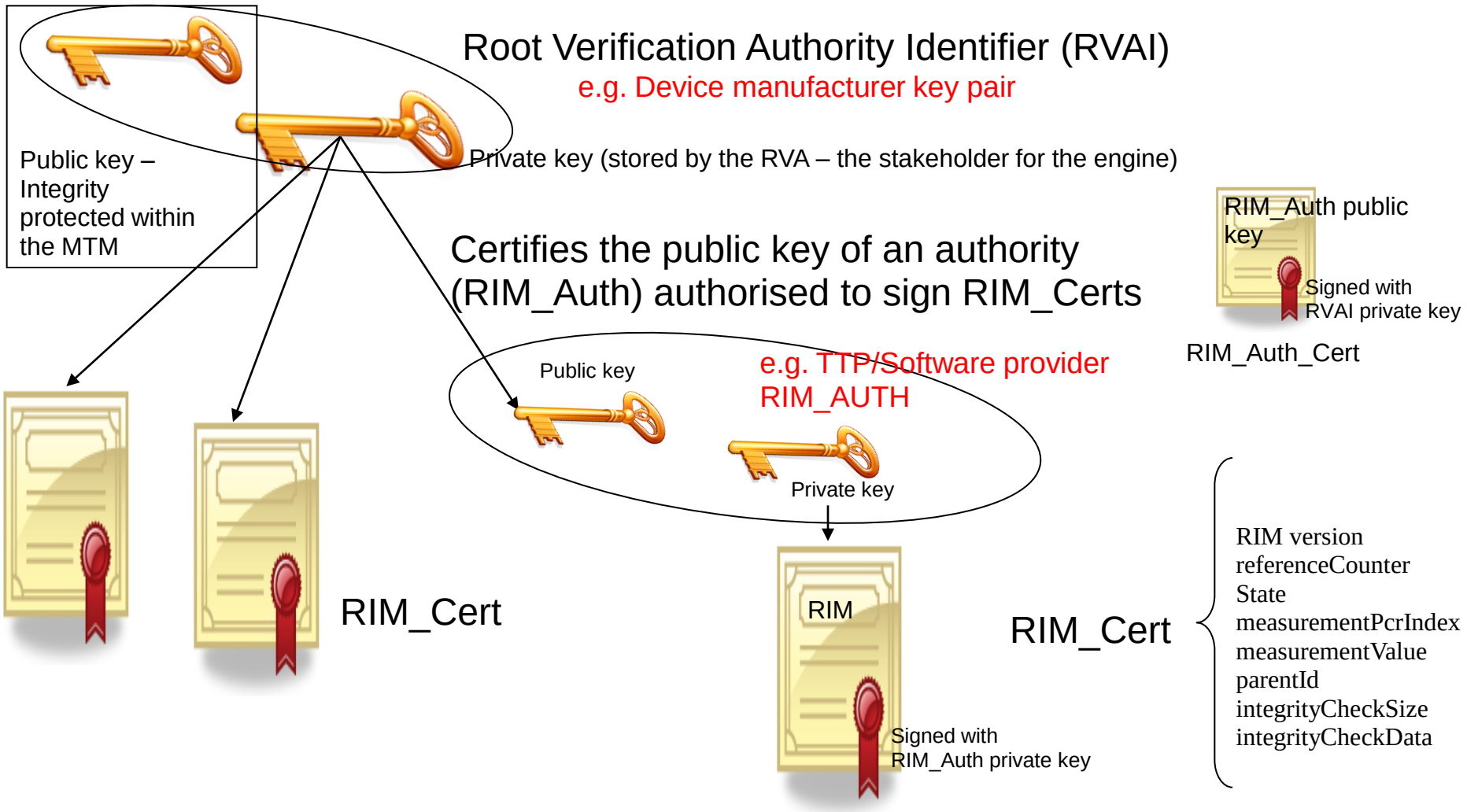
Secure boot
RIM creation/verification

- The parties which create authentic and authorised RIM_Certs are called RIM_Auths:
  - An external RIM_Cert may be authenticated using a digital signature;
  - An internal RIM_Cert may be authenticated using a message authentication code.
- The keys used to verify RIM_Certs are called TPM verification keys.
- For each MTM, a key hierarchy used to authorise RIM_Certs can be set up.

