Sekrety (nie) bezpieczeństwa TPM

Autor: Mateusz Lewczak

Agenda

- Czym jest i z czego się składa TPM?
- Jakie operacje można wykonywać z wykorzystaniem TPMa?
- Jak działa hierarchia kluczy w TPMie?
- Czym są i jaką funkcję posiadają banki PCR?
- Jak działają polityki dostępu do obiektów?

O mnie

- Penetration Tester @ SECURITUM
- Prelegent na konferencjach.

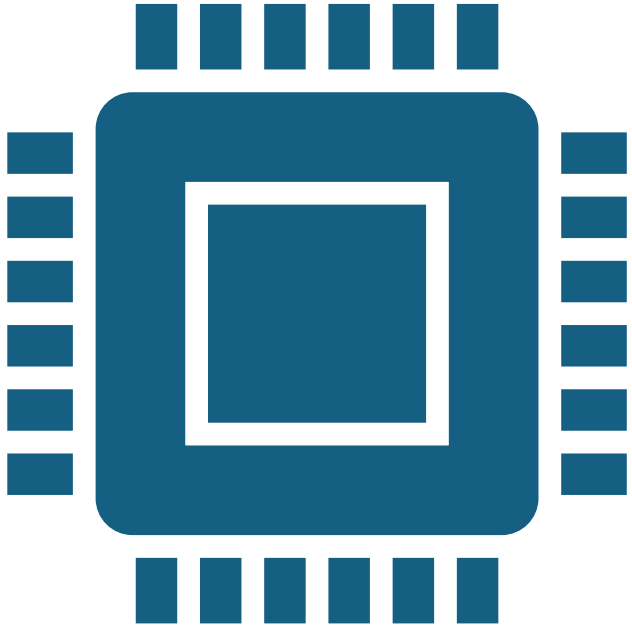
Czym jest Trusted Platform Module?

Rodzaje TPMów

- dTPM – discrete,
- fTPM – firmware,
- vTPM – virtual.

Z czego się składa?

- Procesora kryptograficznego.
- Bezpiecznej pamięci – ulotnej i nieulotnej.
- Generatora liczb losowych (RNG).
- Mechanizmu kontroli dostępu.



Jakie ma funkcje?

Generowanie i przechowywanie kluczy kryptograficznych.

Bezpieczne przechowywanie danych.

Weryfikacja integralności systemu.

Autoryzacja i uwierzytelnianie.

Kto odpowiada za specyfikację?

Czym nie jest?

- Zamiennikiem oprogramowania antywirusowego.
- Środkiem ochrony przed wszystkimi typami ataków.
- Systemem zarządzania uprawnieniami.
- Urządzeniem do szyfrowania całego dysku.

Podstawowe możliwości TPM’a

Operacje bez klucza

- Generowanie liczb losowych:

```
tpm2_getrandom <bytes_num>
```

- Liczenie hasha:

```
tpm2_hash -g sha256
```

Parametry TPM’a

- Wylistowanie grup parametrów:

```
tpm2_getcap -l
```

- Wypisanie danej grupy parametrów:

```
tpm2_getcap <cap_name>
```

Tworzenie kluczy

- Utworzenie klucza głównego (hierarchii):

```
tpm2_createprimary -c primary.ctx
```

- Utworzenie podklucza:

```
tpm2_create -C primary.ctx -c subkey.ctx
```

Rodzaje kluczy

Asymetryczne

ECC

RSA

Symetryczne

AES

Camellia

SM4

Keyed Hash

HMAC

XOR

Tworzenie kluczy cd.

- Utworzenie klucza głównego endorsment (hierarchii):

```
tpm2_createek -c ek.ctx
```

- Utworzenie klucza attestation:

```
tpm2_createak -C ek.ctx -G ecc -c ak_ecc.ctx
```

Operacje z kluczem

- Szyfrowanie:

```
tpm2_encryptdecrypt -c subkey.ctx -o msg.enc msg.txt
```

```
tpm2_rsaencrypt -c subkey.ctx -o msg.enc msg.txt
```

- Deszyfrowanie:

```
tpm2_encryptdecrypt -d -c subkey.ctx -o msg.enc msg.txt
```

```
tpm2_rsadecrypt -c subkey.ctx -o msg.ptext msg.enc
```

Operacje z kluczem cd.

- Podpis:

```
tpm2_sign -c subkey.ctx -g sha256 -o msg.sig msg.txt
```

- Weryfikacja podpisu:

```
tpm2_verifysignature -c subkey.ctx -g sha256 -m msg.txt
-s msg.sig
```

- HMAC hash:

```
tpm2_hmac -c hmac.key --hex data.in
```

Uproszczony przebieg utworzenia podklucza

- Utworzenie klucza głównego (hierarchii):

```
tpm2_createprimary -c p.ctx
```

- Utworzenie i załadowanie podklucza:

```
tpm2_create -C p.ctx -c sk.ctx
```

- Operacje na kluczu:

```
tpm2_sign -c sk.ctx -g sha256 -o msg.sig msg.txt
```

Ogólny przebieg utworzenia podklucza

- Utworzenie klucza głównego (hierarchii):

```
tpm2_createprimary -c p.ctx
```

- Utworzenie podklucza:

```
tpm2_create -C p.ctx -u key.pub -r key.priv
```

- Załadowanie klucza do TPM’a:

```
tpm2_load -C p.ctx -u key.pub -r key.priv -c lk.ctx
```

- Operacje na kluczu:

```
tpm2_sign -c lk.ctx -g sha256 -o msg.sig msg.txt
```

Blob’y

- Blob publiczny (Public Blob):
  - Jest to struktura danych, która przechowuje publiczną część klucza kryptograficznego w TPM.
- Blob prywatny (Private Blob):
  - Zawiera prywatną część klucza kryptograficznego i jest chroniony przez TPM.

Jak są chronione blob’y?

Obiekty, które mogą przemieszczać się poza TPM, muszą być chronione (poufność i integralność). Na przykład obiekty przejściowe (ang. transient) wymagają, aby dane chronione przez TPM (klucz lub pieczęć) były przechowywane poza TPM.

