# PV204 Security technologies

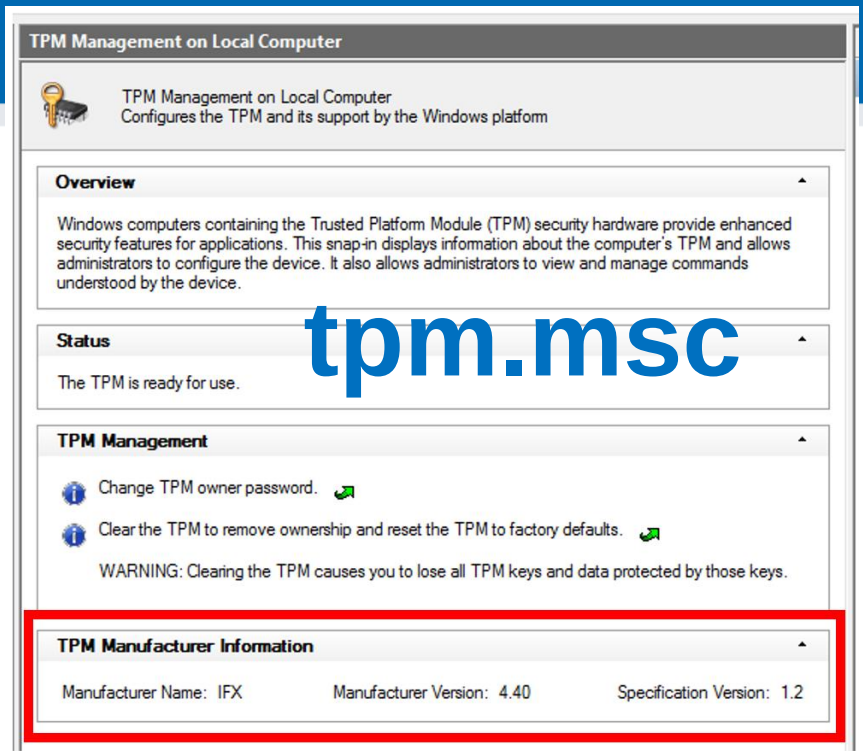
## Trusted Boot, TPM

**Petr Švenda** *svenda@fi.muni.cz* *@rngsec*
Centre for Research on Cryptography and Security, Masaryk University

CROCS
Centre for Research on Cryptography and Security

# FIDDLING WITH TPM: WHAT YOUR MACHINE HAS?

# Try to figure out your situation



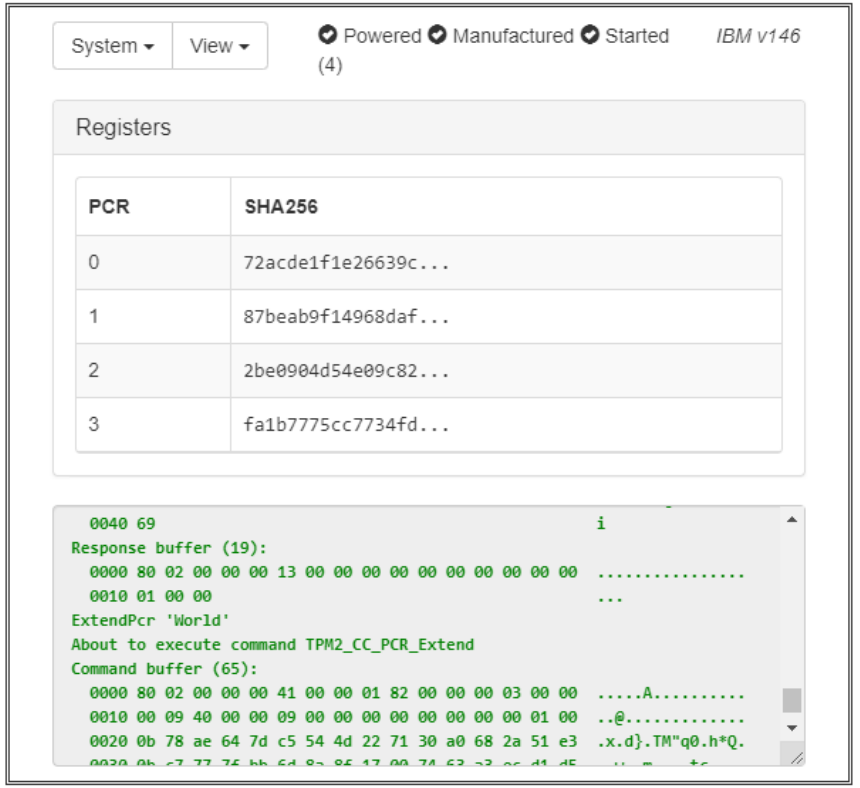

- Not every computer have TPM chip (Apple)
- Some chips are TPM 1.2, but now mostly 2.0
- Windows vs Linux difference
- IF Windows
  - Run in cmdline ‘tpm.msc’, ‘tpmtool getdeviceinformation’
- IF Linux & ‘/dev/tpm0’ exists & TPM 2.0:
  - ‘sudo apt-get update -y’ , ‘sudo apt-get install -y tpm2-tools’
  - Run ‘tpm2_pcrread’ or ‘tpm2_pcrlist’ or ‘tpm2_listpcrs’
- IF Linux & TPM 1.2 || Windows || anything doesn’t work:
  - Use https://labs.play-with-docker.com and TPM simulator

# FIDDLING WITH TPM: TPM-JS

# Google tpm-js

- Welcome
- Introduction
- Properties
- Random
- Keys
- Certificates
- Ownership
- PCRs
- Attestation
- Authorization
- Sealing
- Importing
- Glossary
- Reading

- JavaScript-based introduction, explanation and tutorial
  - https://google.github.io/tpm-js
- Locate binary commands when necessary (used later)
- Task: Understand all basic steps



# Questions

- How can you generate random numbers by TPM's TRNG?
- Who is manufacturer of the TPM in tpm-js?
- Why we have key hierarchy instead of single key?
- Can you create also symmetric keys on TPM? What is the command?
- Who is creating EK certificate? What is its usage?
- Which command is used to measure additional item (driver, app, cfg…)
- Can you extend only one of the PCR registers?
