black hat USA 2020

AUGUST 5-6, 2020

BRIEFINGS

Remote Timing Attacks on TPMs, AKA TPM-Fail

Daniel Moghimi

About Me

- Daniel Moghimi
  - @danielmgmi
  - https://moghimi.org

- Security Researcher

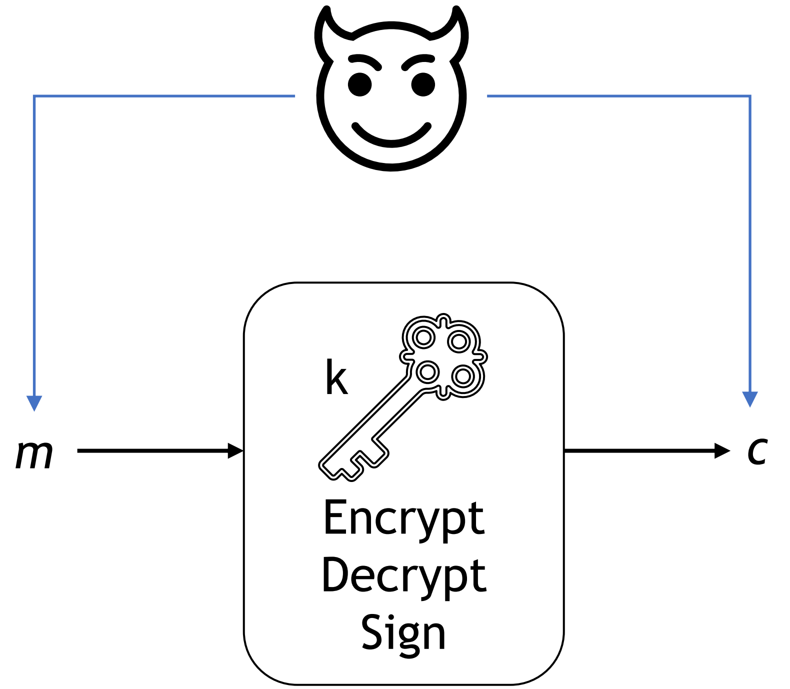
- PhD Candidate @ WPI
  - Microarchitectural Attacks
  - Side Channels
  - Breaking Crypto Implementations
  - Trusted Execution Environment (Intel SGX)

Thanks!

- Berk Sunar @ WPI
- Nadia Heninger @ UCSD
- Thomas Eisenbarth @ UzL
- Jan Wichelmann @ UzL

Cryptanalysis

- Cryptosystem with an input *m*, output *c*, and secret *k*
- Attacker tries to learn *k* by looking at (*m*, *c*)

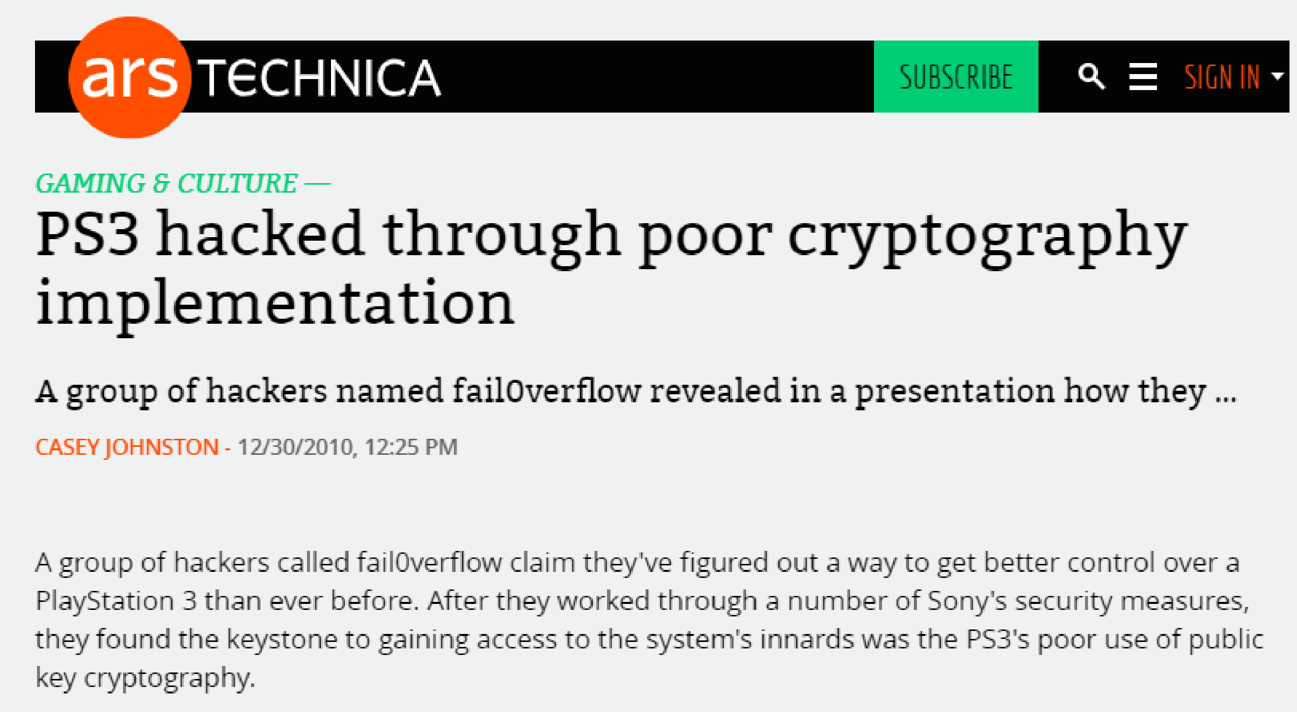


Cryptanalysis

- Cryptosystem with an input *m*, output *c*, and secret *k*
- Attacker tries to learn *k by* looking at (*m*, *c*)

$ECDSA\ Sign$:
$(x_1, y_1) = k_i \times G$
$r_i = x_1 \bmod n$
$s_i = {k_i}^{-1}(z + r_i d) \bmod n$

$s_1 = {k_1}^{-1}(z + r_1 d) \bmod n$
$s_2 = {k_2}^{-1}(z + r_2 d) \bmod n$



GAMING & CULTURE —

PS3 hacked through poor cryptography implementation

A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 12:25 PM

A group of hackers called fail0verflow claim they've figured out a way to get better control over a PlayStation 3 than ever before. After they worked through a number of Sony's security measures, they found the keystone to gaining access to the system's innards was the PS3's poor use of public key cryptography.

