CE Linux Forum

Trusted Boot Loader

Steve Johnson, Panasonic

Chair Security WG

San Jose

April 12, 2006

Synopsis

- Background
- Trusted boot
- Security enhancements to boot loader
- Necessary code
- U-Boot
- Kernel authenticity
- Secure U-Boot
- Conclusions

Background

- Trusted Computing Platform Alliance / Trusted Computing Group – TCPA / TCG
- Trusted Computing
- Trusted Platform Module – TPM

TCG

- Develops, defines, and promotes open standards for hardware-enabled trusted computing and security technologies
  - hardware building blocks
  - software interfaces
  - multiple platforms, peripherals, and devices.
- Primary goal is to protect user's information assets (data, passwords, keys, etc.) from compromise due to external software attack and physical theft.

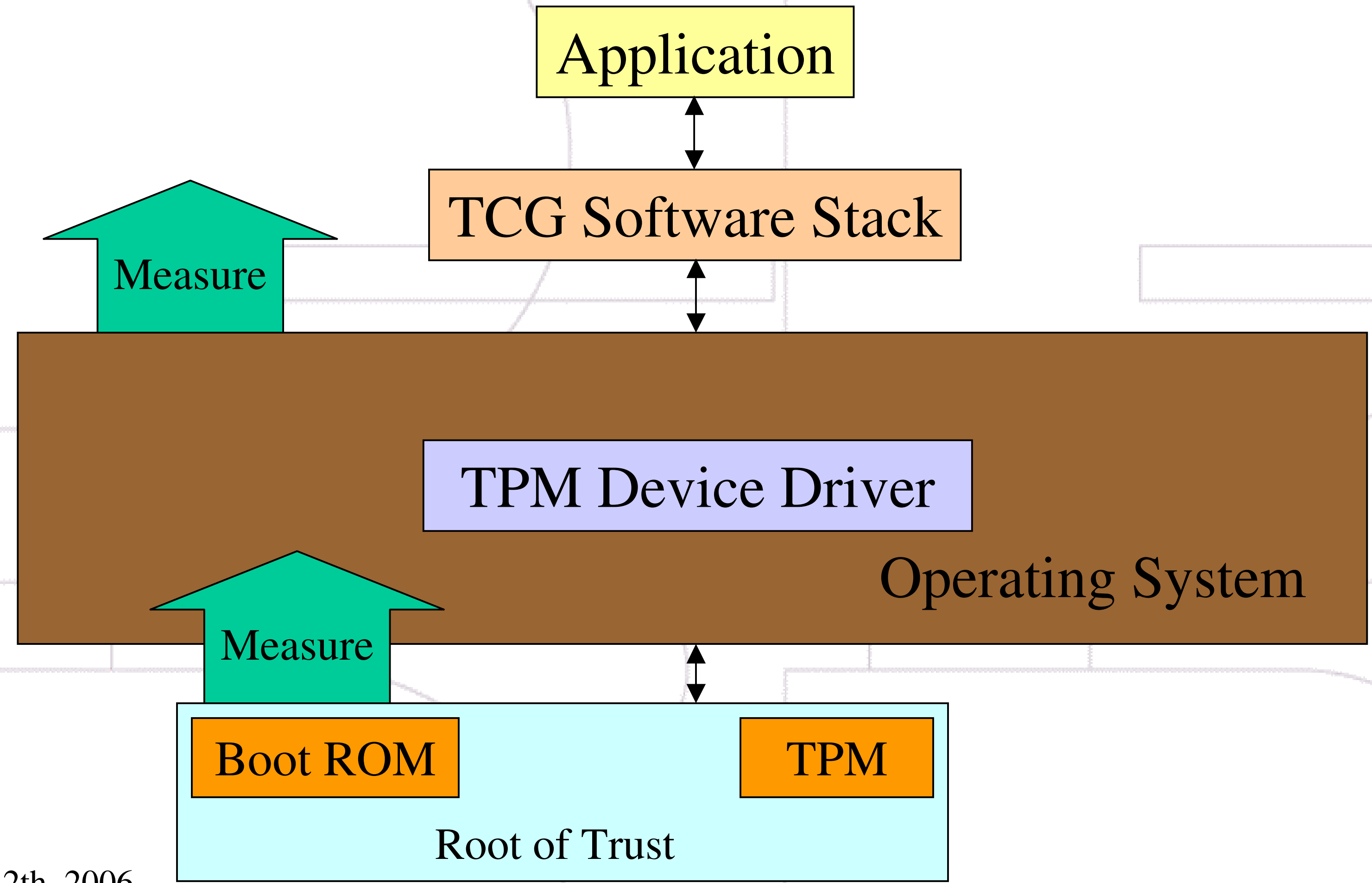
Trust and Trusted Computing

- What is trust?
  - The expectation that a device will behave in a particular manner for a specific purpose
  - System you are forced to trust vs. one that is trustworthy
- What is trusted computing?
  - Technology developed and promoted by the Trusted Computing Group (TCG)

Trusted Computing

- Machine specific public and private keys and certificate chain
- Cryptographic functionality
- Data can be signed with the machine's identification
- Data can be encrypted with the machine's secret key