- Is it enough to prove state of your machine to remote party by reading out PCR registers and send them as file? Why?

# Questions

- What key is used by TPM during Remote Attestation?
- Can you perform Remote Attestation against device you never contacted before? Why?
- Why is there random nonce send by server during the Remote Attestation protocol?
- What is Event Log? Is it stored inside TPM? How you can verify its validity?

# FIDDLING WITH TPM: MS WINDOWS

# Fill a table into Miro https://miro.com/app/board/o9J_lLRWXoI=/

- Name of your computer (e.g., HP ProBook 6470b)
- Supported version of TPM specification
- TPM manufacturer (and manuf. Version)

# Windows: TPM_PCR collection tool

- Optional: please consider helping us with TPM research by periodic PCR collection
- Download TPM_PCR.exe v0.1.8 from https://github.com/petrs/TPM_PCR/releases
- Save to suitable directory (e.g., C:\Users\you\tpm)
- Run TPM_PCR.exe
- Type Y to confirm scheduling (will be performed every day at 19:00)
- Send PCR_measurements.zip to svenda@fi.muni.cz
  - And send it again when asked at the end of semester

# Windows: tpmtool

```
h:\Documents\Develop>tpmtool getdeviceinformation

-TPM Present: True
-TPM Version: 2.0
-TPM Manufacturer ID: IFX
-TPM Manufacturer Full Name: Infineon
-TPM Manufacturer Version: 7.63.3353.0
-PPI Version: 1.3
-Is Initialized: True
-Ready For Storage: True
-Ready For Attestation: True
-Is Capable For Attestation: True
-Clear Needed To Recover: False
-Clear Possible: True
-TPM Has Vulnerable Firmware: False
-PCR7 Binding State: 3
-Maintenance Task Complete: True
-TPM Spec Version: 1.16
-TPM Errata Date: Wednesday, September 21, 2016
-PC Client Version: 1.00
-Is Locked Out: False
```