Cryptanalysis

- Cryptosystem with an input *m*, output *c*, and secret *k*
- Attacker tries to learn *k by* looking at (*m*, *c*)

$ECDSA\ Sign$:

$(x_1, y_1) = k_i \times G$

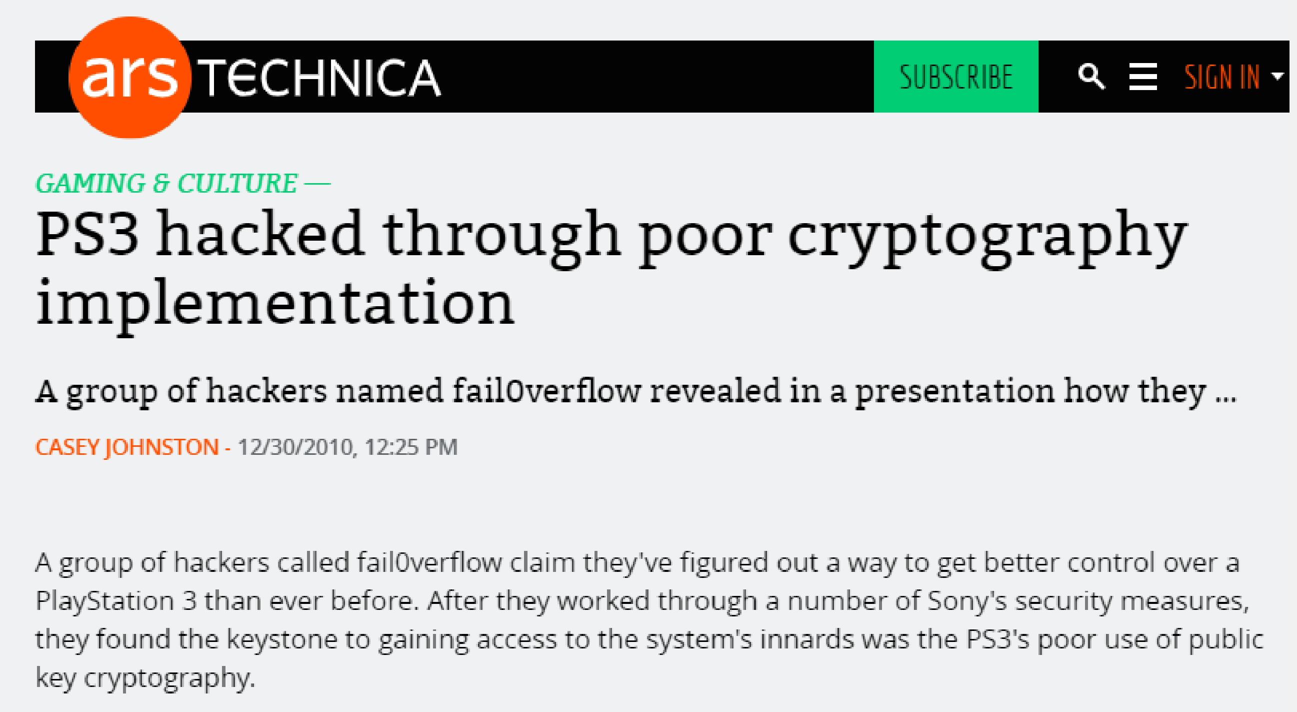
$r_i = x_1 \ mod\ n$

$s_i = k_i^{-1}(z + r_i d) \ mod\ n$

$k_1 = k_2 = k_n$

$s_1 = k^{-1}(z + r_1 d) \ mod\ n$

$s_2 = k^{-1}(z + r_2 d) \ mod\ n$



GAMING & CULTURE —

PS3 hacked through poor cryptography implementation

A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 12:25 PM

A group of hackers called fail0verflow claim they've figured out a way to get better control over a PlayStation 3 than ever before. After they worked through a number of Sony's security measures, they found the keystone to gaining access to the system's innards was the PS3's poor use of public key cryptography.

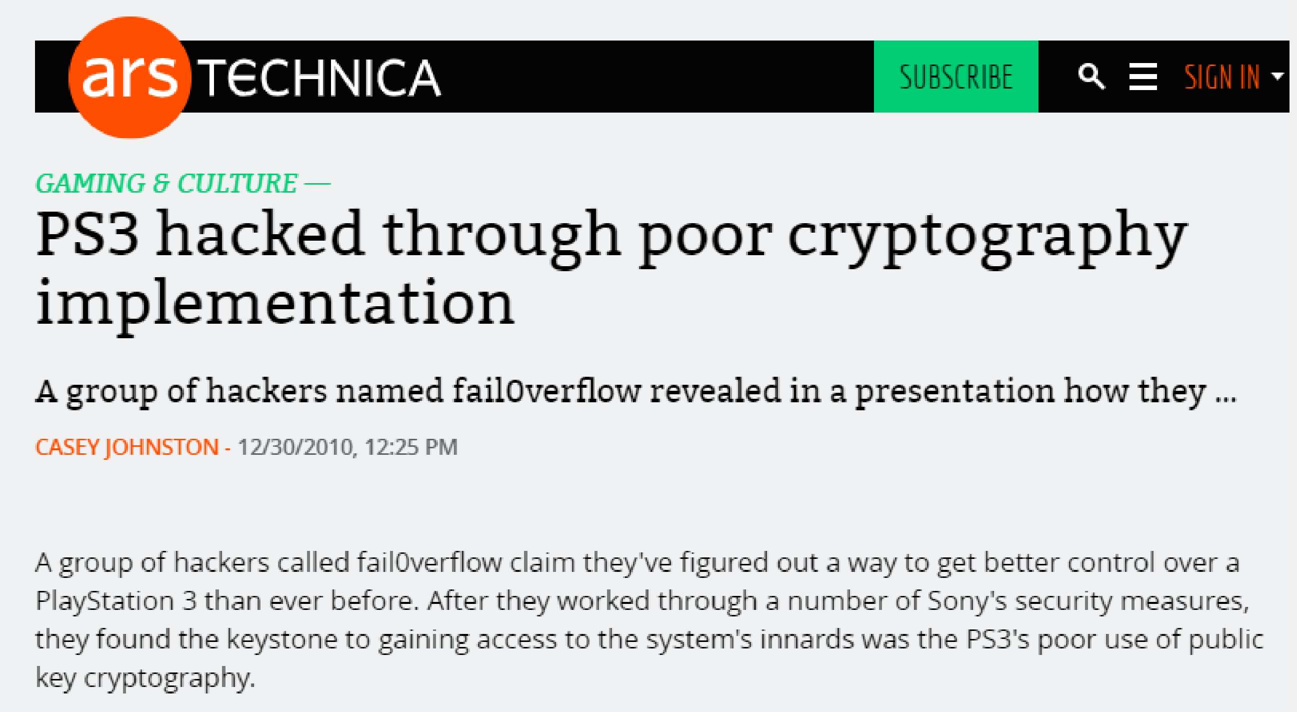
Cryptanalysis

- Cryptosystem with an input *m*, output *c*, and secret *k*
- Attacker tries to learn *k by* looking at (*m*, *c*)

$$\mathit{ECDSA\ Sign}:$$
$$(x_1, y_1) = k_i \times G$$
$$r_i = x_1 \ mod \ n$$
$$s_i = {k_i}^{-1}(z + r_i d) \ mod \ n$$

$$k_1 = k_2 = k_n$$
$$s_1 = k^{-1}(z + r_1 d) \ mod \ n$$
$$s_2 = k^{-1}(z + r_2 d) \ mod \ n$$

$$s_2 - s_1 = (r_2 - r_1) \ d \ mod \ n$$



GAMING & CULTURE —

PS3 hacked through poor cryptography implementation

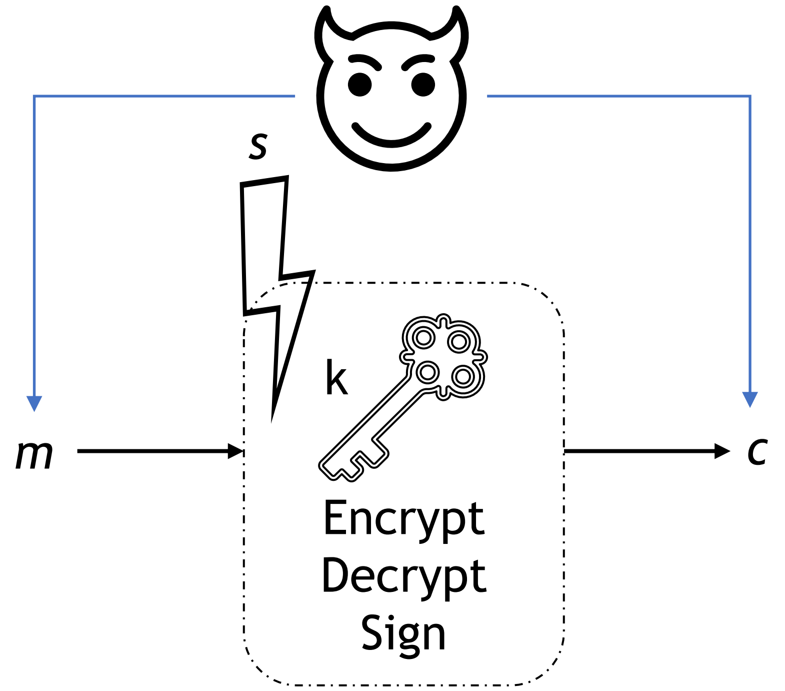
A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 12:25 PM

A group of hackers called fail0verflow claim they've figured out a way to get better control over a PlayStation 3 than ever before. After they worked through a number of Sony's security measures, they found the keystone to gaining access to the system's innards was the PS3's poor use of public key cryptography.

