

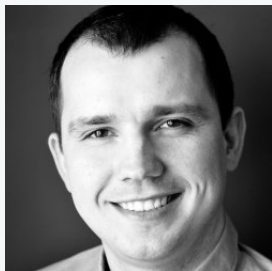
# Enabling TPM 2.0 on coreboot based devices

European Coreboot Conference 2017


Piotr Król and Kamil Wcisło








Piotr Król

-  @pietrushnic
-  piotr.krol@3mdeb.com
-  [linkedin.com/in/krolpiotr](https://www.linkedin.com/in/krolpiotr)
-  [facebook.com/piotr.krol.756859](https://www.facebook.com/piotr.krol.756859)



Kamil Wcisło

-  @mek\_xgt
-  kamil.wcislo@3mdeb.com
-  [linkedin.com/in/kamil-wcislo-86a83189](https://www.linkedin.com/in/kamil-wcislo-86a83189)
-  [facebook.com/mek.xgt](https://www.facebook.com/mek.xgt)

## Why?

- security is important
- in times of IoT devices have to be hardened against attacks
- devices have to have some kind of proof they run good software

## Why?

- security is important
- in times of IoT devices have to be hardened against attacks
- devices have to have some kind of proof they run good software

## What we want to show?

- some of our struggles and conclusions when trying to enable secure boot in open firmware device

## Why?

- security is important
- in times of IoT devices have to be hardened against attacks
- devices have to have some kind of proof they run good software

## What we want to show?

- some of our struggles and conclusions when trying to enable secure boot in open firmware device

## Some comments...

- this is still WIP, there are many things to be done
- we're just the beginners in this area

- Security concepts overview
- Introduction to TPM 2.0
- TPM hardware
- State of support in UEFI and SeaBIOS
- State of Linux kernel and user space tools
- Verified boot in coreboot
- Conclusions

### The CIA triad

- Confidentiality
- Integrity
- Authorization



### The CIA triad

- Confidentiality
- Integrity
- Authorization

### and...

- Availability
- Non-repudation

- cryptochip for boot verification, platform verification, disk encryption, etc.
- slow by design
- contains cryptographic functions, hashing algorithms, key generation and protection mechanisms, RNG, integrity measurement mechanisms
- talks with the host platform using various interfaces (LPC, I2C, SPI)
- PCRs
  - shielded locations
  - modifiable only by using `PCRExtend` function
  - resettable only by rebooting system
- has secure storage
- it's pasive!
- has possibility to restrict access to certain keys, if measurements don't match

## secure boot vs measured boot

### secure boot

- uses cryptographic functions
- secure boot prevents booting other fw
- needs some kind of root of trust
- asserts the boot

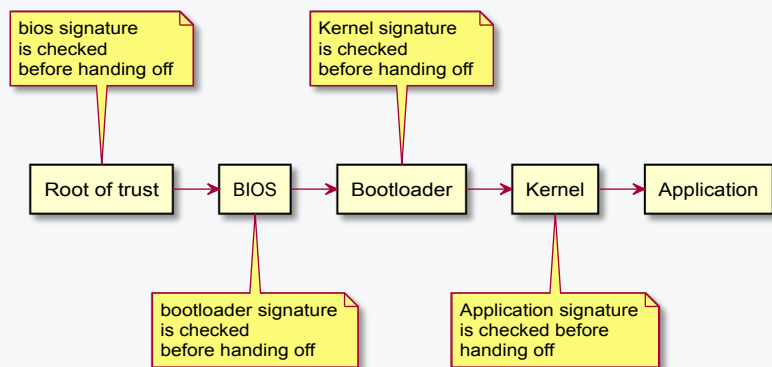
## secure boot vs measured boot

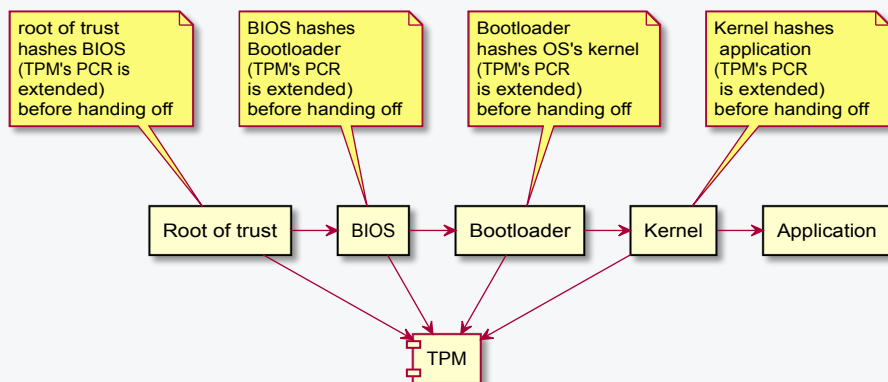
### secure boot

- uses cryptographic functions
- secure boot prevents booting other fw
- needs some kind of root of trust
- asserts the boot