- Basic information about your TPM device
  - **`tpmtool getdeviceinformation`**
- Collect extended information
  - **`tpmtool gatherlogs C:\Users\Public\tpm`**
  - Investigate file TpmInformation.txt
    - Device info, Capabilities, Supported Algorithms, Supported Commands, PCRs…
- Additional info about measured drivers etc.
  - **`tpmtool parsetcglogs > tpm_logs.txt`**
- Now continue with tpm2-tools steps (as for Linux)

# Questions

- What is your TPM version? Who is the vendor of your TPM?
- What symmetric and asymmetric crypto algorithms are supported?
- Can you locate Event Log? What is inside?
- Is command **`TPM2_PCR_Reset`** supported by your TPM? What is it behavior (use man page)? Is it breaking security assumptions for PCR values?

# FIDDLING WITH TPM: TPM2-TOOLS

# TUTORIAL FOR TPM 2.0 (IS->09_TRUSTEDBOOT->TPM2_COMMANDS.TXT)

# Tasks to achieve

1. Make real TPM accessible or run Docker with TPM simulator
2. Generate Random data from TPM chip (tpm2_getrandom)
3. Generate Random data using raw command (tpm2_send_command)
4. Figure out manufacturer ID of your TPM chip (raw cmd)
5. Obtain list of PCR registers (tpm2_listpcrs)
6. Extend PCR_01 with specific hash (raw TPM2_CC_PCR_Extend)
7. Generate new primary RSA keypair (raw TPM2_CC_CreatePrimary)
8. Create Remote Attestation report (raw TPM2_CC_Quote)

# Fill a table into Miro https://miro.com/app/board/o9J_lLRWXoI=/

- Name of your computer (e.g., HP ProBook 6470b)
- Supported version of TPM specification
- TPM manufacturer (and manuf. Version)
- Generate random data (tick OK when done)
- Generate random data using raw cmd (tick OK when done)
- Obtain list of PCR registers (tick OK when done)
- Extend PCR_01 with specific hash (tick OK when done)
- Generate new primary RSA keypair (tick OK when done)
- Create Remote Attestation report (tick OK when done)

**All commands are in *tpm2_commands.txt* file !!!**

# Running in Docker (with TPM simulator)

```
// Create account on docker.com
// Visit https://labs.play-with-docker.com

// Download and run docker image with TPM2 simulator
docker pull starlabio/tpm2-emulator
docker run -it starlabio/tpm2-emulator

// Try to generate 16 random bytes
tpm2_getrandom 16
// possible error: Failed to initialize tcti context: 0x1
// Problem: no TPM available, solution: run simulator

// Run simulator
tpm_server -rm &

// Try to generate random data again
tpm2_getrandom 16
// ERROR: TPM2_GetRandom Error. TPM Error:0x100
// Problem: TPM is not yet initialized (shall happen during boot, but not the case fo simulator)

// Startup TPM (with real TPM, it is already send by BIOS)
tpm2_startup --clear
```

Try to run ‘xxd’ command. If not found, then:

1. cd /usr/bin
2. wget https://www.fi.muni.cz/~xsvenda/xxd
3. chmod +x xxd
4. cd ~
5. wget https://www.fi.muni.cz/~xsvenda/vim

# The remaining commands are same for simulator as well as real TPM

# Generate random data (TPM2_CC_GetRandom)

```
// Try to generate random data
tpm2_getrandom 16
// Expected response if OK
Client accepted
Client accepted
0x12 0x5C 0xDA 0x46 0x40 0xBF 0x65 0x98 0x93 0xB2 0x12 0x0A 0x82 0xCE 0x64 0xC1
// our random data
TPM command server listening on port 2321

//
// We will now use raw commands for TPM extracted from https://google.github.io/tpm-js
//

// Any raw command in raw (get raw commands from https://google.github.io/tpm-js)
// Prepare raw command TPM2_CC_GetRandom
echo '80 01 00 00 00 0c 00 00 01 7b 00 0a' | xxd -r -p > getrandom.bin
// Send it to simulator (use -T device:/dev/tpm0 for physical TPM)
tpm2_send_command  -tcti socket:127.0.0.1:2321 < getrandom.bin > resp.bin
// Display response in hexa
cat resp.bin |  xxd

// Get Manufacturer string (TPM2_CC_GetCapability)
echo '80 01 00 00 00 16 00 00 01 7a 00 00 00 06 00 00 01 05 00 00 00 01' | xxd -r -p > getmanuf.bin
tpm2_send_command  -tcti socket:127.0.0.1:2321 < getmanuf.bin > resp.bin
cat resp.bin |  xxd
```