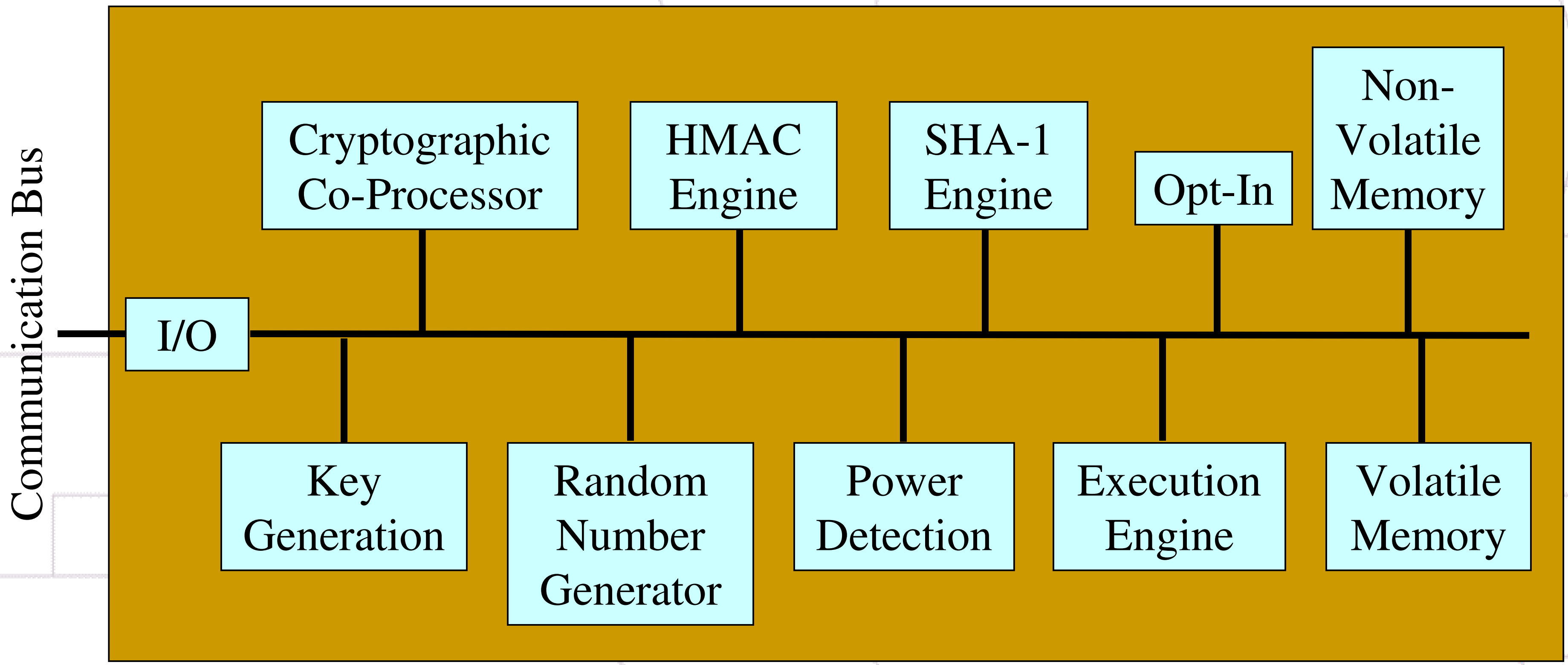
TCG components



TPM activities

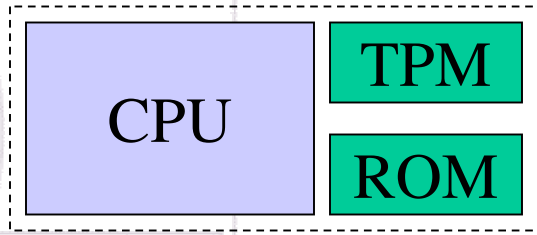
- Boot loader measures boot through kernel and initrd
- Initrd has TPM unseal kernel master key
- If a match, TPM releases kernel master key
- Key used to generate keys for further stages
- If measurements don't match, boot is halted

TPM major components

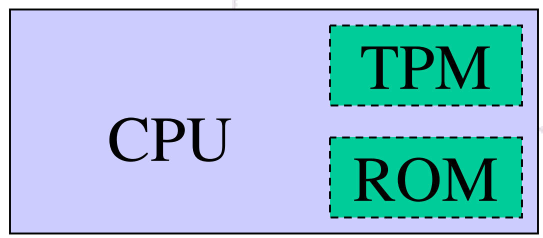


Necessary TPM hardware

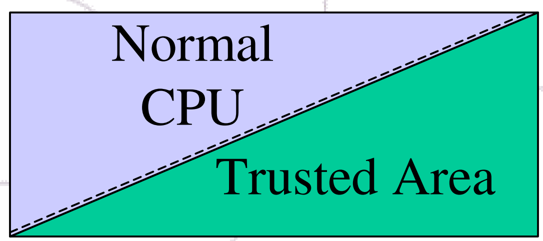
Discrete TPM



Embedded TPM



Software TPM



Trusted boot

- Trusted boot loader
- Secure boot loader

Security levels for boot loader

	Security Features					Ease of Management
	Software			Hardware		
	CRC ECC	Hash	Signa ture	Write Protected Bootloader	TPM	
Normal Boot	O	-	-	-	-	Easy, but no protection
Secure Boot (by digest)		O		Root of Trust (Reference Value)		Bad
Secure Boot (by signature)		O	O	Root of Trust (Signer's public key)		Good + Easy to update OS image without modifying Bootloader
Trusted Boot		O		Root of Trust	Root of Trust (Secure Storage)	Good (for connected device) + Device Authentication + Integrity Protection + Integrity Report

Security enhancements

- Simple integrity check
  - Error checks and recovery
- Secure boot
  - Ensure secure initial state
  - Ensure only an un-tampered system is run
- Trusted/authenticated boot
  - Ensure a secure initial state
  - Ensure only an un-tampered system is run