Jest to widoczne w narzędziach takich jak tpm2_create(1), gdzie opcja -r zwraca te chronione dane. Ten blob zawiera wrażliwe części obiektu. Wrażliwe części obiektu są chronione przez obiekt nadrzędny, wykorzystując szyfrowanie symetryczne rodzica i HMAC.

Źródło: https://github.com/tpm2-software/tpm2-tools/blob/master/man/common/protection-details.md

Wykorzystanie klucza głównego

- Utworzenie klucza głównego (hierarchii):

```
tpm2_createprimary -c p.ctx
```

- Operacje na kluczu:

```
tpm2_sign -c p.ctx -g sha256 -o msg.sig msg.txt
```

Zapieczętowanie sekretów

- Zapieczętowanie (ang. *seal*) danych:

```
tpm2_createprimary -c prim.ctx
```

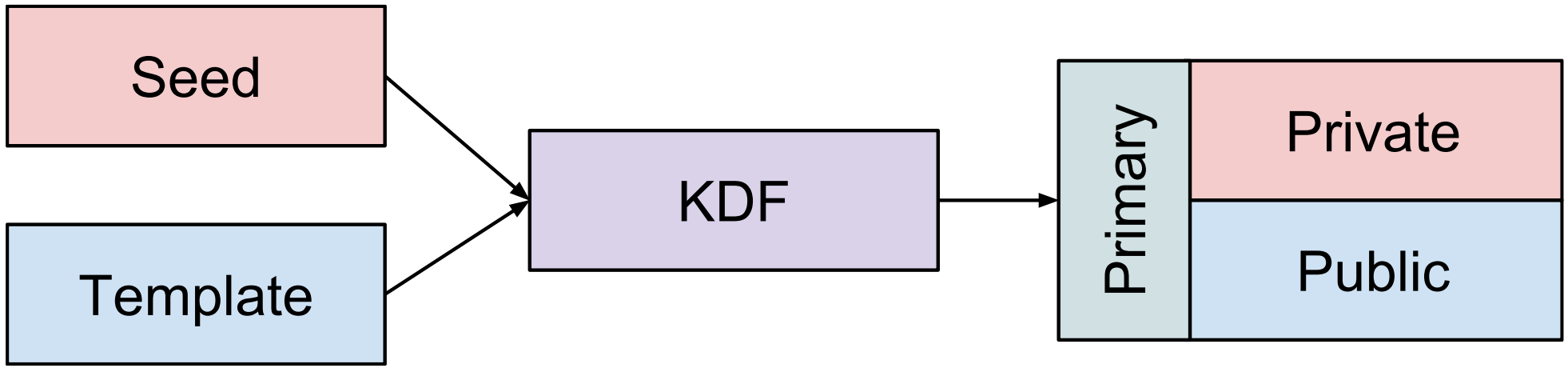
```
echo "secret" | tpm2_create -C prim.ctx -c seal.ctx -i-
```