# List and extend PCRs (TPM2_CC_PCR_Extend)

```
// Try to obtain PCRs
tpm2_listpcrs
// Will be mostly zero for simulator, but non-zero for real TPM

// TPM2_CC_PCR_Extend command (small enough to paste directly into echo)
echo '80 02 00 00 00 41 00 00 01 82 00 00 00 01 00 00 00 09 40 00 00 09 00 00 00 00 00 00 00
00 01 00  0b 18 5f 8d b3 22 71 fe 25 f5 61 a6 fc 93 8b 2e 26 43 06 c 30 4e da 51 80 07 d1 76 48
26 38 19 69' | xxd -r -p > extend.bin
// Send raw command
tpm2_send_command  -tcti socket:127.0.0.1:2321 < extend.bin > resp.bin
// Display response in hexa
cat resp.bin |  xxd

// Display PCRs again and observe that PCR_01 has changed 'PCR_01: 7a fe da f7 f1 b9 a5 e7...'
tpm2_listpcrs
```

# Remote attestation (TPM2_CC_Quote)

```
// Remote attestation (https://google.github.io/tpm-js/#pg_attestation)
// Locate, extract and convert to binary TPM2_CC_CreatePrimary command
// from https://google.github.io/tpm-js/#pg_attestation
vim createprimary.bin, insert mode, paste from tpm-js, :%!xxd -r, :wq

  0000 80 02 00 00 00 61 00 00 01 31 40 00 00 01 00 00  .....a...1@.....
  0010 00 09 40 00 00 09 00 00 00 00 00 00 04 00 00 00  ..@.............
  0020 00 00 38 00 01 00 0b 00 05 00 72 00 00 00 10 00  ..8.......r.....
  0030 14 00 0b 08 00 00 00 00 00 00 20 e3 b0 c4 42 98  .......... ...B.
  0040 fc 1c 14 9a fb f4 c8 99 6f b9 24 27 ae 41 e4 64  ........o.$'.A.d
  0050 9b 93 4c a4 95 99 1b 78 52 b8 55 00 00 00 00 00  ..L....xR.U.....
  0060 00

// IMPORTANT: after xxd-r inside VIM, there might be additional 0x0a character - must be removed

// Send to TPM
tpm2_send_command  -tcti socket:127.0.0.1:2321 < createprimary.bin > resp.bin
// Display response in hexa
cat resp.bin |  xxd

// Locate, extract and covert to binary TPM2_CC_Quote command from https://google.github.io/tpm-js/#pg_attestation
// Freshness challenge is '0.ncmv81u451'
// IMPORTANT: after xxd-r inside VIM, there might be additional 0x0a character - must be removed
vim quote.bin, insert mode, paste from tpm-js, :%!xxd -r, :w, :q

  0000 80 02 00 00 00 35 00 00 01 58 80 00 00 00 00 00  .....5...X......
  0010 00 09 40 00 00 09 00 00 00 00 00 00 0c 30 2e 6e  ..@..........0.n
  0020 63 6d 76 38 31 75 34 35 31 00 10 00 00 00 01 00  cmv81u451.......
  0030 0b 03 0f 00 00

tpm2_send_command  -tcti socket:127.0.0.1:2321 < quote.bin > resp.bin
// Display response in hexa, shall be ~408B long and contains 'TCG' string
cat resp.bin |  xxd
```

# Homework – no new homework this week ☺

- Please consider participating in our TPM2 data collection experiment
  - If OS Windows: https://github.com/petrs/TPM_PCR/releases
  - Any OS: https://is.muni.cz/go/tpm