Side-Channel Cryptanalysis

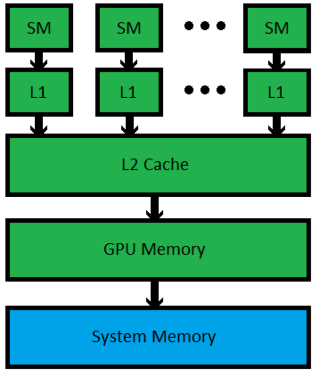
- Cryptosystem with an input *m*, output *c*, and secret *k*
- Attacker tries to learn *k* by looking at (*m*, *c*) and signal *s*



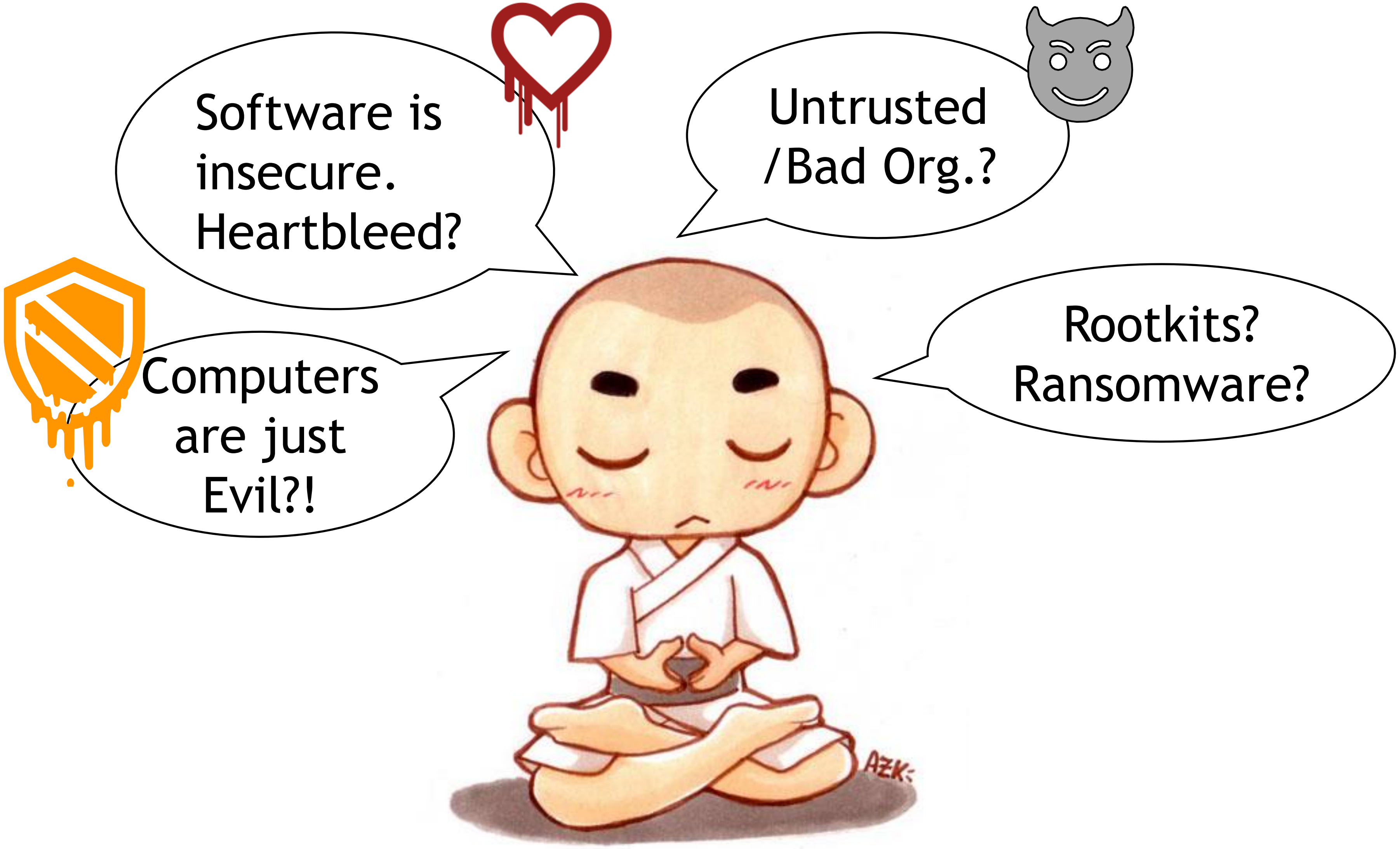
Side-Channel Attacks

- Channels
  - Power Analysis
  - EM Analysis
  - …
  - *Timing Analysis*
  - *CPU Side Channels*