- Odpieczętowanie (ang. *unseal*) danych:

```
tpm2_unseal -c seal.ctx
```

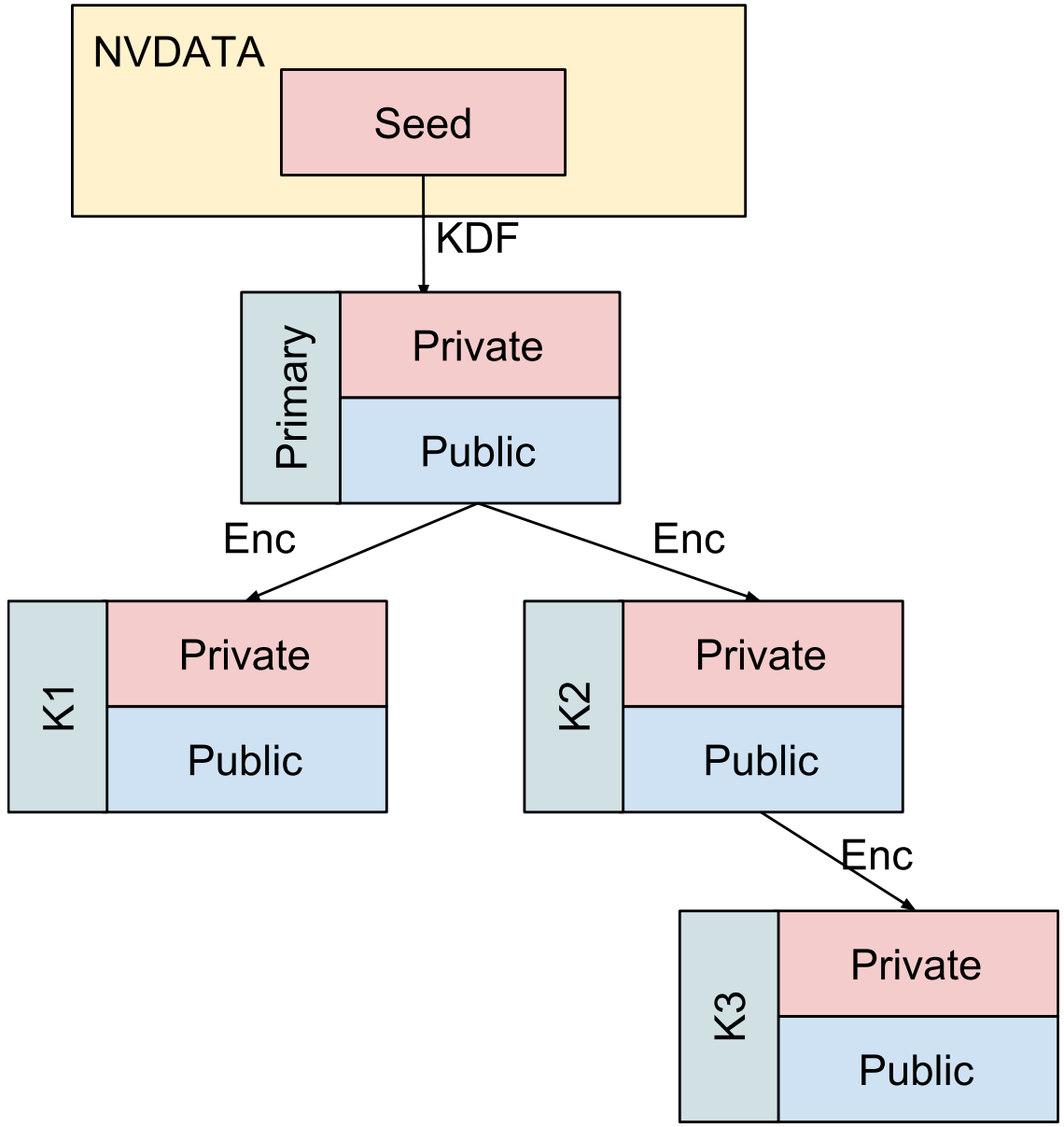
Zarządzanie kluczami

Klucze główne



Źródło: https://google.github.io/tpm-js/#pg_keys

Logiczne struktury kluczy



Źródło: https://google.github.io/tpm-js/#pg_keys

Przechowywanie kluczy

- Utworzenie podklucza powoduje eksport części publicznej i zaszyfrowanej części prywatnej.
- Klucz można zapisać w pamięci nieulotnej:

```
# tpm2_evictcontrol -c subkey.ctx
```

- `persistent-handle: 0x81000000`
- `action: persisted`

Rodzaje hierarchii

- Owner (a.k.a. Storage) Hierarchy:
  - Przeznaczony dla aplikacji, do dowolnego użytku.
  - W teorii seed może ulec zmianie (`tpm2_clear`).
- Platform Hierarchy:
  - Przeznaczony dla producentów sprzętu.
  - W teorii seed może ulec zmianie (`tpm2_changepps`).
- Endorsement Hierarchy:
  - Wykorzystywany w identyfikacji TPM’a.
  - W teorii seed może ulec zmianie (`tpm2_changeeps`).

Rodzaje hierarchii cd.

- Null hierarchy
  - Wykorzystywany do tymczasowego przechowywania kluczy.
  - Seed zmiania się przy każdym resecie.

Integralność maszyny – banki PCR

Banki PCR (Platform Configuration Register)

Rejestry wewnątrz układu.

Mogą być rozszerzane.

„Zapewniają” integralność środowiska.