# OLDER TUTORIAL (TPM 1.2)

# Organizational

- Not every computer have TPM chip
  - Make groups with at least one TPM-enabled computer
  - Use own computer (if TPM-enabled) or the provided ones
- Usage differs between TPM1.2 and 2.0 (but backward compatible)
- Usage of TPM differs between Windows and Linux
  - We will focus on Windows, but you may try Linux as well
- Prepared software (Windows, Linux)
  - Preconfigured binaries and cheetsheet in IS 07_TPM.zip
  - Or https://www.fi.muni.cz/~xsvenda/tpm.zip
  - Use printed cheetsheet

# Questions to answer

1. Figure out maker and version of TPM chip
2. Obtain number of OS boot counts
3. Obtain list of PCR registers
4. Generate new RSA keypair and export public key
5. Seal (encrypt) data so only your machine will be able to decrypt
6. (optional) Try to create Remote Attestation report

# Fill the table on the whiteboard ☺

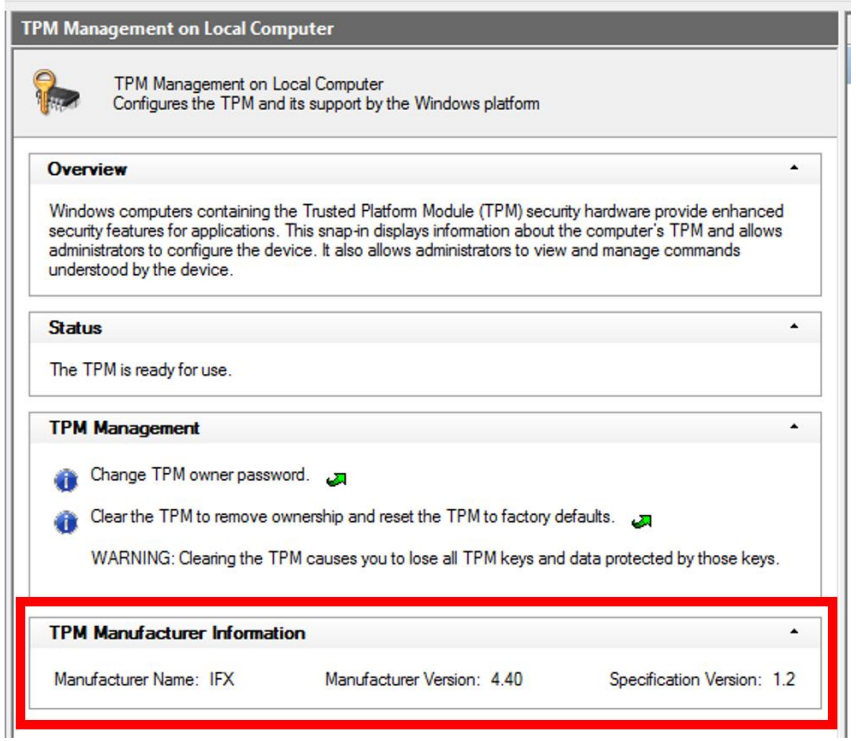
- Name of your computer (e.g., HP ProBook 6470b)
- Supported version of TPM specification
- TPM manufacturer and manuf. version
- Number of OS Boot Counts
- Platform Configuration Registers #used / #unused
- Generate on-TPM RSA key (tick OK when done)
- Seal data (tick OK when done)
- Remote Attestation (tick OK when done)

# Sending TPM commands, tools

- ISO/IEC 11889 standard for secure crypto-processor
- Versions published by Trusted Computing Group
  - TPM 1.2 (2011), TPM 2.0 (2016, not compatible with 1.2)
  - https://trustedcomputinggroup.org
- Windows: Microsoft PCPTool, TSS.MSR
- Linux: tpm_tools, GUI TPMManager
- (All tools and scripts available in single package)
  - https://www.fi.muni.cz/~xsvenda/tpm.zip

# Is TPM chip inside my computer?

- Windows
  - WinButton+R
  - tpm.msc (requires admin)
- Linux
  - sudo apt-get install tpm-tools
  - tpm_setactive -s
  - tpm_setactive -a
  - tpm_version
  - (systemctl restart tcsd)
- Check BIOS settings (TPM or Security chip…)