Secure boot
Supporting PKI

- Each engine must be pre-configured with a public key called the Root Verification Authority Identifier (RVAI).
- This key must be integrity protected, e.g. via ROM or NV storage in the MTM (it can also be signed using a key stored within the MTM).
- The Root Verification Authority (who owns the private key) is the stakeholder for the engine. It acts as the root CA, and can:
  - directly sign RIM_Certs;
  - delegate the authority to sign RIM_Certs to RIM_Auths in the form as RIM_Auth_Certs;
  - delegate the authority to authorise RIM_Auths in the form of RIM_Auth_Certs.

Secure boot – RIM provisioning (external RIM_Certs)



RIM_Auth_Cert

tag	TPM_TAG_VERIFICATION_KEY
usageFlags	defines the capabilities for the key defined in key data, i.e. whether it can authorise RIM_Auths or sign RIM_Certs
parentID	Identifier for the key used to authorise the key contained in keyData If parentID = none then this is a “root key”
myID	Identifier for the key structure
referenceCounter	Defines the validity of the structure
keyAlgorithm	Identifier for the algorithm to be used with the key held in keyData
keyScheme	The method by which the integrityCheckData can be verified
extensionDigestSize	Length in bytes of the buffer extensionDigest
extensionDigest	Contains a hash of proprietary extension data
keySize	Length of the buffer KeyData
keyData	Contains the key for verifying the integrityCheckData field
integrityCheckSize	The length of the integrityCheckData buffer
integrityCheckData	An integrity check for the TPM_Verification_Key The method by which to verify is defined in the object referenced by parentId

RIM_Auth_Validity_Lists

- If the Root Verification Authority (or other RIM_Auth acting as a CA) wishes to revoke delegated authorisation, then it SHOULD do so by signing a periodic RIM_Auth_Validity_List indicating the identifiers of its delegates that are still valid.
- Every RIM_Auth which signs Validity Lists MUST ensure it always has signed a Validity List whose “valid from” and “valid to” fields in UTCtime format enclose the current date and time.
- Whether or not a RIM_Auth signs RIM_Auth Validity Lists MUST be indicated by a usage flag in the TPM_Verification_Key structure.

RIM_Cert

tag	TPM_TAG_RIM_CERTIFICATE
rimVersion	A version number for the RIM certificate
referenceCounter	Defines the validity of the structure
state	Defines the state the system must be in at the time of use
measurementPcrIndex	The PCR index that is to be extended using the defined measurement value
measurementValue	The measurement value to be extended to the specified PCR
parentId	The identifier for the key used to verify this structure
extensionDigestSize	Length in bytes of the buffer extensionDigest
extensionDigest	Contains a hash of proprietary extension data
integrityCheckSize	The length of the integrityCheckData buffer
integrityCheckData	An integrity check for the TPM_RIM_CERTIFICATE The method to be used to verify it is defined in the object referenced by parentId

RIM_Validity_Lists

- RIM_Auths that are able to sign RIM Certificates SHOULD be able to revoke such certificates (typically also issuing a replacement).
- If a RIM_Auth is able to revoke its RIM_Certs, then it SHOULD do so by signing a periodic RIM_Validity_List indicating the serial numbers of its certificates that are still valid.
- Every RIM_Auth which signs Validity Lists MUST ensure that it signs Validity Lists whose “valid from” and “valid to” fields (in UTCtime format) enclose the current date and time.
- Whether or not a RIM_Auth signs RIM Validity Lists MUST be indicated by a key-usage flag in the TPM_Verification_Key structure.