Funkcje PCR

Pomiar i integralność

Tworzenie łańcucha zaufania

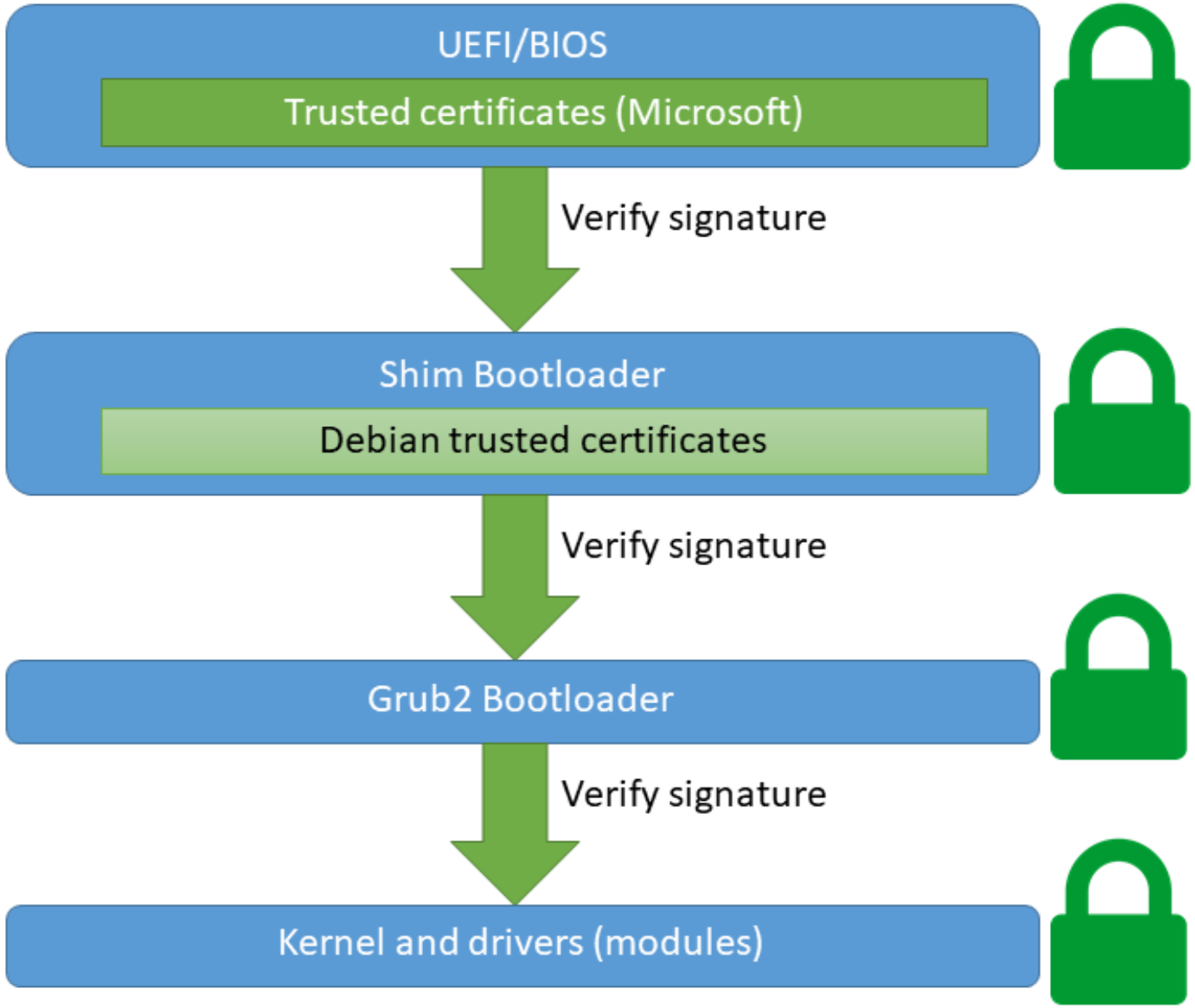
Autoryzacja dostępu

Pomiar (extend) w banku PCR

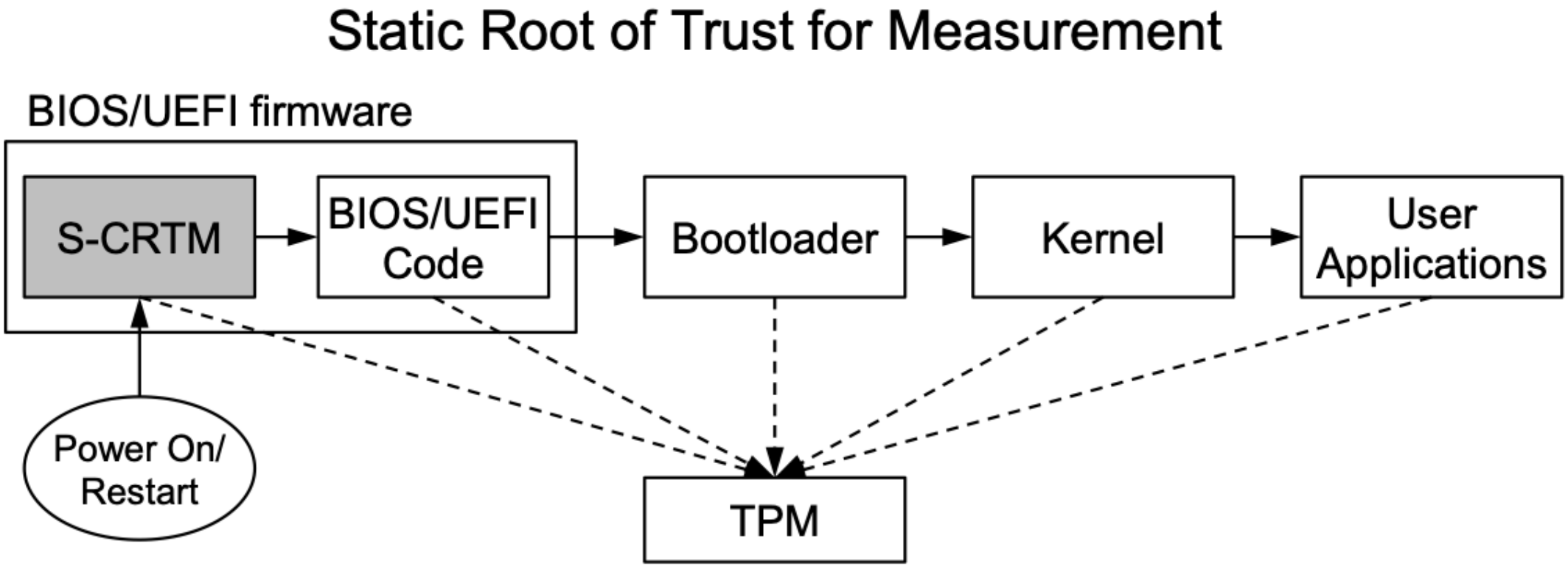
$$PCR_{initial} = 0x0000000000000000000000$$

$$PCR_{new} = Hash(PCR_{current} \,||\, measure)$$

UEFI Secure Boot



Kto to mierzy?



https://blog.elcomsoft.com/2021/01/understanding-bitlocker-tpm-protection/

Przeznaczenie banków

- PCR0 Core System Firmware executable code (aka Firmware)
- PCR1 Core System Firmware data (aka UEFI settings)
- PCR2 Extended or pluggable executable code
- PCR3 Extended or pluggable firmware data
- PCR4 Boot Manager Code and Boot Attempts
- PCR5 Boot Manager Configuration and Data
- PCR6 Resume from S4 and S5 Power State Events

Źródło: https://wiki.archlinux.org/title/Trusted_Platform_Module

Przeznaczenie banków cd.

- PCR7 Secure Boot State
- PCR8 Hash of the kernel command line
- PCR9 Hash of the initrd and EFI Load Options
- PCR10 Reserved for Future Use
- PCR11 Hash of the Unified kernel image
- PCR12 Overridden kernel command line, Credentials
- PCR13 System Extensions
- PCR14 shim's MokList, MokListX, and MokSBState.

Źródło: https://wiki.archlinux.org/title/Trusted_Platform_Module

Operacje na bankach PCR

- Odczyt wartości:

```
tpm2_pcrread sha256:0,7
```

- Rozszerzenie wartości (pomiar):

```
tpm2_pcrextend 4:sha256=<hash>
```

- Reset wartości (tylko bank 23, czasami 16):

```
tpm2_pcrreset 23
```

Sesje w TPMie

Rodzaje sesji

PASSWORD SESSION

HMAC SESSION

POLICY SESSION

Password Session

- Najprostsze i najmniej bezpieczna opcja:

```
tpm2_createprimary -c p.ctx
```

```
tpm2_create -C p.ctx -c sk.ctx -p passwd123
```

```
tpm2_sign -c sk.ctx -o msg.sig -p passwd123 msg.txt
```

HMAC Session

- Zapewnia integralność.
- Uniemożliwia wykonanie Signature Replay Attack dzięki nonce.
- Wprowadza dodatkowy context:

```
tpm2_startauthsession --hmac -S sess.ctx
tpm2_create -C p.ctx -c rsa.ctx -p session:sess.ctx+pwd
tpm2_flushcontext session.ctx
```

Policy Session

- Nie zapewnia poufności, ani integralności komunikacji.
- Wykorzystywana do definiowania polityk dostępu do obiektów:

```
tpm2_startauthsession --policy-session -S session.ctx
```

Szyfrowanie komunikacji

- W konfiguracji sesji możemy ustawić szyfrowanie dwustronne:

```
tpm2_startauthsession --hmac -S sessions.dat
```

```
tpm2_sessionconfig --enable-decrypt --enable-encrypt
session.dat
```

Najprostsza polityka dostępu (Password Session)

- Password Session:

```
tpm2_createprimary -c p.ctx
tpm2_create -C p.ctx -c sk.ctx -p passwd123
tpm2_sign -c sk.ctx -o msg.sig -p passwd123 msg.txt
```

Dictionary Lockout

- Czasowa blokada dostępu:

```
tpm2_createprimary -c p.ctx
```

```
tpm2_create -C p.ctx -c sk.ctx -p passwd123
```

```
tpm2_sign -c sk.ctx -o msg.sig -p passwd msg.txt
```

- ```
  tpm:session(1):the authorization HMAC check failed and DA counter incremented
  ```
  ```

Dictionary Lockout

- Liczenie nieudanych prób.
- Limit prób.
- Blokada (Lockout).
- Czas blokady.
- Reset blokady.

Dictionary Lockout

- Domyślne parametry:

```
tpm2_getcap properties-variable
```

- Zmiana parametrów:

```
tpm2_dictionarylockout -s -n 5 -t 6 -l 7 -p passwd
```

- Reset licznika:

```
tpm2_dictionarylockout -c -p passwd
```

Polityki dostępu

Bezpieczne klucze

- Jak dotąd każdy kto miał dostęp do TPM’a, mógł uzyskać dostęp do naszych kluczy.
- Dostęp do kluczy może być strzeżony na wiele sposobów.
- Możemy ograniczyć możliwe operacje z kluczem.
- A nawet ograniczyć ilość odwołań.

Password/Secret Policy

```
tpm2_startauthsession -S sess.ctx
tpm2_policypassword -S sess.ctx -L policy.dat
tpm2_flushcontext sess.ctx

tpm2_createprimary -C o -c prim.ctx
tpm2_create -g sha256 -G aes -c sk.ctx -C prim.ctx -
L policy.dat -p testpswd
```

PCR Policy

```
tpm2_createprimary -C e -g sha256 -G ecc -c primary.ctx
tpm2_pcrread -o pcr.dat "sha256:0,1,2,3"
tpm2_startauthsession -S session.dat
tpm2_policypcr -S session.dat -l "sha256:0,1,2,3" -f
pcr.dat -L policy.dat
tpm2_flushcontext session.dat
```

Signed Policy

```
tpm2_loadexternal -C o -G rsa -u public.pem -c signing_key.ctx
```

```
tpm2_startauthsession -S session.ctx
```

```
tpm2_policysigned -S session.ctx -g sha256 -s signature.dat -f rsassa -c signing_key.ctx -L policy.signed
```

```
tpm2_flushcontext session.ctx
```

Ticket Policy

- Tworzenie polityki:

```
tpm2_policysigned -S session.ctx -g sha256 -s
signature.dat -f rsassa -c signing_key.ctx -L
policy.signed
```

- Tworzenie biletu:

```
tpm2_policysigned -S session.ctx -g sha256 -s
signature.dat -f rsassa -c signing_key.ctx -x
nonce.test --ticket tic.ket --timeout time.out -t
0xFFFFFE0C
```

Ograniczenie operacji na kluczu