- Threat Models:
  - Physical Access
  - Local Access (Co-location)
  - Remote



Secure Elements



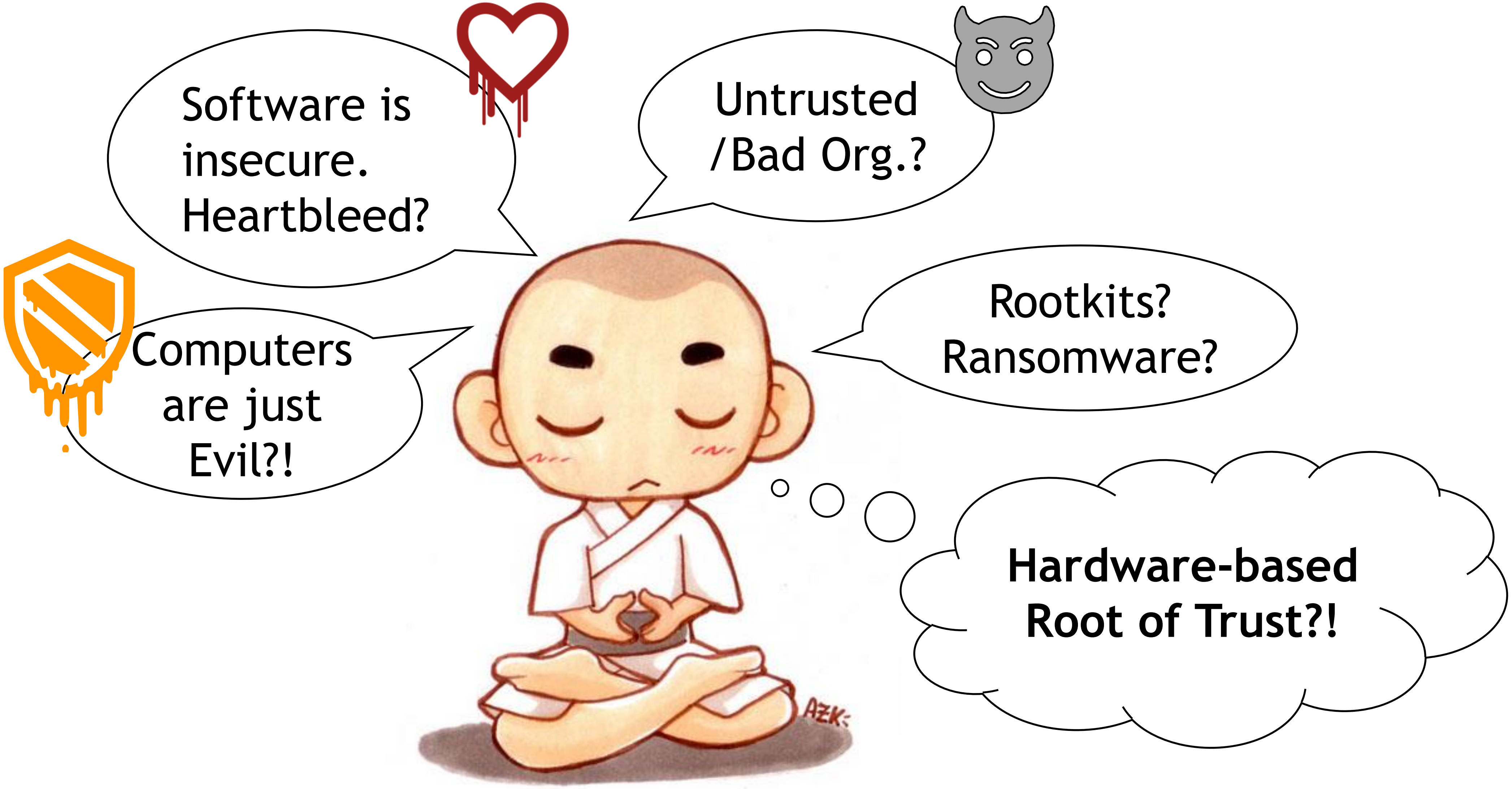
Software is insecure. Heartbleed?

Untrusted /Bad Org.?

Computers are just Evil?!

Rootkits? Ransomware?

Secure Elements



Software is insecure. Heartbleed?

Untrusted /Bad Org.?

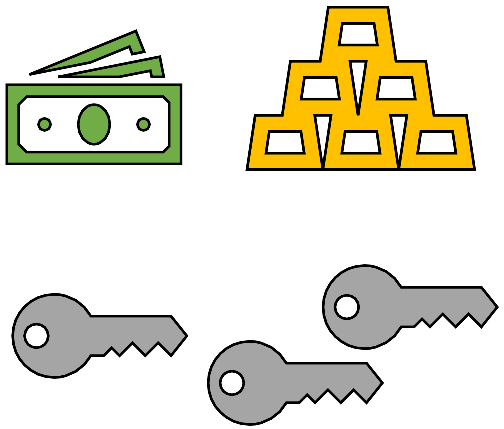
Computers are just Evil?!

Rootkits? Ransomware?

Hardware-based Root of Trust?!

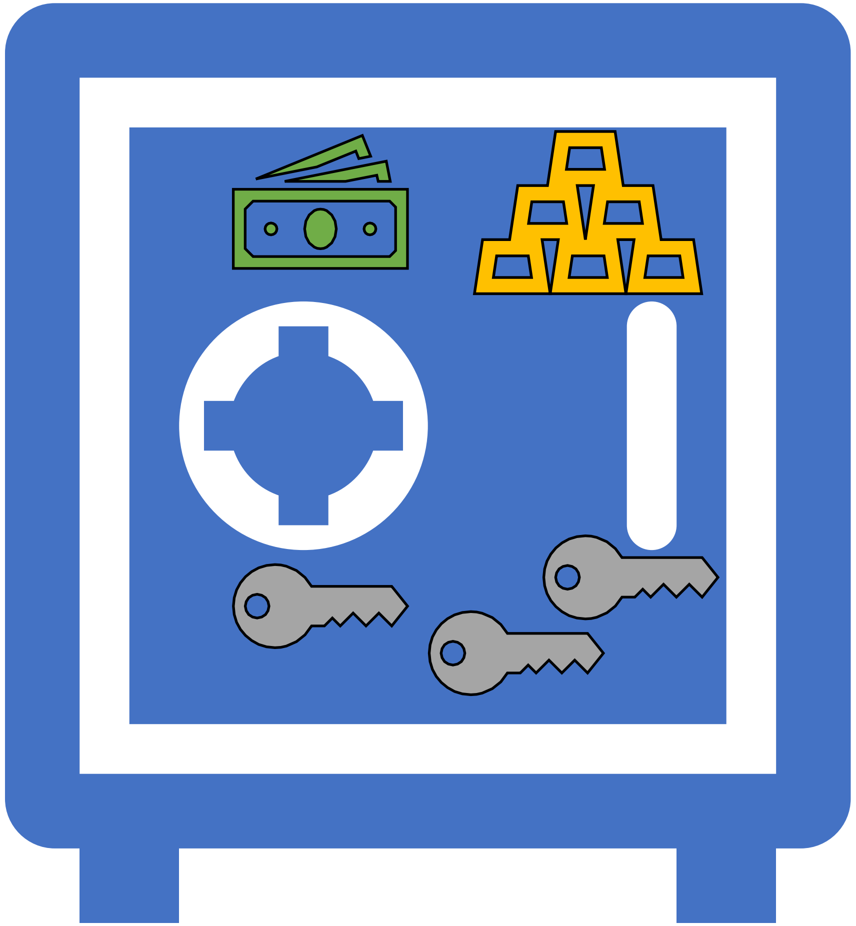
Trusted Platform Module (TPM)

- Security Chip for Computers?
- Tamper Resistant
- Side-Channel Resistant
- Crypto Co-processor



Trusted Platform Module (TPM)