### measured (trusted) boot

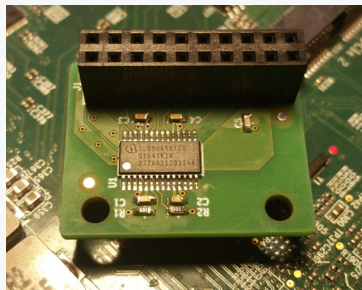
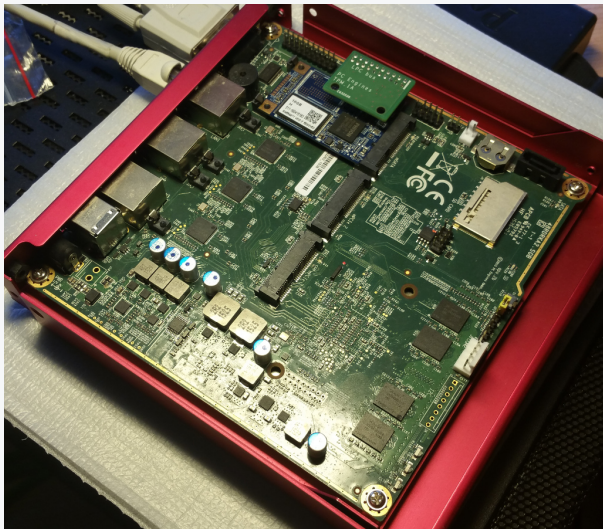
- uses hashes (cryptographic hashes)
- measured boot notifies about compromised fw
- TPM is needed (extend PCRs) to be secure
- proves the boot
- could be used to certify that system is in expected state





- TPM is international standard for a secure microprocessor
- There are many hw implementations
- TPM 2.0 is not backwards compatible
- Differences
  - supported algos
  - private keys vs. primary "seeds" and KDF
  - API

- We're using PC Engines APU2 board as a reference platform
- TPM is Infineon SLB9665 chip with LPC interface





- UEFI was first to have TPM2.0 support
- coreboot supports TPM 2.0 (                      &&                      )
  - drivers for devices are in
  - TPM API is in
- SeaBIOS also supports TPM2.0 (                      )
  - drivers are in
  - API (TCGBIOS) is in

- support for TPM 2.0 in kernel added in 4.0 and improved in 4.4
  - kernel module in our case
  - resource manager added in 4.12
- 3 userspace stacks available
  - (Intel) <https://github.com/01org/tpm2-tss.git>
  - (IBM) <https://sourceforge.net/projects/ibmtpm20tss/>
  - (Google) [https://chromium.googlesource.com/chromiumos/third\\_party/tpm2/](https://chromium.googlesource.com/chromiumos/third_party/tpm2/)

After changing the board's `bootloader` to select `linux` and board booted, but Linux had problems initializing the module.

In Linux, we had to insert the `tpm` module manually:

```
insmod /lib/modules/$(uname -r)/kernel/drivers/char/tpm/tpm.o
```

After this, the device was somewhat usable, but not in the full potential.

`tpm` and `tpm2.0` (Intel's TPM2.0 stack) was able to communicate with the device.

It's seems that TPM is not getting initialized correctly in the firmware.

After checking the debug output, it was obvious our TPM chip was detected as the wrong one:

It detected the chip as the TPM1.2 compliant one and the startup sequence returned errors.

TPM2.0 has different startup sequence than 1.2.

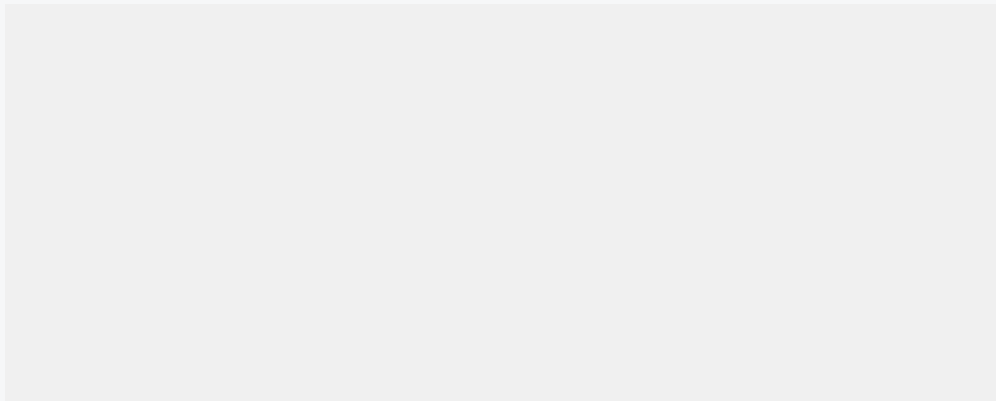
TPM2.0 spec is not easily readable, despite having code examples.