```
tpm2_startauthsession -S session.dat
```

```
tpm2_policycommandcode -S session.dat -L policy.dat
TPM2_CC_Unseal
```

```
tpm2_flushcontext session.dat
```

Ograniczenie dostępu do hierarchii

```
tpm2_changeauth -c owner newpass
tpm2_changeauth -c endorsement newpass
tpm2_changeauth -c lockout newpass
```

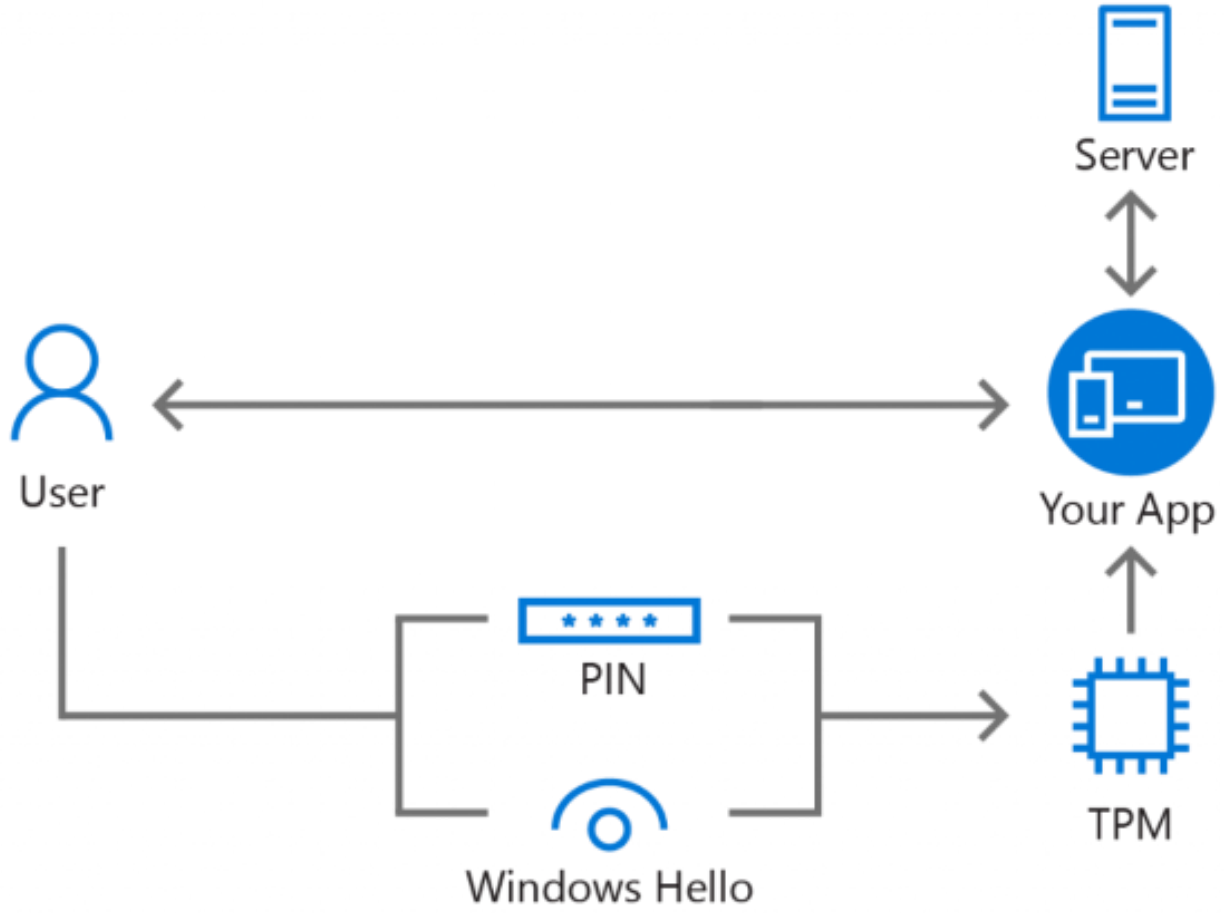
Aplikacje wykorzystujące TPM

OpenConnect

- Zabezpieczenie kluczy prywatnych,
- Uwierzytelnianie przy użyciu TPM,
- Zwiększone bezpieczeństwo.

Windows Hello for Business

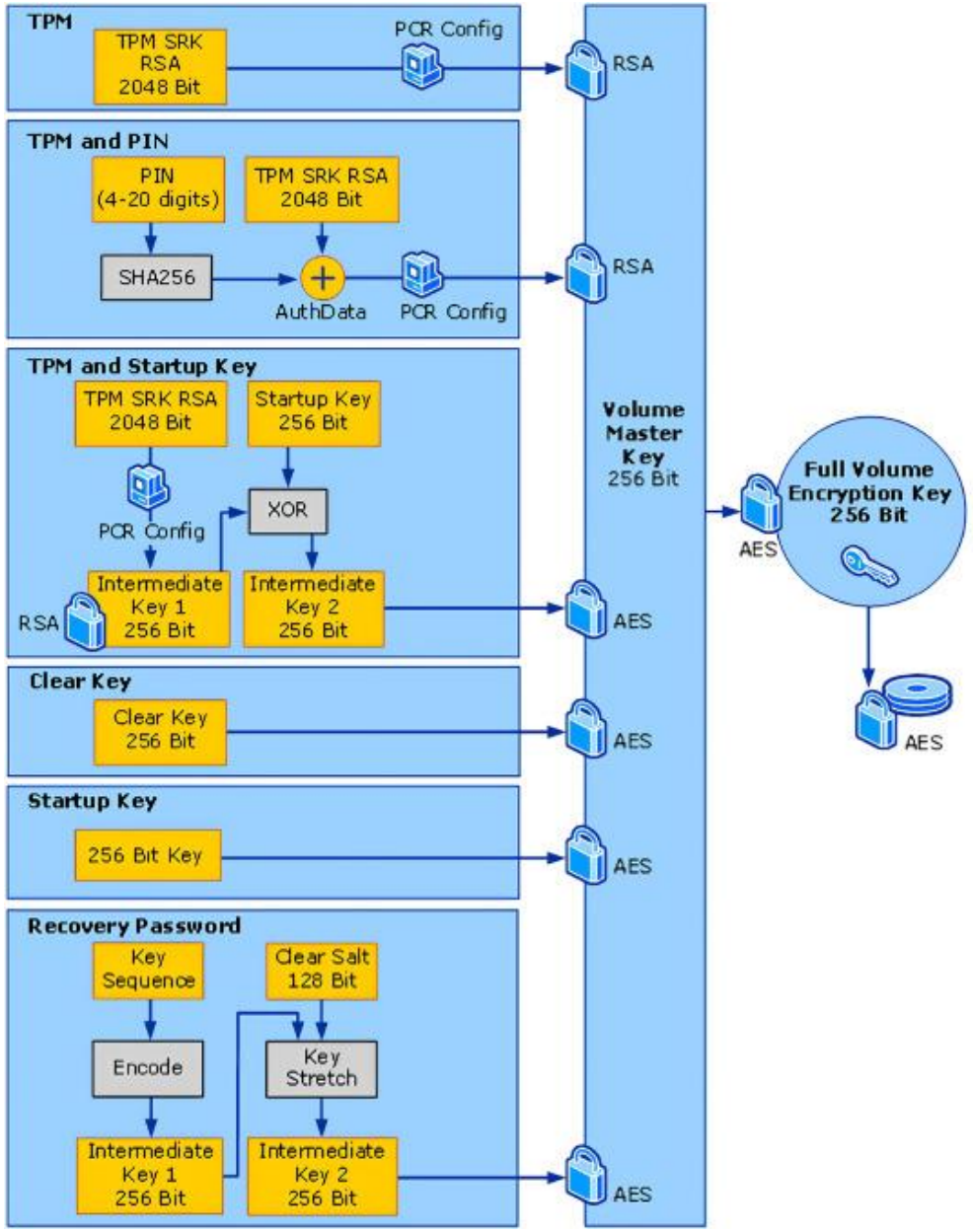
- Rejestracja klucza,
- Uwierzytelnianie,
- Zarządzanie tożsamością,
- Eliminacja haseł.



BitLocker

- Zabezpieczenie klucza szyfrującego dysk,
- Integralność procesu uruchamiania (Boot Integrity),
- Tryby uwierzytelniania,
- Ochrona przed atakami słownikowymi,
- Automatyczne odblokowanie po weryfikacji integralności.
  - Just don’t.

BitLocker

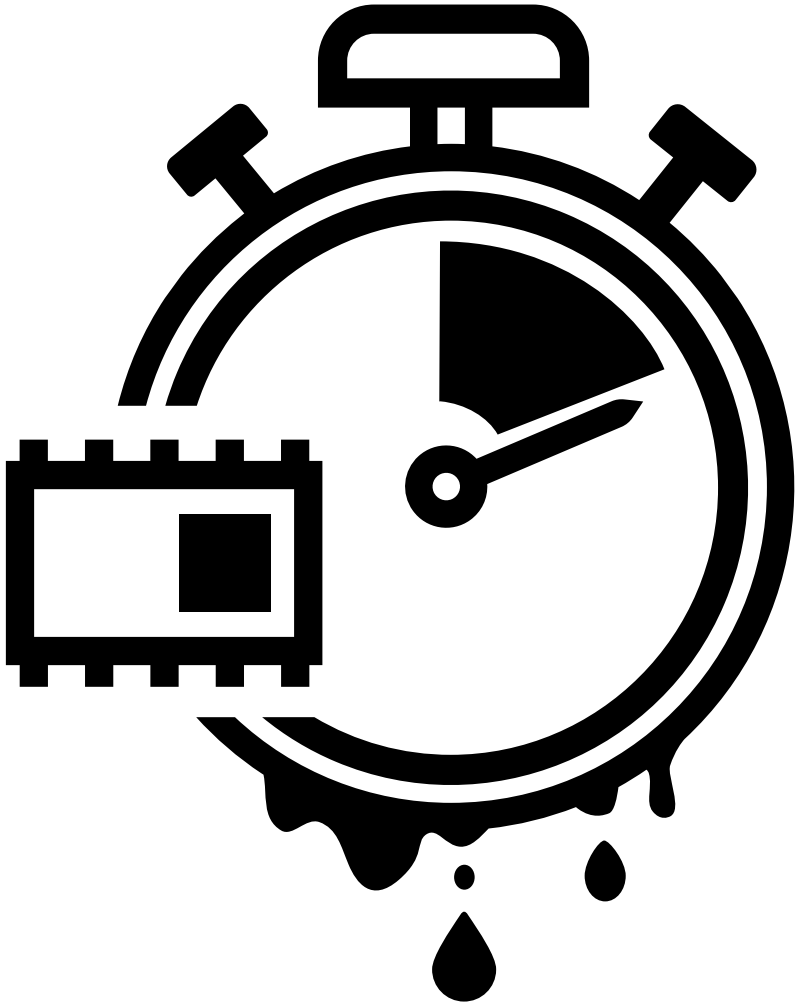


Źródło: https://www.sciencedirect.com/science/article/pii/S266628172300015X

Znane podatności

TPM-FAIL (CVE-2019-11090/16863)