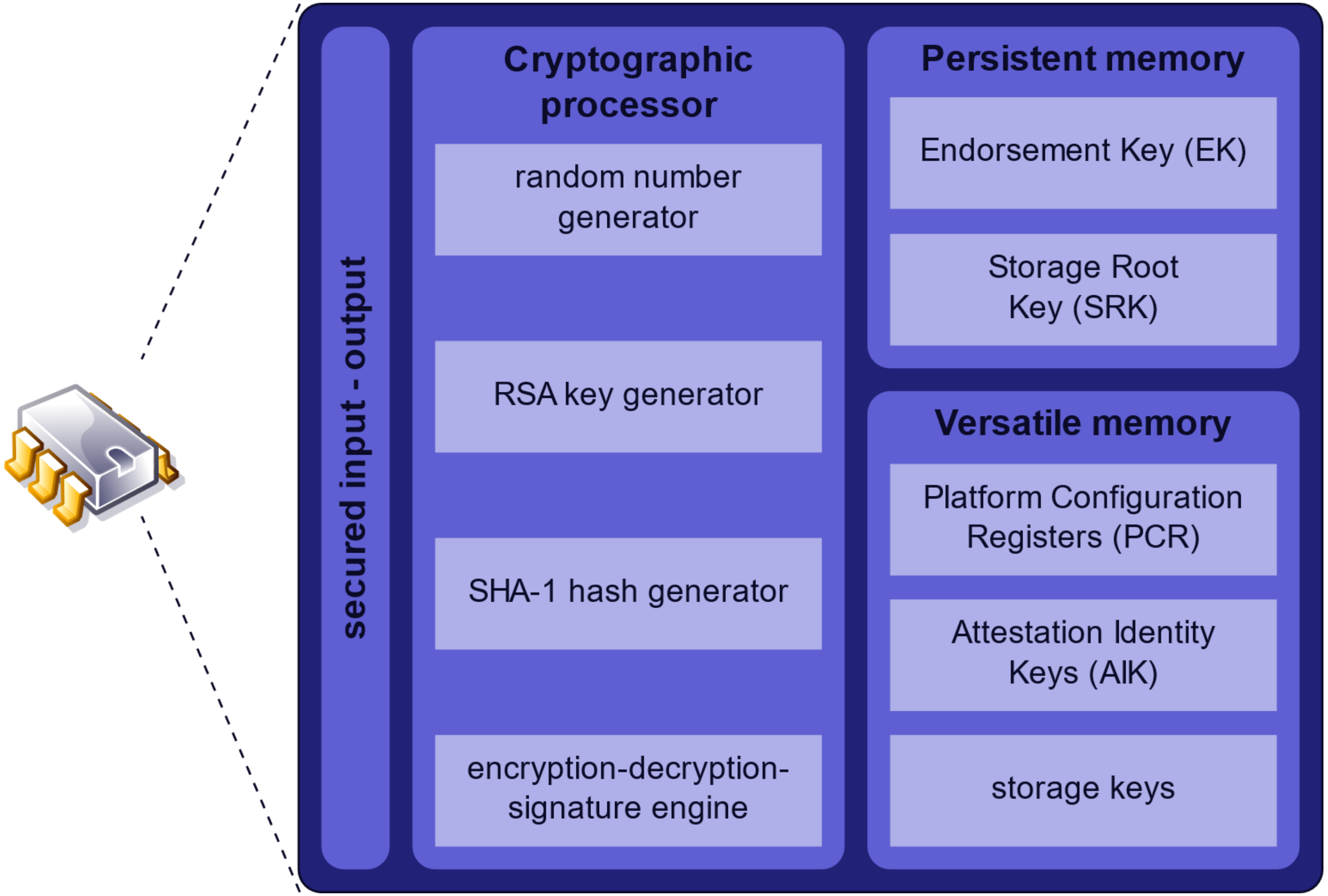
- Security Chip for Computers?
- Tamper Resistant
- Side-Channel Resistant
- Crypto Co-processor



Trusted Computing Base

Trusted Platform Module (TPM)

- Cryptographic Co-processor, specified by **Trusted Computing Group**
  - Secure Storage
  - Integrity Measurement
  - TRNG
  - Hash Functions
  - Encryption
  - **Digital Signatures**



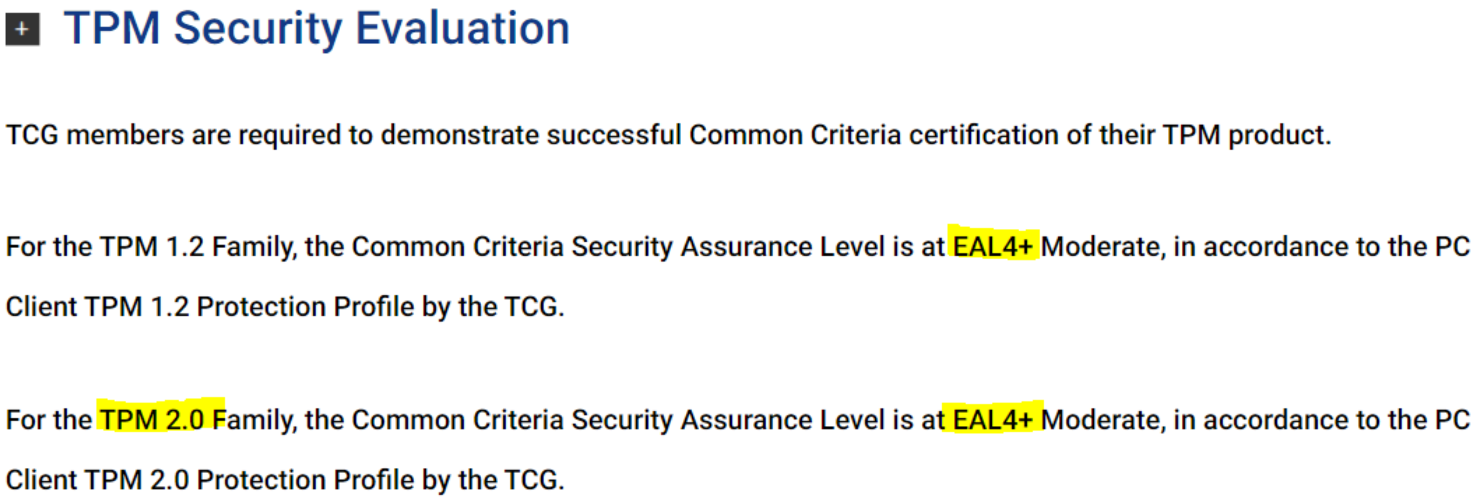
TPM – Digital Signatures

- Applications
  - Trusted Execution of Signing Operations
  - Remote Attestation
- TPM 2.0 supports Elliptic-Curve Digital Signature
  - ECDSA
  - ECSchnorr
  - ECDAA (Anonymous Remote Attestation)

Trusted Computing Group

- https://trustedcomputinggroup.org/membership/certification/

- https://trustedcomputinggroup.org/membership/certification/tpm-certified-products/

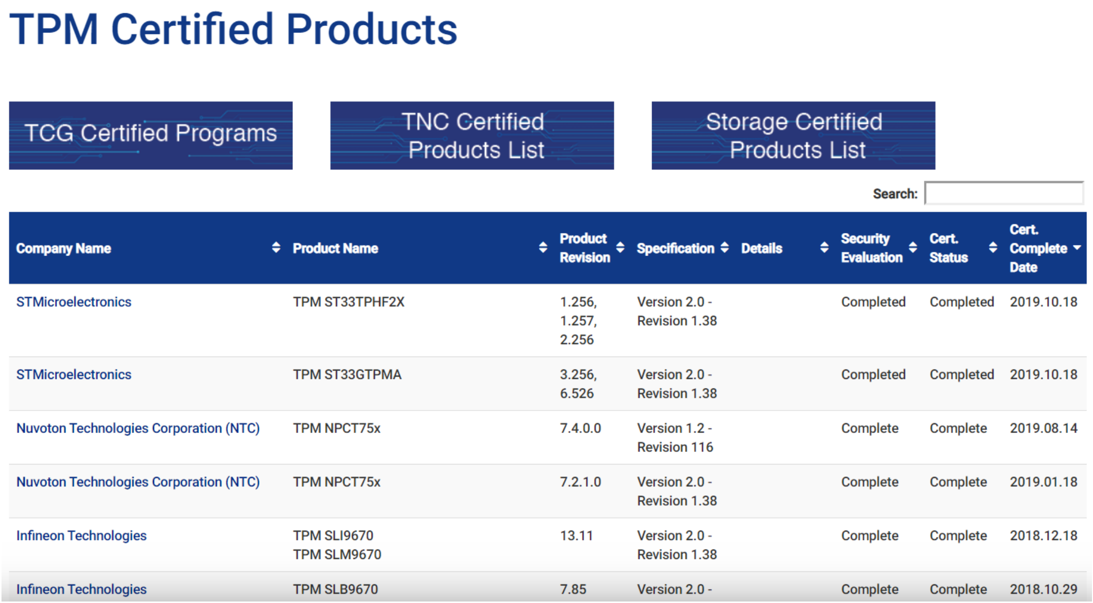


TPM Security Evaluation

TCG members are required to demonstrate successful Common Criteria certification of their TPM product.