  - Measure and report

Trusted boot

- Each boot step is measured and stored
- A sequence of measured values (stored measurement log)
- Executable code and associated information could be measured before it is executed

GRUB booting

- Stage 1
  - Initialization
  - Detect geometry of “loading drive”
  - Load the first sector of Stage 1.5
  - Jump to start of Stage 1.5
- Stage 1.5
  - Load the rest of Stage 1.5
  - Jump to the starting address
  - Load Stage 2
  - Jump to start of Stage 2

GRUB booting

- Stage 2
  - Load kernel
  - Jump to kernel start

Trusted GRUB booting

- Stage 1 measures stage 1.5 after loading it
- Stage 1.5 measures stage 2 after loading it
- Stage 2 measures stage 1.5
- Stage 2 measures kernel

Required code and components

- Boot loader
  - Crypto functions
  - Hash
  - Asymmetric cipher (RSA)
- Hardware
  - Write protected initial boot code ROM
  - Flash memory with boot block protection
  - TPM

U-Boot

- Open source firmware for embedded
  - PowerPC, ARM, MIPS, x86, …
- Command line
  - Information commands
  - Memory commands
  - Flash memory commands
  - Execution commands
  - Download
  - Environment variables
  - Special
  - Miscellaneous

U-Boot boot process

- Invoke U-Boot
- Starts running from ROM
- Relocates itself to RAM
- Initial setup and environment checks
- Locate the kernel and decompress it
- Check CRC of kernel
- Transfer control to kernel image
- Kernel boots