- Side-channel Attack,
- Zdientyfikowane w fTPM (Intel) oraz dTPM (STMicroelectronics).



Źródło: https://tpm.fail/

Napper (CVE-2018-6622)

- Dotyczyła większości TPM’ów.
- Była związana ze złym zarządzaniem TPM’a.

Źródło: https://github.com/kkamagui/napper-for-tpm

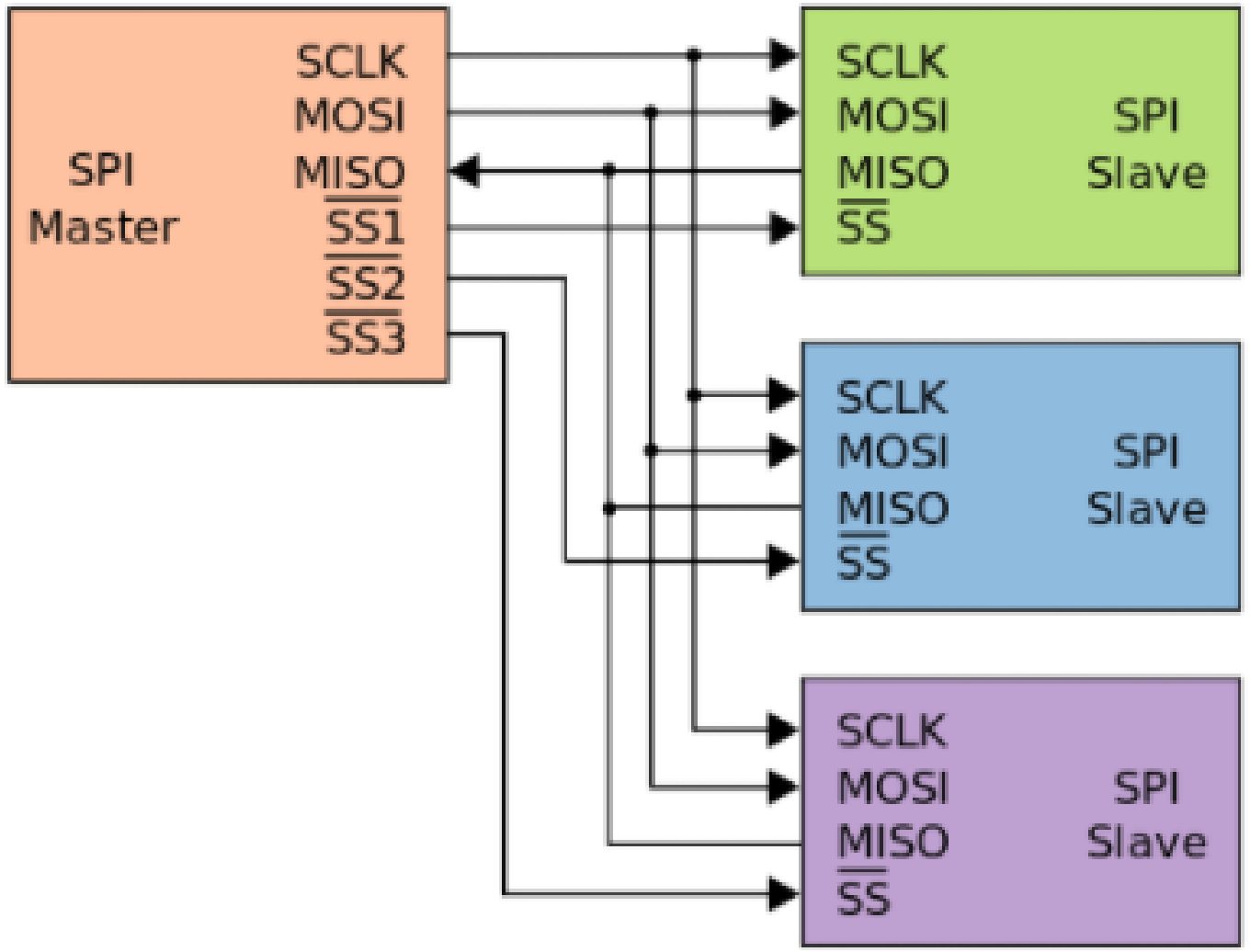
Memory Corruption (CVE-2023-1018/1017)

- Błędy typu oraz Write.
- CVE-2023-1017:
  - Out-of-bound Write.
- CVE-2023-1018:
  - Out-of-bound Read.

Problemy z FDE + TPM

SPI Sniffing

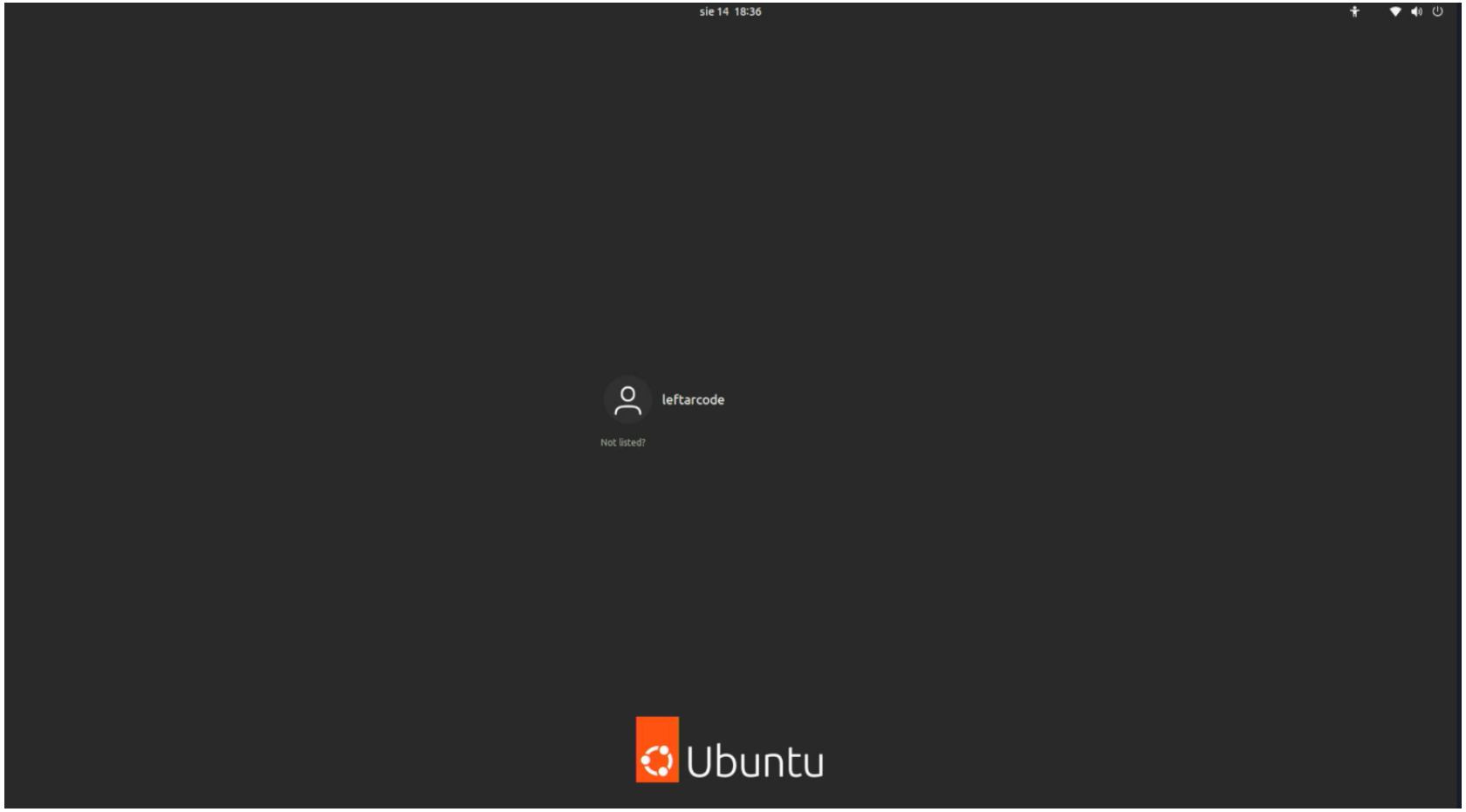
- BitLocker słabo chroni nasze sekrety w komunikacji z TPM’em.
- Możemy podsłuchać komunikację i wyciągnąć klucz.



LUKS i Clevis – słaba polityka PCR

- Clevis to narzędzie do automatycznego odblokowania dysku zaszyfrowanego LUKS’em.
- Większość tutoriali opiera się o bank:
  - PCR 7 – Secure Boot State,
  - PCR 1 – Ustawienia UEFI (m.in. Boot Order),
  - PCR 0 – firmware.
- To za mało.

Po odblokowaniu



Atak na LUKS + Clevis – ofiara

- Ubuntu 22.04.
- Skonfigurowany LUKS + Clevis:
  - Tylko bank PCR 7.

Atak na LUKS + Clevis – atak #1