For the TPM 1.2 Family, the Common Criteria Security Assurance Level is at EAL4+ Moderate, in accordance to the PC Client TPM 1.2 Protection Profile by the TCG.

For the TPM 2.0 Family, the Common Criteria Security Assurance Level is at EAL4+ Moderate, in accordance to the PC Client TPM 2.0 Protection Profile by the TCG.

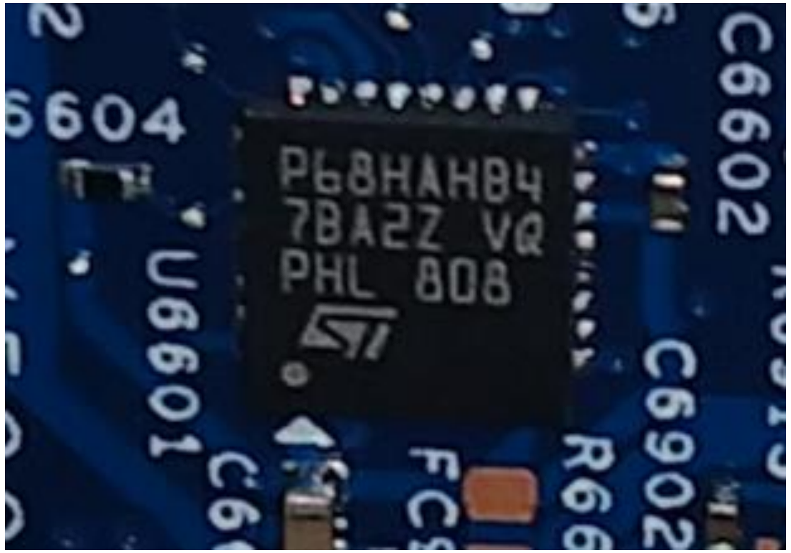


TPM Certified Products

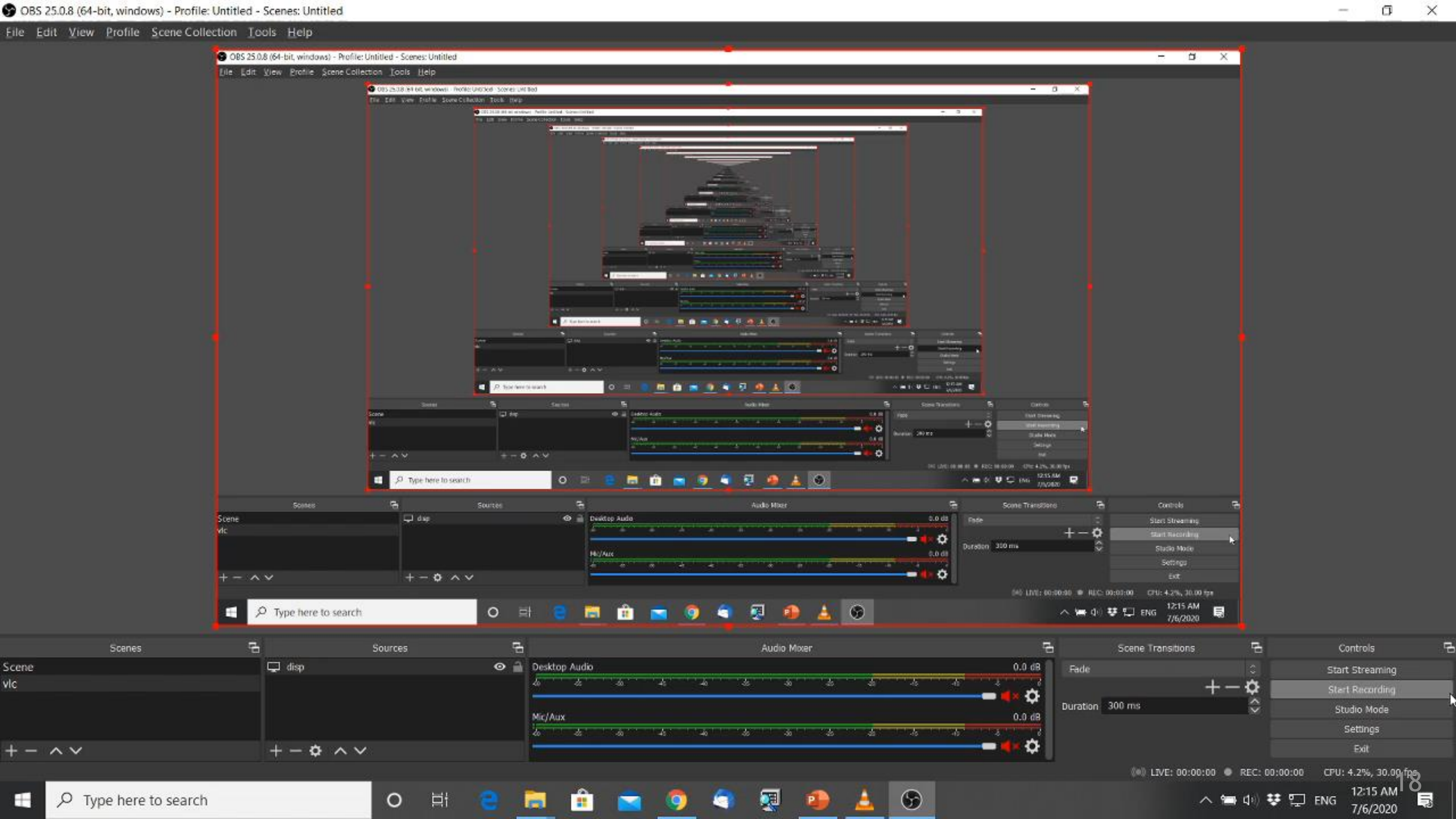
Company Name	Product Name	Product Revision	Specification	Details	Security Evaluation	Cert. Status	Cert. Complete Date
STMicroelectronics	TPM ST33TPHF2X	1.256, 1.257, 2.256	Version 2.0 - Revision 1.38		Completed	Completed	2019.10.18
STMicroelectronics	TPM ST33GTPMA	3.256, 6.526	Version 2.0 - Revision 1.38		Completed	Completed	2019.10.18
Nuvoton Technologies Corporation (NTC)	TPM NPCT75x	7.4.0.0	Version 1.2 - Revision 116		Complete	Complete	2019.08.14
Nuvoton Technologies Corporation (NTC)	TPM NPCT75x	7.2.1.0	Version 2.0 - Revision 1.38		Complete	Complete	2019.01.18
Infineon Technologies	TPM SLI9670 TPM SLM9670	13.11	Version 2.0 - Revision 1.38		Complete	Complete	2018.12.18
Infineon Technologies	TPM SLB9670	7.85	Version 2.0 -		Complete	Complete	2018.10.29

STMicroelectronics ST33TPHF2ESPI

- ST33TPHF2ESPI Data Brief
  - https://www.st.com/resource/en/data_brief/st33tphf2espi.pdf



- ST33TPHF2ESPI CC Evaluation
  - https://www.ssi.gouv.fr/uploads/2018/10/anssi-cible-cc-2018_41en.pdf