In 4.13, Linux has quite good support for TPM 2.0.

Also contains TPM 2.0 starting sequence.

is the pretty good source of knowledge now.

Because Linux is trying to fix firmware shortcomings, it's got startup procedure (i.e. "manual startup" used when inserting the tpm module with `tpmreset`). E.g.:



- It seems that SLB9665 has the same VID/DID as SLB9660
- Added the conditional to identify the chip as SLB9665, when using TPM2.0
- Added additional conditional to use startup sequence for TPM2.0

Done in `tpm2-udev-rules`. It seems this driver has support only for TPM1.2.

Now Linux inserts the `tpm2-udev-rules` module automatically.





There is TPM2.0 driver in

As far as I understand it, it's used exclusively in , at least for now.

Maybe this could be leveraged to remove redundancy with drivers? (Maybe even that's the plan... ;-))

SeaBIOS also doesn't detect TPM2.0 chip correctly.

After hacking the `tpm20_detect` to return TPM2.0 version, we got TPM menu in SeaBIOS's bootmenu. We can clear the TPM there.

I can see that measurements are being done, e.g. functions like:

are used. They contain code to hash the target and extend the TPM.

- vboot
  - concepts
  - firmware image
  - usage in Chromebooks
- coreboot
- SeaBIOS/depthcharge

1. starts in RO part, where bootblock exists
2. verstage is where vboot is
3. control from bootblock is passed to verstage
4. vboot verifies rw section of firmware
5. if this fails, falls back to ro firmware and boots recovery

We wanted to have the verified booting capabilities.

It wasn't so trivial to implement.

vboot uses different flash layout to work:

- several CBFS regions: (RO) , , ,

Our platform uses hardware blobs.

- AGESA
- amdfw (PSP)

1. First we need to create FMD file (FMAP source)
  - need to place blobs in correct places
  - need to correctly place RO section (bootblock has to be at the end of flash)
  - takes a bit of hacking
2. Second need to add some boilerplate function to mainboard dir (e.g. write protect detection routine - . vboot requires them in order to detect that RO area is protected.
3. Third we have to enable vboot, . We have to point to our flash layout file.
4. Some hacking with blobs adding was required.







- enable verified boot capabilities in coreboot, SeaBIOS
- TrustedGRUB is there
- investigate the usage of vboot
- prove it works
- test, test, test...
- Is it possible to harden it even more? (what about the PSP?)

- Adding new chip in coreboot is quite easy. TPM support is there.
- Using the source brings better results than using the huge spec.
- Understanding the platform boot process and blob placement is crucial, when trying to enable vboot.
- One can have verified boot capabilities on custom coreboot platform.

- recent news - vulnerable RSA key generated by Infineon's hardware
- can decode RSA private key from public key
- if the RSA primes are generated as truly random numbers attack is not viable
- ECC keys are not affected
- key test tools available: [https://crocs.fi.muni.cz/public/papers/rsa\\_ccs17](https://crocs.fi.muni.cz/public/papers/rsa_ccs17)

- Linux and coreboot (obvious ;) source code
- <https://danielmiessler.com/study/infosecconcepts/>
- <https://forums.juniper.net/t5/Security-Now/What-s-the-Difference-between-Secure-Boot-and-Measured-Boot/ba-p/281251>
- <https://blog.hansenpartnership.com/tpm2-and-linux/>
- <https://firmwaresecurity.com/2016/01/15/seabios-gets-tpm2-security/>
- <https://www.coreboot.org/git-docs/Intel/vboot.html>
- [https://www.coreboot.org/images/f/f1/Tpm - Philipp.pdf](https://www.coreboot.org/images/f/f1/Tpm_-_Philipp.pdf)
- [https://elinux.org/images/6/6e/ELC2017 TPM2-and-TSS Tricca.pdf](https://elinux.org/images/6/6e/ELC2017_TPM2-and-TSS_Tricca.pdf)

Thanks!

# Thanks!

Q/A