- Partycja boot’owania nie jest zaszyfrowana.
- A żaden z jego plików nie jest mierzony przez bank PCR 7.
- Możemy zmienić ustawienia GRUB’a i wejść w tryb „recovery” z rootem.

Atak na LUKS + Clevis – atak #2

- Partycja boot’owania nie jest zaszyfrowana.
- A żaden z jego plików nie jest mierzony przez bank PCR 7.
- Możemy zmienić podmienić initramfs.
- Działa na wersje bez Clevis’a.

Atak na LUKS + Clevis – atak #3

- Dla takiego samego Chain of Trust dostaniemy ten sam PCR 7.
- Wystarczy obok zainstalować ten sam system.

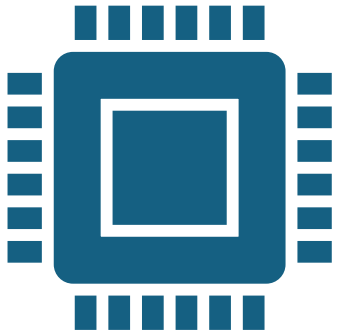
Atak na LUKS + Clevis – rozwiązanie

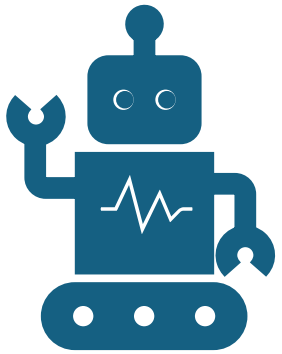
- Ustaw silne hasło na UEFI Setup.
- Rozważ rozszerzenie ustawień o kolejne banki m.in.:
  - PCR1,
  - PCR8,
  - PCR9.
- Nigdy nie korzystaj z automatycznego odblokowania dysku.

A co na to BitLocker?

- Opera się o PCR 7 i 11.
- Bank PCR 11 to blokada – „lock”.

Koniec





LI: @mateusz-lewczak

GH: @leftarcode

Bibliografia

- https://semiengineering.com/securely-store-your-credentials-and-cryptographic-keys-in-tpm-2-0/