Scene
vlc

disp

Desktop Audio 0.0 dB

Mic/Aux 0.0 dB

Fade

Duration 300 ms

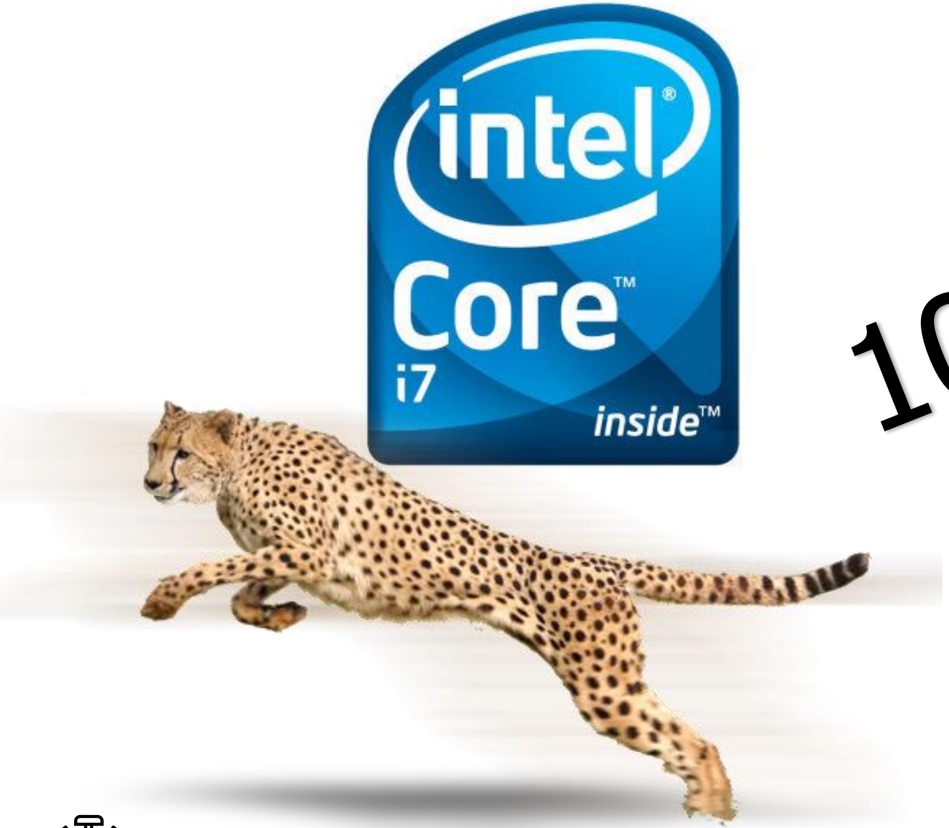
LIVE: 00:00:00 REC: 00:00:00 CPU: 4.2%, 30.00 fps

12:15 AM
7/6/2020

Are TPMs really side-channel resistant?

High-resolution Timing Test

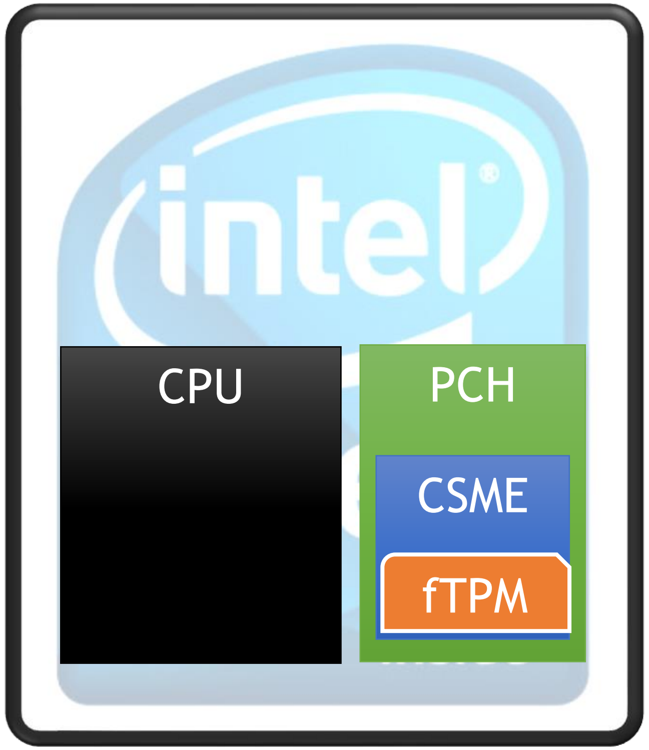
- TPM frequency ~= 32-120 MHz
- CPU Frequency is more than 2 GHz



100x faster!!

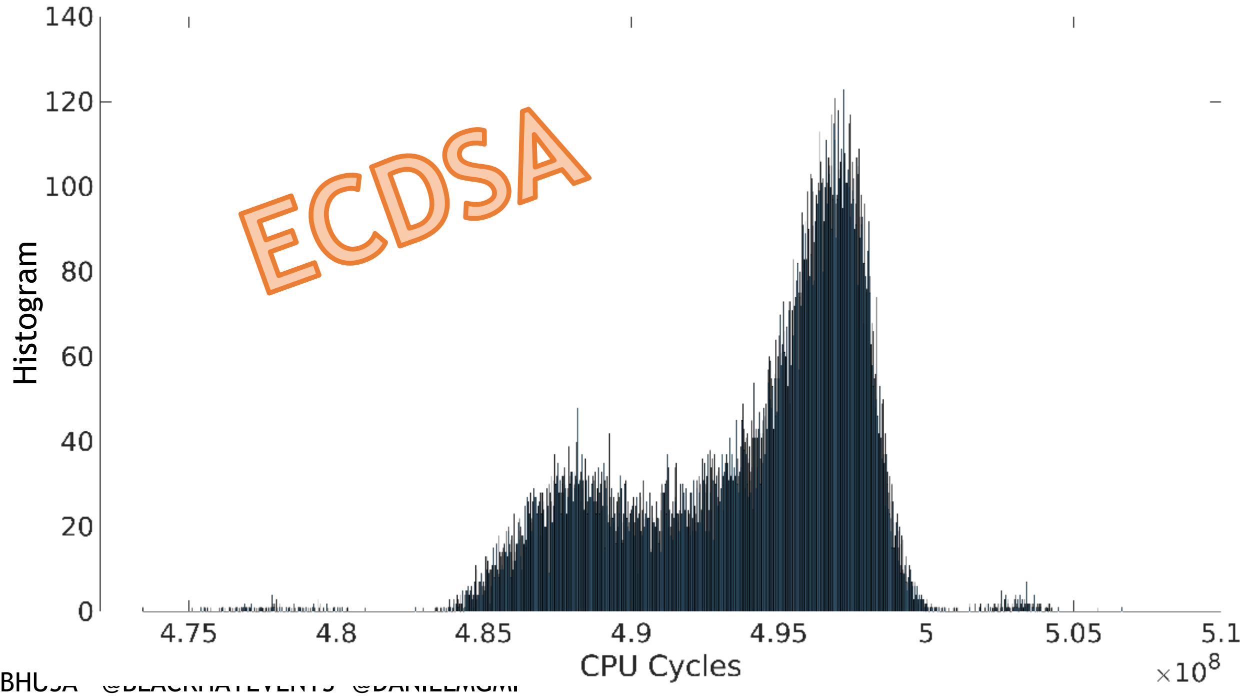
High-resolution Timing Test – Intel PTT (fTPM)