Internal and external RIM_Certs

- The full set of external RIM_Certs, RIM Validity Lists, RIM_Auth_Certs and RIM_Auth_Cert revocation information (RIM_Auth Validity Lists etc.) defines a complex privilege structure.
- It is not required that each verification agent (especially the RTV) is able to process this whole structure during each boot and hence determine what is really a valid RIM.
- This problem is addressed by using a special RIM Conversion Agent to process all of the external RIM_Certs and map from External RIM_Certs to Internal RIM_Certs.
- These Internal RIM_Certs can then be more easily handled by the RTV and other verification agents.

Maintaining integrity after boot I

Maintenance:

- Hardware protection of software:
  - The most basic form of hardware protection is implementation of program code in ROM, and storage of critical and unchanging data in ROM and/or One Time Programmable (OTP) memory.
  - More advanced levels of additional hardware protection (e.g. epoxy coverings, and protection against power analysis and fault induction attacks) can then be added to protect this ROM and OTP.
  - The design intention is that capabilities protected by hardware cannot be interfered with by either rogue software or physical attacks on the device.

Maintaining integrity after boot II

- Software isolation using hardware:
  - For example, via:
    - putting critical functionality on a separate chip, e.g. using a TPM;
    - software support (e.g. full virtualisation).
- Simplified software:
  - It is commonly accepted that it is easier to certify that code has no errors if the code is limited in size and complexity.

Maintaining integrity after boot III

- Software restriction:
  - At the device OS level, a comparatively simple/inexpensive approach is to not provide any tools to import native code onto the platform.
  - If download of applications is required (it may not be required), this is only enabled via an application environment based on a virtual machine.
  - A related approach is to use an operating system with strong access control and a policy system, which can be tailored to protect the resources.
  - In particular, APIs available to applications running on the OS are segregated into at least two privilege classes, and the OS prevents applications using the more privileged APIs unless they are recognised at install-time as “trusted” under the security policy.

Maintaining integrity after boot IV

- Software load:
  - Security checking of software prior to installation:
    - Compatible device information?
    - RIM – application TIM match?
    - Is code recognised as trusted?
    - Signed? Revoked?
    - Matches malware signature?
  - Security checking prior to launch.

Detection/Reaction:

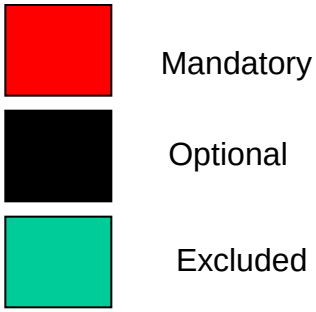
- Run-time measurement, verification and reaction (watch-dog mechanism):
  - RIM_run;
  - RIM_run_Auth;
  - Time-based;
  - Event-based.

The MRTM

- Secure boot support

- Admin start-up and state
- Admin testing
- Admin opt-in
- Admin ownership
- The GetCapability commands
- Auditing
- Administrative functions
- Storage functions
- Migration
- Maintenance
- Cryptographic functions
- Endorsement key handling
- Identity creation and activation

- Integrity collection and reporting
- Changing AuthData
- Authorisation sessions
- Delegation
- Non-volatile memory
- Session management
- Eviction
- Timing ticks
- Transport sessions
- Monotonic counter
- Direct anonymous attestation

