Programming for the TPM
and other practical topics

Ariel Segall
ariels@alum.mit.edu

Day 2

Approved for Public Release: 12-2749
Distribution unlimited
License

All materials are licensed under a Creative Commons “Share Alike” license.

- http://creativecommons.org/licenses/by-sa/3.0

You are free:

- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work
- to make commercial use of the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
FAQ: “Where can I buy commercial products with TPM feature X?”
Usual Answer: “You can’t yet”.

- Software vendors have generally not started integrating TPMs
  - Demand not there
- Only a few isolated products provide support
  - Usually specialized
  - Not always friendly to other applications
- For today, expect to build your own, or convince vendor to support
The Exceptions

- Most mature: trusted computing work in open source community
  - Largely driven by IBM, European OpenTC initiative, grad students
  - tpm-tools: Linux package (orphaned?) with basic command line utilities
  - Thunderbird integration: TPM protection of key store
  - tboot: GRUB (boot loader) version with extra TPM compatibility, features
  - Generally aimed at individual tinkerers

- Microsoft beginning TPM integration
  - Prominently: Bitlocker drive encryption
  - Automatic provisioning tools, but do not use
    - Do not meet security recommendations
    - Reports of incompatibility with anything but Bitlocker

- Wave Software does enterprise TPM integration
Two primary approaches:

- **Trusted Software Stack**
  - “High-level” (C) API for TPM; back end handles some complexity
  - TrouSerS on Linux

- **Driver-level coding**
  - Byte arrays for TPM’s direct consumption, or close to it
  - Microsoft’s Trusted Base Services
  - Flicker
Advantages of Each Approach

TSS:
- C API allows integration at many applications’ level
- Manages authorization sessions, keys for you
- Book about how to use it!

Driver-level:
- TPM spec (while complicated) relatively well-defined
- Very clean if comfortable working at low level
- For simple applications, much lower overhead
Downsides to Each Approach

TSS:
- Spec is *even more complicated* than TPM, and less well-written
- Multiple abstraction levels, unclear how to use
- High overhead for even simple applications
- Debugging extremely difficult

Driver-level:
- Managing nonces and authorization sessions complicated and fragile
- Lower-level than many applications
- Difficult to read and debug unless driver or kernel programmer
- Only documentation is TPM spec
- Debugging extremely difficult
Drilling Down (Slightly)

- TSS
- Driver-level

Note: Either of these could be a multi-day course on its own!
Spec from TCG; intended to be standard interface to TPM

“TSS” really refers to two pieces:
- API for coding for the TSS
- Back-end driver which exports API, handles TPM communications

Working implementations:
- TrouSerS (Linux; buggy port to Windows 7)
  - trousers package in most standard Linux distributions
- NTRU stack (Windows XP; port to Windows 7 not yet well tested)
  - ...neither perfect, but fairly reliable
What the TSS Does For You

- **Authorization Sessions**
  - Associate passwords with keys, other resources
  - In some implementations, secure password input

- **Basic Key Management**
  - Swap keys out when TPM out of space
    - Rarely necessary feature today
  - In some implementations, stores created keys in internal store
result = Tspi_Context_Create( &hContext);
result = Tspi_Context_Connect(hContext, NULL);
// Get the TPM handle
result = Tspi_Context_GetTpmObject(hContext, hTPM);
// Get the SRK handle
result = Tspi_Context_LoadKeyByUUID(hContext, TSS_PS_TYPE_SYSTEM, SRK_UUID, &hSRK);
// Get the SRK policy
result = Tspi_GetPolicyObject(hSRK, TSS_POLICY_USAGE, &hSRKPolicy);
// Then set the SRK policy to be the well known secret
result = Tspi_Policy_SetSecret(hSRKPolicy, TSS_SECRET_MODE_SHA1, 20, wks);
result = Tspi_Context_CreateObject(hContext, TSS_OBJECT_TYPE_RSAKEY, initFlags, &hESS_Bind_Key);
result = Tspi_Key_CreateKey(hESS_Bind_Key, hSRK, 0);
Resources exist!

Dave Challener (author of much of TSS spec) wrote book: 
*A Practical Guide to Trusted Computing*

Has also taught short workshops whose materials are online
- On your quick reference sheet
Drilling Down

- TSS
- Driver-level
Driver Level Variations

Nothing so coordinated as TSS standard!

- Used when in extremely minimal environments
  - Flicker: running in CPU secure mode, stripped down
- Windows 7 native support: TBS
  - TBS is (theoretically) a direct pass-through to TPM
  - **TBS modifies code unpredictably! Serious problem.**
- Homebrew your own driver!
Assemble your data structures, based on TPM structures spec
Assemble your command blob, based on TPM command spec
Send to TPM
Deconstruct response blob, based on TPM command spec
Deconstruct relevant data structures, based on TPM structures spec
Interpret and use as needed
Driver-Level Code: Example

```c
int slb_TPM_Extend(unsigned char *buffer,
unsigned long pcrindex, unsigned char *hash){
int res;
((unsigned int *)buffer)[0] = 0x0000c100;
((unsigned int *)buffer)[1] = 0x00002200; /* length = 34 */
((unsigned int *)buffer)[2] = 0x00001400;
*(((unsigned int *) (buffer+10)))=ntohl(pcrindex);
TPM_COPY_TO(hash, 4, TCG_HASH_SIZE);
res = slb_tis_transmit(buffer, 34, TCG_BUFFER_SIZE,
TIS_LOCALITY_2);
TPM_COPY_FROM(hash, 0, TCG_HASH_SIZE);
return res < 0 ? res : (int) ntohl(*(((unsigned int *)
(buffer+6))));
}
```

Code copyright Jon McCune and Bernhard Kauer, released under GPL 2
Adoption severely slowed by difficulty of use

The vast majority of applications use a fraction of TPM commands

No need for full range of options

• Just build in recommended choices—advanced users can brew their own

Make conceptually atomic actions take one command

• Hide key handling; hide authorization sessions; hide intermediate steps

Use TPM at the level people understand it

Straightforward project; just needs someone to do it.
No good choices today!
- TSS overcomplicated and high overhead
- Driver-level API overcomplicated, extremely low-level

Support architectures exist, but not universally
- Windows support particularly patchy

Lots of room for improvement, and vendors
Questions?