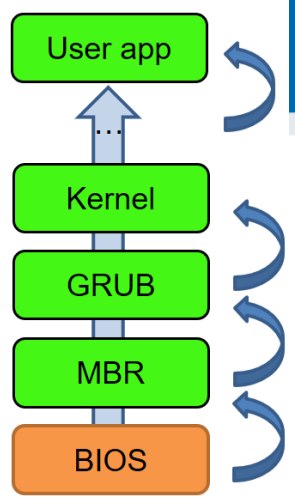
# TPM platform info

- Provides information about your platform state
- **W:** PCPTool.exe GetPlatformCounters
- **L:** not readily available, try
  - sudo cat /sys/kernel/security/tpm0/ascii_bios_measurements
  - sudo cat /sys/kernel/security/ima/ascii_runtime_measurements

```
<PlatformCounters>
  <OsBootCount>44</OsBootCount>
  <OsResumeCount>2</OsResumeCount>
  <CurrentBootCount>0</CurrentBootCount>
  <CurrentEventCount>66</CurrentEventCount>
  <CurrentCounterId>179136858</CurrentCounterId>
  <InitialBootCount>0</InitialBootCount>
  <InitialEventCount>64</InitialEventCount>
  <InitialCounterId>179136858</InitialCounterId>
</PlatformAttestation>
```

Reboot =>

```
<PlatformCounters>
  <OsBootCount>45</OsBootCount>
  <OsResumeCount>0</OsResumeCount>
  <CurrentBootCount>0</CurrentBootCount>
  <CurrentEventCount>67</CurrentEventCount>
  <CurrentCounterId>179136858</CurrentCounterId>
  <InitialBootCount>0</InitialBootCount>
  <InitialEventCount>67</InitialEventCount>
  <InitialCounterId>179136858</InitialCounterId>
</PlatformAttestation>
```




# Platform attestation – PCR registers

- **W:** PCPTool.exe GetPCRs
- **L:** cat `find /sys/class/ -name "tpm0"`/device/pcrs

*Table 12-1. Example PCR Allocation*

| PCR Number | Allocation |
|---|---|
| 0 | BIOS |
| 1 | BIOS configuration |
| 2 | Option ROMs |
| 3 | Option ROM configuration |
| 4 | MBR (master boot record) |
| 5 | MBR configuration |
| 6 | State transitions and wake events |
| 7 | Platform manufacturer specific measurements |
| 8–15 | Static operating system |
| 16 | Debug |
| 23 | Application support |

```
bug>PCPTool.exe GetPCRs
<PCRs>
  <PCR Index="00">8cb1a2e093cf41c1a726bab3e10bc1750180bbc5</PCR>
  <PCR Index="01">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
  <PCR Index="02">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
  <PCR Index="03">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
  <PCR Index="04">1e3c5e15b5f023765147535e092d22d7c17421e1</PCR>
  <PCR Index="05">75acbe8a48ba02a85d6301b33005d08678176c87</PCR>
  <PCR Index="06">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
  <PCR Index="07">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
  <PCR Index="08">0000000000000000000000000000000000000000</PCR>
  <PCR Index="09">0000000000000000000000000000000000000000</PCR>
  <PCR Index="10">0000000000000000000000000000000000000000</PCR>
  <PCR Index="11">ebb98df76613280f20dc38221143a9e727399486</PCR>
  <PCR Index="12">67afac5ca0fc6c9a3d881d681121f7d43d0c7128</PCR>
  <PCR Index="13">be1d9bd7318a9140b26f00a5283f37a6111bb1e5</PCR>
  <PCR Index="14">7f599cd09efefc7422085a0f490f8f1cba8761a8</PCR>
  <PCR Index="15">0000000000000000000000000000000000000000</PCR>
  <PCR Index="16">0000000000000000000000000000000000000000</PCR>
  <PCR Index="17">ffffffffffffffffffffffffffffffffffffffff</PCR>
  <PCR Index="18">ffffffffffffffffffffffffffffffffffffffff</PCR>
  <PCR Index="19">ffffffffffffffffffffffffffffffffffffffff</PCR>
  <PCR Index="20">ffffffffffffffffffffffffffffffffffffffff</PCR>
  <PCR Index="21">ffffffffffffffffffffffffffffffffffffffff</PCR>
  <PCR Index="22">ffffffffffffffffffffffffffffffffffffffff</PCR>
  <PCR Index="23">0000000000000000000000000000000000000000</PCR>
</PCRs>
```