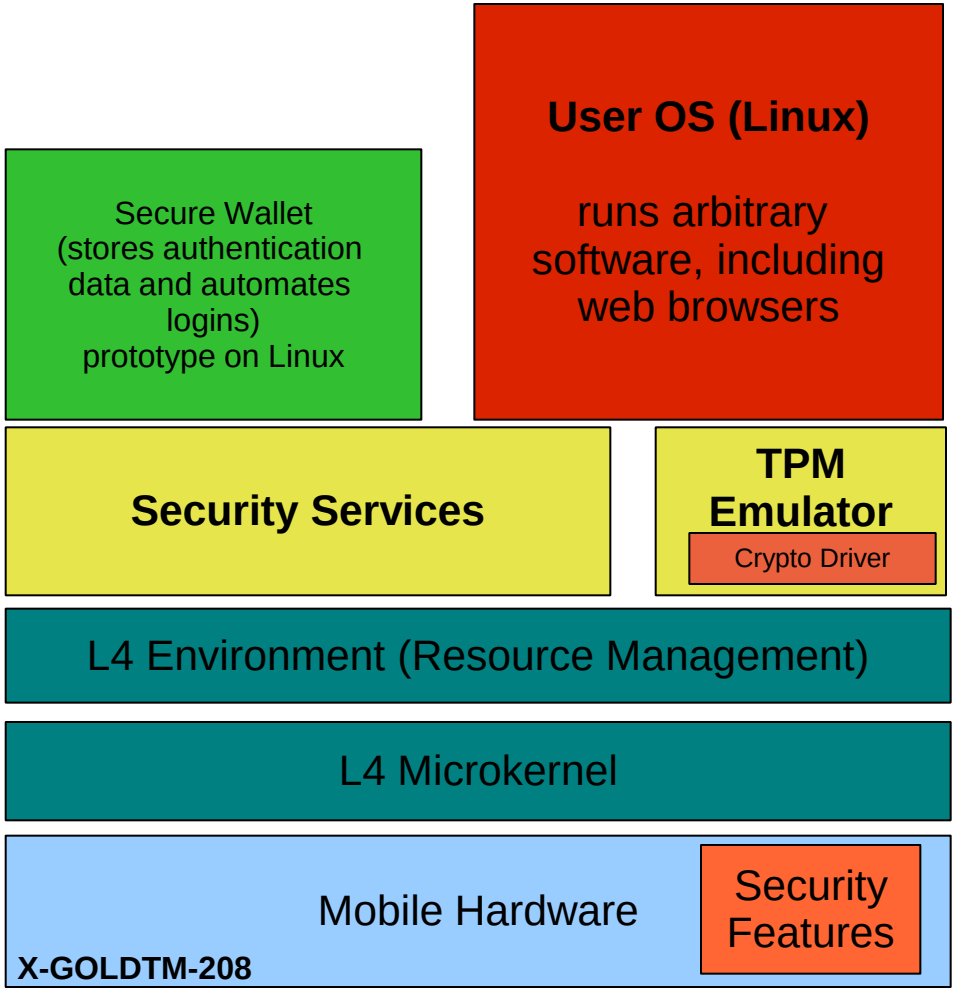


The MLTM

- Admin start-up and state
- Admin testing
- Admin opt-in
- Admin ownership
- The GetCapability commands
- Auditing
- Administrative functions
- Storage functions
- Migration
- Maintenance
- Cryptographic functions
- Endorsement key handling
- Identity creation and activation
- Integrity collection and reporting
- Changing AuthData
- Authorisation sessions
- Delegation
- Non-volatile memory
- Session management
- Eviction
- Timing ticks
- Transport sessions
- Monotonic counter
- Direct anonymous attestation

Software Architecture in OpenTC

- Proof of concept
- OpenTC system partially ported to mobile HW
- PC-based prototype
- Current status on mobile platform:
  - Microkernel
  - L4 + L4env
  - TPM emulator
  - Crypto Driver
  - L4Linux



Acknowledgements

- I would like to thank all the partners in the OpenTC project who helped us develop a wide range of teaching materials.
- I must particularly thank Eimear Gallery, who worked on OpenTC almost from the beginning to the end, for preparing a large proportion of the material presented in this lecture.

The Open-TC project was co-financed by the EC (11/05-4/09).

If you need further information, please visit the website www.opentc.net or contact the coordinator:

Technikon Forschungs- und Planungsgesellschaft mbH
Richard-Wagner-Strasse 7, 9500 Villach, AUSTRIA
Tel. +43 4242 23355 – 0
Fax. +43 4242 23355 – 77
Email coordination@opentc.net