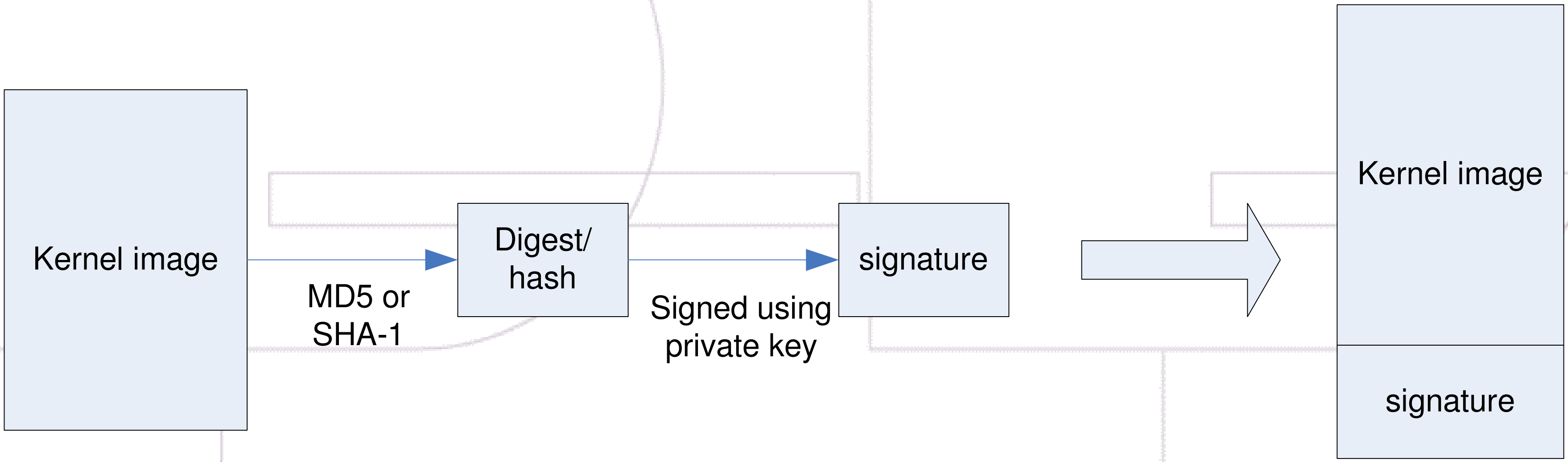
U-Boot security

- Only knows CRC
- Basically a sophisticated checksum
- CRC good for finding random errors in a transmission
- Little protection against malicious attacks

Signed kernel

- Hash calculated from kernel binary
  - MD5 or SHA-1
  - Use private key of public/private key pair to encrypt digest
- Signature appended to kernel image as meta-data