# Platform info

- Obtain information about your platform
- Version info: pcptool GetVersion
- Get platform counters: pcptool GetPlatformCounters

```
<PlatformCounters>
  <OsBootCount>44</OsBootCount>
  <OsResumeCount>2</OsResumeCount>
  <CurrentBootCount>0</CurrentBootCount>
  <CurrentEventCount>66</CurrentEventCount>
  <CurrentCounterId>179136858</CurrentCounterId>
  <InitialBootCount>0</InitialBootCount>
  <InitialEventCount>64</InitialEventCount>
  <InitialCounterId>179136858</InitialCounterId>
</PlatformAttestation>
```

Reboot =>

```
<PlatformCounters>
  <OsBootCount>45</OsBootCount>
  <OsResumeCount>0</OsResumeCount>
  <CurrentBootCount>0</CurrentBootCount>
  <CurrentEventCount>67</CurrentEventCount
  <CurrentCounterId>179136858</CurrentCou
  <InitialBootCount>0</InitialBootCount>
  <InitialEventCount>67</InitialEventCount>
  <InitialCounterId>179136858</InitialCounter
</PlatformAttestation>
```

# Encrypt data only for your TPM (Windows)

- (RSA key with name *openlab* already generated)

1. Export public key
   – PCPTool.exe GetPubKey openlab openlab.pub
2. Encrypt data by public key
   – PCPTool.exe Encrypt openlab.pub Hello msg_enc.bin
3. Decrypt only on your machine
   – PCPTool.exe Decrypt openlab msg_enc.bin

# Encrypt data only for your TPM (Linux)

- Sealed storage using Root Storage Key

```
echo "Hello World!" > cleartext.txt
tpm_sealdata --well-known --infile cleartext.txt > encrypted.txt
tpm_sealdata -z -i cleartext.txt > encrypted.txt
cat encrypted.txt
tpm_unsealdata --srk-well-known --infile encrypted.txt
tpm_unsealdata -z -i encrypted.txt
```

- The proper way for TPM encryption is installing the openssl_tpm_engine,however the repository is not maintained anymore and does not build on my system (not even with OpenSSL 0.9.8) https://sourceforge.net/p/trousers/openssl_tpm_engine/ci/master/tree/
- There is a newer patched version, but too complicated: https://blog.hansenpartnership.com/using-your-tpm-as-a-secure-key-store/

# Holy grail: Remote attestation

- Apps running on your computer measured in PCRs
- Your TPM contains unique Endorsement key
- You can generate Attestation key inside TPM (AIK)
  - And sign AIK by Endorsement key (inside TPM)
- You can sign your PCRs by AIK (inside TPM)
- Remote party can verify signature on AIK key
  - Using public key of Endorsement key
- Remote party can verify signature on PCRs
  - Using public key of AIK key
- Remote party now knows what you are running

# Attestation keys

1. Create attestation identity key (AIK)
   – CreateAIK AIK_NAME filename aikNonce
2. Get public part of attestation key
   – GetPubAIK
3. Authentication of generated AIK to remote entity
   – Omitted (challenge-response and endorsement key used)
4. Get platform attestation signed by AIK
   – GetPlatformAttestation

# 1. Create attestation key

- PCPTool.exe CreateAIK myAIK test.tmp 1234

```
<AIK>
 <RSAKey size="283" keyName="myAIK">
  <Magic>RSA1<!-- 0x31415352 --></Magic>
  <BitLength>2048</BitLength>
  <PublicExp size="3">
   010001
  </PublicExp>
  <Modulus size="256" digest="520aabc242eddb488d1c3da30f56b4268222982a">
   9ddc3bb99eab0d9…d0fb46a48224cf15e9
  </Modulus>
  <Prime1/>
  <Prime2/>
 </RSAKey>
 <IdentityBinding size="568">01010000000000079139f69c93c042496a8e958ec5930662c6c
     ccafbf000000010…093873f194ce7b68ef667f00eca2090adad3
 </IdentityBinding>
</AIK>
```

