Introduction to the Trusted Platform Module

Design Goals and Capabilities

CSC 495/680 Lecture
September 13-?, 2010

References:
• A Practical Guide to Trusted Computing, Chapters 2-3
• Trusted Computing Group documentation

TPM Design Goals

• Book lists:
  – Secure report the environment that booted
  – Securely store data
  – Securely identify the user and system
  – Support standard security systems and protocols
  – Support multiple users on the same system while preserving security among them
  – Be produced inexpensively
• Book states will be FIPS 140-2 and CC EAL3 (or EAL4+)
  – What does this mean?

What is “Assurance”??

• “Assurance” refers to “ways of convincing others that a model, design, and implementation are correct.”
  – From “Security in Computing” by Charles and Shari Pfleeger
  – I’d add “ways of convincing others or yourself…”
• Can you quantify “confidence levels”?
• Need language for assurance levels and properties, so we can see if a system is appropriate.
• Assurance tools: evaluation, testing, formal verification

Evaluation Standards

• A key characteristic of “trusted systems” is a security-centric evaluation
• Valuable properties:
  – Fit systems into a well-understood framework
  – Use consistent language and criteria
• Influential evaluation standards:
  – TCSEC (“Orange Book”): U.S. DoD
  – ITSEC: European framework
  – U.S. Federal Criteria: NIST standard (not DoD-specific)
  – Common Criteria: Merges successful ideas from other standards

Common Criteria

• Overview
  – Separates features from assurance
  – Functionally general-purpose, based on Protection Profiles and vendor-defined Security Targets
  – Assurance levels given as Evaluation Assurance Levels
• How it works:
  – Evaluations by commercial testing labs accredited by NIST’s National Voluntary Laboratory Accreditation Program (NVLAP)
    • Called the “Common Criteria Testing Laboratories (CCTL)
  – U.S. National Information Assurance Partnership Common Criteria Evaluation and Validation Scheme (CCEVS) Validation Body – managed by NIST and NSA
    • Approves CCTLs
    • Maintains NIAP Validated Products List

Common Criteria Evaluation Assurance Levels (EALs)

• EAL-1: Functionally tested
• EAL-2: Structurally tested
• EAL-3: Methodically tested and checked
  – Thorough testing, but not requiring controlled design process
• EAL-4: Methodically designed, tested, and reviewed
  – Reflects good traditional software development practices from design forward
• EAL-5: Semiformally designed and tested
• EAL-6: Semiformally verified design and tested
• EAL-7: Formally verified design and tested
Design Goal 1
Securely report the environment that booted

- Obvious fact 1: You can’t trust software to tell you whether it is trustworthy
  - Malicious software would just lie!
  - Honest software in untrustworthy environment can’t tell if corrupted!

- A TPM should be tightly tied into system from very beginning of boot sequence
  - Tight integration makes a TPM different from a smartcard
  - TPM uses Platform Configuration Registers (PCRs) to securely hold measurements/logs of boot process
    - Initial, very small, trusted part of BIOS kicks things off
    - Core Root of Trust for Measurement (CRTM)
  - Each stage in boot process measures/records next stage before executing

Source: Figure 2.1 in book (p.16)

IPL: “Initial Program Load” (boot loader)
IPS: ??? Typo?

Note: Does not stop bad code from loading!!! Just makes it possible to determine when system is trusted (and can lock secrets to trusted environments).

Design Goal 1
Securely report the environment that booted

- TPM (v1.1) has at least 16 PCRs
  - Can only be reset through system reboot
  - PC-specific implementation defines use of first 8
  - Remainder can be used in custom system-specific ways

- TPM (v1.2) adds 8 more – dynamic PCRs
  - With support from rest of the system (CPU+chipset) can be reset in carefully controlled situations
    - Intel calls this support “TXT” (Trusted Execution Technology)
    - AMD calls it “SVM” (Secure Virtual Machine)

Design Goal 1
Securely report the environment that booted

- How are measurements securely reported?
  - Scenario: A user Alice wants to know what’s in PCR0
    - Alice can be a local or remote user

What is in PCR0?
729a4...c3

Signature solve our problem – but not quite right here

- Problem 1: Does the signature mean it came from a TPM?
  - Solution: (PK, SK) is an identity key, certified by a PrivacyCA

Design Goal 1
Securely report the environment that booted

But… Alice doesn’t talk directly to the TPM

Lots of layers between Alice and TPM – and any could be corrupted

Good value is 729a4...c3

Maybe even a network connection here

729a4...c3

Verify(PK, S)

Bad things can happen in here...