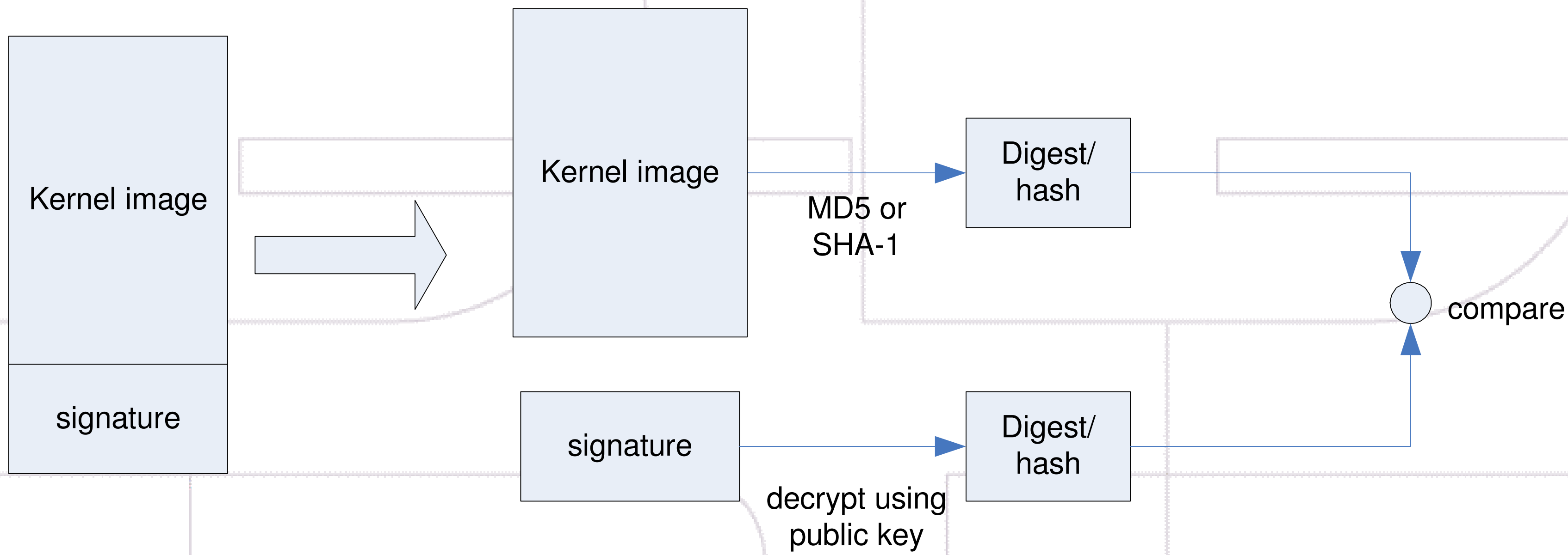
Signed kernel



Kernel image authenticity

- Boot loader decompresses kernel image and meta-data
- Signature is extracted and decrypted using public key
- Hash is calculated from kernel image
- If signature matches hash, the kernel image is authentic

Signed kernel with u-boot



Secure U-Boot process

- Invoke u-boot
- Starts running from ROM
- Relocates itself to RAM
- Initial setup and environment checks
- Locate the kernel and decompress it
- Check CRC of kernel
- Authenticate kernel
- Transfer control to kernel image
- Kernel boots

U-Boot booting process

- Preliminary setup
  - CPU
  - Memory
- Relocate self to RAM
- Initialize ARM boot
  - Flash
  - Environment
  - IP & MAC address

U-Boot booting

- Initialize ARM boot (continued)
  - Devices
  - Console
  - Interrupts
  - Ethernet
- Boot kernel
  - Read image header
  - Decompress image
  - Transfer control to kernel

Required modifications

- Identify appropriate places in u-boot for modifications
  - Between decompress image and transfer control to kernel
- Add hash code
- Add encryption/decryption code
- Add key handling

Hardware based protection

- Not striving for full TCG compliance
- "Secure" boot loader is sufficient for first step
- Where to store stuff?

CE Linux Forum

U-Boot start

```
U-Boot 1.1.4 (Mar 29 2006 - 10:01:55)

DRAM:  32 MB
Flash: 32 MB
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
OMAP1510 Innovator #
```

Innovator flash

```
.
.
.
OMAP flash: using static partition definition
Creating 5 MTD partitions on "omap-flash":
0x00000000-0x00020000 : "BootLoader"
0x00020000-0x00060000 : "Params"
0x00060000-0x00260000 : "Kernel"
0x00260000-0x01000000 : "Flash0 FileSys"
0x01000000-0x02000000 : "Flash1 FileSys"
```

U-Boot parameters

- 256K total
- Room for key information

Roadmap

- Verify boot image
- Hardware based protection
  - Protection of ROM, boot block, flash memory
- Complete TCG trusted boot
  - Need TPM
  - TPM driver
  - TPM initialization
  - TPM APIs (Library)
  - Integrate boot image verification and boot loader protection

Conclusions

- Secure boot is needed
- Trusted boot exists for BIOS based systems with TPM
- Not a lot required for “secure” boot for embedded systems

Links

- U-Boot
  - Documentation
    - http://www.denx.de/wiki/DULG/Manual
  - Project home page
    - http://sourceforge.net/projects/u-boot
- TCG
  - https://www.trustedcomputinggroup.org/home
- TPM
  - https://www.trustedcomputinggroup.org/groups/tpm/

Links

- TPM device driver for Linux
  – http://sourceforge.net/projects/tpmdd
- TCG Software Stack implementation
  – http://sourceforge.net/projects/trousers
- TCG patch for GRUB
  – http://trousers.sourceforge.net/grub.html