# 2. Get public part of attestation key

- PCPTool.exe GetPubAIK test.tmp AIKPub.key

```
<RSAKey size="283" keyName="AIK">
  <Magic>RSA1<!-- 0x31415352 --></Magic>
  <BitLength>2048</BitLength>
  <PublicExp size="3">
    010001
  </PublicExp>
  <Modulus size="256"
    digest="520aabc242eddb488d1c3da30f56b4268222982a">
    9ddc3bb99eab0d913cd…0a40de6d62424b9a311
  </Modulus>
  <Prime1/>
  <Prime2/>
</RSAKey>
```

# 3. Get platform attestation

- PCPTool.exe GetPlatformAttestation myAIK attestation.tmp 4321
  - TpmAttGeneratePlatformAttestation() called internally
  - Large XML file is produced

- Why is AIK relevant for platform attestation?
- Why make sense to have multiple AIKs?
- Why is nonce 4321 included?

# 4. Platform attestation – PCR registers

```
<PlatformAttestation size="30591">
  <Magic>PADS<!-- 0x53444150 --></Magic>
  <Platform>TPM_VERSION_12</Platform>
  <HeaderSize>28</HeaderSize>
  <PcrValues size="480">
    <PCR Index="0">8cb1a2e093cf41c1a726bab3e10bc1750180bbc5</PCR>
    <PCR Index="1">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
    <PCR Index="2">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
    <PCR Index="3">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
    <PCR Index="4">68fffb7e5c5f6e6461b3527a0694f41ebd07e4e1</PCR>
    <PCR Index="5">8e33d52190def152c9939e9dd9b0ea84da25d29b</PCR>
    <PCR Index="6">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
    <PCR Index="7">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
    <PCR Index="8">0000000000000000000000000000000000000000</PCR>
    <PCR Index="9">0000000000000000000000000000000000000000</PCR>
    <PCR Index="10">0000000000000000000000000000000000000000</PCR>
    <PCR Index="11">b2a83b0ebf2f8374299a5b2bdfc31ea955ad7236</PCR>
    <PCR Index="12">7c84e69cd581eefd7ebe1406666711fd4fda8aa8</PCR>
    <PCR Index="13">01788a8a31f2dafcd9fe58c5a11701e187687d49</PCR>
    <PCR Index="14">26cda47f1db41bedc2c2b1e6c91311c98b4e2246</PCR>
    <PCR Index="15">0000000000000000000000000000000000000000</PCR>
    <PCR Index="16">0000000000000000000000000000000000000000</PCR>
    <PCR Index="17">ffffffffffffffffffffffffffffffffffffffff</PCR>
    <PCR Index="18">ffffffffffffffffffffffffffffffffffffffff</PCR>
    <PCR Index="19">ffffffffffffffffffffffffffffffffffffffff</PCR>
    <PCR Index="20">ffffffffffffffffffffffffffffffffffffffff</PCR>
    <PCR Index="21">ffffffffffffffffffffffffffffffffffffffff</PCR>
    <PCR Index="22">ffffffffffffffffffffffffffffffffffffffff</PCR>
    <PCR Index="23">0000000000000000000000000000000000000000</PCR>
  </PcrValues>
```

# 4. Platform attestation – EFI boot info

```
<EV_EFI_GPT_Event PCR="05" EventDigest="46bd9620d741812ca1c2bd82c375a5f58d258bea" Size
  4546492050415254000001005c00000015ff4614000000001000000000000af32cf1d000000002
  <!-- EFI.PART.....F.....2.....2.....4...N.2.Eg.....9K7.....M.j...y...
 </EV_EFI_GPT_Event>
 <EV_EFI_Boot_Services_Application PCR="04" Digest="65bb31b71a030a3fe93ba4d64e4ae0cedabb
  18e086a40000000058ad1800000000000000010000000034000000000000040430005c00450
  <!-- .....X.....4.....0...E.F.I...B.o.o.t...B.o.o.t.x.6.4...e.f.i..... -->
 </EV_EFI_Boot_Services_Application>
```