presented by

Evolving Hardware-Based Security: Firmware Transition to TPM 2.0

UEFI Summerfest – July 15-19, 2013

Dick Wilkins, Ph.D.

Phoenix Technologies, Ltd.

Agenda

- Introduction
- Background
- Why TPM 1.2 to 2.0
- Differences
- Phoenix's model
- Wrap-up & Questions

Introduction

- The Trusted Computing Group upgraded their Trusted Platform Module spec from 1.2 to 2.0. We're going to talk about:
  - Briefly, what is a TPM?
  - What's different in 2.0
  - What does this mean to firmware (BIOS)?
- **Note:** UEFI does not require TPM and only mentions TPM/TCG as an "FYI" external reference in the UEFI spec
  - So why do we care?

Background

- A TPM is:
  - Tamper-resistant functionality, state and operations (hardware and/or software)
  - Protected storage for keys and certificates
  - Platform Configuration Registers (PCRs)
  - Cryptographic engine
  - And more

PCRs

- Cannot be written directly
  - *extend(i, v) := pcr[i] ← hash(pcr[i], v)*
- Extending PCRs with hashes of code and data during boot can be compared to previous boots
- Firmware created log entries allow detection of "where things went wrong"
- This approach allows "*measured*" and "*authenticated*" boot
- This is not the same as "secure boot" as enabled by the UEFI spec

Sealed Storage

- Sealing uses TPM cryptographic support with PCRs to provide secure storage
  - "*sealing*" provides a key, a set of PCR values and some data
  - The result of sealing is a "*blob*" of data
    - That can only be unsealed by the TPM that sealed it
    - Can only be unsealed if the current PCRs match those used to seal the data

TPM 2.0 algorithm flexibility

TPM 1.2

- Support for three algorithms
  - SHA1 – hash
  - RSA – asymmetric
  - XOR – symmetric
  - AES is allowed in limited cases

TPM 2.0

- Support for:
  - Any hash algorithm with a fixed digest size
  - Any asymmetric algorithm that has a public and private portion
  - Any symmetric algorithm

Why is this important?

- SHA1 is considered unsuitable for future use
- Just changing to another hash algorithm was not a long term solution
- Regional differences require that a single asymmetric solution was not acceptable
  - USA – Suite B
  - China – 'Suite C' (SM3, SMS4, 256-bit ECC curve)
  - Russia – GOST
  - Germany – Brainpool
  - expect this list to grow

What else?

- A long list of additional functionality requested by users
- A list of little used and deprecated functionality to be removed
- Resolved confusing TPM enablement, activation and ownership (solved largely with later client interface specs)

Relatively Easy Transition

- Same command/response paradigm
- Very similar command format
- 1 to 1 relationship between many old and new commands

  TPM_* ~= TPM2_*

Simplifies Usage

- Removed the confusing enabled/activated/disabled/deactivated states
  - It's either there or not there (ACPI table)
  - If present, it can be used by firmware even if not exposed to the OS
- The end user meets fewer prompts
  - Only required to authorize TPM clear

Differences examples

- With removal of enabled/activated states
  - The TPM no longer tracks Physical Presence states internally that firmware must manage
- TPM clear is implemented – with appropriate Physical Presence – by:
  - a TPM2_ClearControl command
  - And then TPM2_Clear

Second example

- Extend difference
  - For TPM 1.2, a PCR Extend includes only the hash digest value
  - For TPM 2.0, an Extend includes a list of one or more hash digests with algorithm identifiers
    - Intended to allow Extends of more than one bank of PCRs

Phoenix's Implementation

- One driver supports 1.2 and 2.0 TPMs
  - If 2.0 is not detected, fall back to 1.2
- A low-level communication protocol abstracts the device
  - Hardware or firmware TPMs appear identical to the driver
- Our understanding is Windows 8 has a similar TPM abstraction for applications

Closing Remarks

Questions?

Thanks for attending the UEFI Summerfest 2013

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by