S = Sign(SK, 729a4...c3)
Design Goal 1
Securely report the environment that booted

- **Problem 2**: Could S be a replay of an earlier captured S?

- **Solution**: Send a random (non-repeating) nonce along with request

  Signatures solve our problem – this one is done right!

Nonce, PCR0?

Good things can happen in here….

S = Sign(SK, Nonce || 729a4…c3)

Verify(PK, S)

Design Goal 2
Securely store data

- Secure storage depends on cryptography and keys
- Keys are classified according to their use …
  - Storage keys, Binding keys, Identity keys, Signature keys
- … their properties …
  - Migratable? Restricted to certain environment (PCRs)?
- … and authorization
  - Do you need to know a secret to use the key?

Best way to understand TPM keys is from the specification

- Storage keys, Binding keys, Identity keys, Signature keys
- their properties
- Migratable? Restricted to certain environment (PCRs)?
- and authorization
  - Do you need to know a secret to use the key?

- Best way to understand TPM keys is from the specification

TPM Keys
The Data in a TPM_KEY12 Structure

- TPM_STRUCTURE_TAG
- TPM_KEY_USAGE
- TPM_KEY_FLAGS
- TPM_AUTH_DATA_USAGE
- TPM_KEY_PARAMS
- TPM_STORE_PUBKEY
- TPM_ENC_SCHEME
- TPM_SIG_SCHEME
- TPM_ALGORITHM_ID

Important: Keys have a single use! So an identity key can only be used to sign TPM-generated data (unlike a signing key) – so if you get something signed by an identity key, you know where the data came from…

TPM Keys
The Data in a TPM_KEY12 Structure

<table>
<thead>
<tr>
<th>TPM_STRUCTURE_TAG</th>
<th>tag</th>
<th>How can this key be used?</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPM_KEY_USAGE</td>
<td>keyUsage</td>
<td></td>
</tr>
<tr>
<td>TPM_KEY_FLAGS</td>
<td>keyFlags</td>
<td></td>
</tr>
<tr>
<td>TPM_AUTH_DATA_USAGE</td>
<td>authDataUsage</td>
<td></td>
</tr>
<tr>
<td>TPM_KEY_PARAMS</td>
<td>algorithmParms</td>
<td></td>
</tr>
<tr>
<td>TPM_STORE_PUBKEY</td>
<td>pubKey</td>
<td></td>
</tr>
<tr>
<td>TPM_ENC_SCHEME</td>
<td>encScheme</td>
<td></td>
</tr>
<tr>
<td>TPM_SIG_SCHEME</td>
<td>sigScheme</td>
<td></td>
</tr>
<tr>
<td>TPM_ALGORITHM_ID</td>
<td>algorithmID</td>
<td></td>
</tr>
</tbody>
</table>

Ex: TPM_ALG_RSA
Ex: TPM_ES_RSAESOAPKCS15_SHA1_DER
Ex: TPM_ES_RSAESOAPKCS15_SHA1
Ex: TPM_ES_RSAESOAPKCS15_SHA1_MGF1
Ex: TPM_ES_RSAESOAPKCS15
Ex: TPM_ES_RSAESOAPKCS15
Ex: TPM_ES_RSAESOAPKCS15_DER
Ex: TPM_ES_RSAESOAPKCS15
Ex: TPM_ES_RSAESOAPKCS15
Ex: TPM_ES_RSAESOAPKCS15

Algorithm-specific
TPM Keys
The Data in a TPM_KEY12 Structure

Example parm structure for the RSA algorithm:

- `UINT32 keyLength`
- `UINT32 numPrimes`
- `UINT32 exponentSize`
- `BYTE[] exponent`

Public exponent – use 0 for “standard exponent” (65,537)

TPM Keys
Are you paying attention?

- `AG`
- `tag`
- `fill`
- `keyUsage`
- `keyFlags`
- `SAGE`
- `authDataUsage`
- `algorithmParms`
- `PCRInfoSize`
- `PCRInfo`
- `pubKey`
- `encDataSize`
- `encData`

Very sensitive info – how key can be used, can it migrate, does it need authorization…
What stops an attacker from simply changing this?

TPM Keys
Hierarchy

- `Storage Root Key (SRK)`
- `TPM_KEY_STORAGE`
- `More keys…`
- `Key1`
- `TPM_KEY_SIGNING`
- `Key2`
- `TPM_KEY_STORAGE`
- `migratable`
- `Key3`
- `TPM_STORE_PRIVKEY`
- `more keys…`

Basic idea: Storage keys protect other keys, and non-migratable keys can only be under other non-migratable keys…

TPM Keys
Typical Key Structures: Migratable Multi-User Hierarchy

As a result:
- Owner cannot migrate user keys directly
- Parent of user keys is non-migratable SRK, so can’t be migrated that way

TPM Keys
Typical Key Structures: Migratable Multi-User Hierarchy

- `Storage Root Key (SRK)`
- `TPM_KEY_STORAGE`
- `user base keys with migration auth unknown to administrator/owner`
TPM Keys

Typical Key Structures: Nonmigratable Keys

- **Storage Root Key (SRK)**
- **AIK 1**
- **AIK 2**

**Properties:**
- All keys can be certified to be usable only by the TPM
- Strong tie to this particular platform

**Down-side:**
- Can’t be backed up or migrated in case of machine failure/upgrade (although “maintenance” mode a possibility)

**Properties:**
- Book shows AIK under user non-migratable keys – that’s wrong!

**Design Goal 3**

Securely identify user and system – Creating an identity (simplified)

1. Check Cert(EKpub)
2. Pick random sess key S
3. Create/Sign Cert(AIK)
4. B = E(S, Cert(AIK))
5. X = E(EKpub, <S, AIK>)

Key points:
- Only a legitimate TPM can decrypt X
- Will only allow decryption of Cert(AIK) if it really is one of our AIKs

As a result:
- AIK certificates prove that the AIK is bound to a TPM

**TPM Keys**

Is a non-migratable key really tied to a TPM?

- Already talked about modifying the migratable flag
- Since parent key must be non-migratable it is tied to this TPM (induction hypothesis!), so can only be loaded on this TPM
- Final concern: Can we create a key externally (so we know the secret key) and create the TPM_KEY12 marked “non-migratable” ourselves?
  - No: This is one role for the tpmProof secret (stored in migrationAuth)

**TPM Keys**

How is a key made ready for use?

- **TPM_LoadKey** does this (simplified version):
  - Is specified parent key a TPM_KEY_STORAGE?
  - Are we authorized to use the parent key?
  - Decrypt encData using parent key
  - Check pubDataDigest for authenticity of public data
  - Is authentication required?
    - If yes, match provided secret with decrypted usageAuth
  - If key is non-migratable, is migrationAuth = tpmProof?
  - Are PCRs valid?

**TPM_ActivateIdentity(X)**

Decrypts X with EKsec,
and if AIK is one of our AIKs, release S
 Decrypt B using S → Now Alice has Cert(AIK)

Key points:
- Only a legitimate TPM can decrypt X
- Will only allow decryption of Cert(AIK) if it really is one of our AIKs

As a result:
- AIK certificates prove that the AIK is bound to a TPM

**Design Goal 5**

Support and isolate multiple users

- One argument for not being able to get SRK private key
  - If SRK private key were known, entire storage tree could be decrypted
  - More politically correct than "you can’t get it because we don’t trust you, the owner of the machine"

- Keys further down in the storage hierarchy have individual authorization secrets (set when the key is created)
  - No "superuser access" that can access all keys (outside TPM)
  - Can a rootkit capture user’s keystrokes entering passphrase?
    - Theoretically the integrity protection can stop this (no rootkits!)
    - Future plans include hardware “trusted path” (encrypted keyboard so only encrypted data can be sniffed)

**Additional TPM Capabilities**

Secure (Pseudo) Random Numbers

- Secure random/pseudo-random numbers are important for many security protocols (session keys, etc.)
- Examples of bad “random” numbers in protocols:
  - Online blackjack game with non-cryptographic PRNG
  - SSL session key derived from small seed (date and PID)
- A standard, dependable, secure PRNG is very useful
- Then the book talks about using the TPM random generation for things like Monte Carlo simulation:
  - This is completely silly – no need for “security”, just uniformity, and CPU can generate a good uniform sequence much faster than the TPM
Some New Capabilities of Version 1.2

• Certifiable Migratable Keys (CMKs)
  – Something in between 1.1 migratable and non-migratable
  – Committed to certain migration authorities (MAs) when key created
  – Certificate then says: This key is under the control of these MAs

• Monotonic Counters
  – State maintained across reboots and power cycles
  – Counters can be incremented and don’t wrap – values don’t repeat

• Direct Anonymous Attestation
  – A (much) more complex way of authenticating an AIK
  – Does not reveal AIK even to PrivacyCA

• Delegation of Owner-Authorized Commands