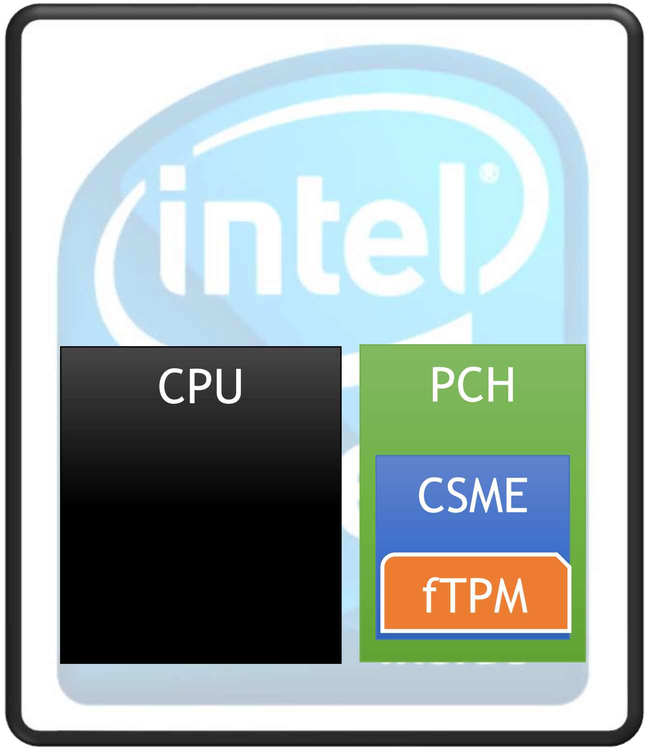
- Intel Platform Trust Technology (PTT)
  - Integrated firmware-TPM inside the CPU package
  - Runs on top of Converged Security and Management Engine (CSME)
  - Standalone low power processor
  - Has been around since Haswell



High-resolution Timing Test – Intel PTT (fTPM)

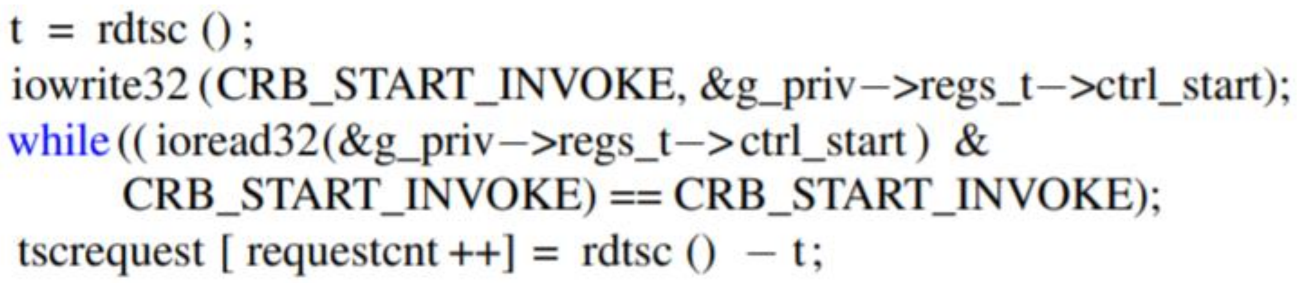
- Intel Platform Trust Technology (PTT)
  - Integrated firmware-TPM inside the CPU package
  - Runs on top of Converged Security and Management Engine (CSME)



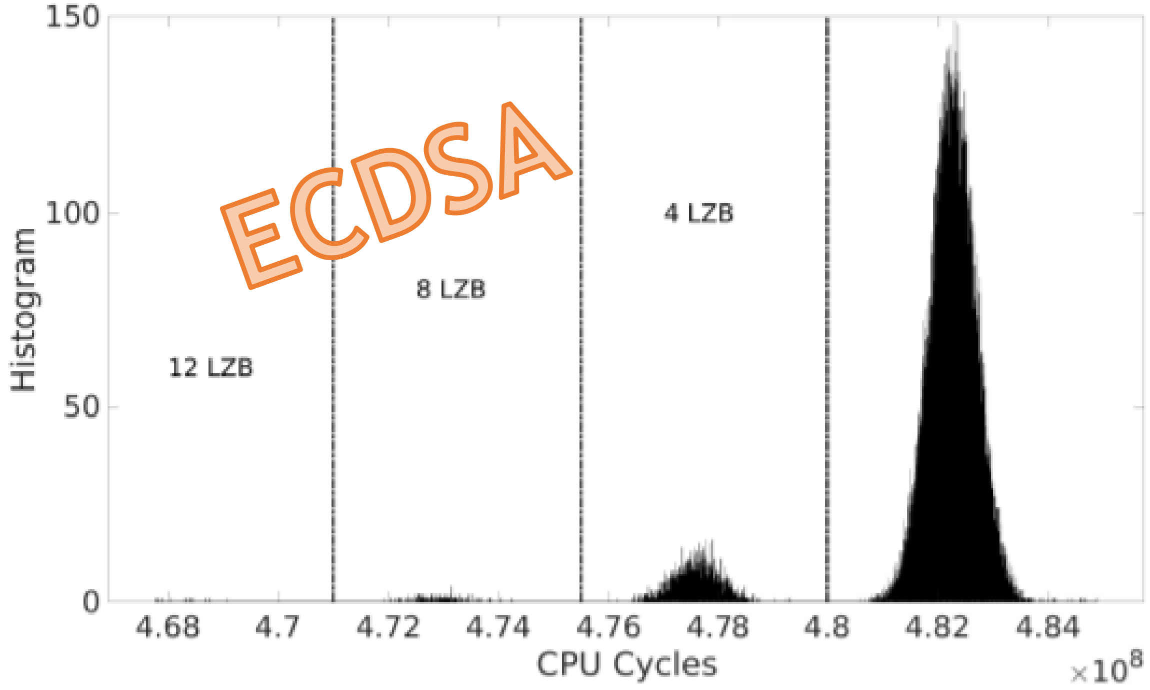


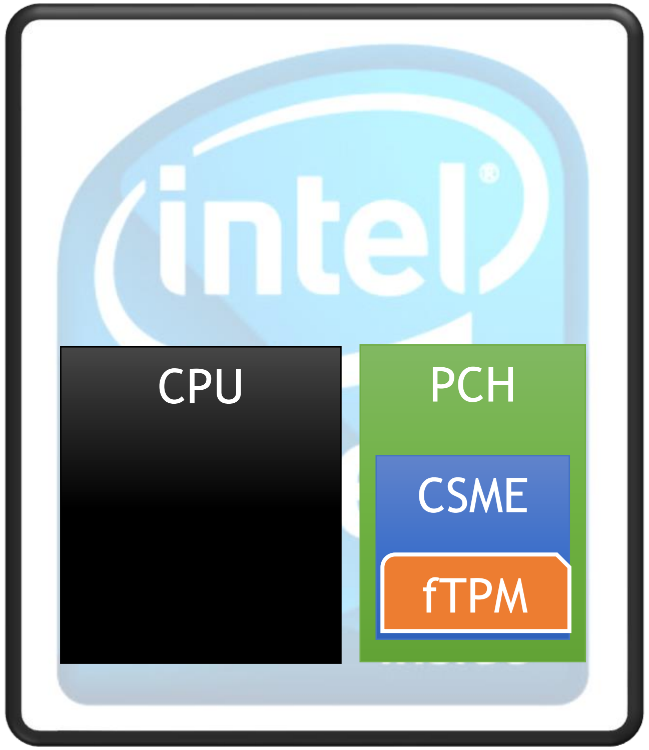
High-resolution Timing Test – Intel PTT (fTPM)

- Linux TPM Command Response Buffer (CRB) driver
- Kernel Driver to increase the Resolution



```
t = rdtsc ();
iowrite32 (CRB_START_INVOKE, &g_priv->regs_t->ctrl_start);
while (( ioread32(&g_priv->regs_t->ctrl_start ) &
        CRB_START_INVOKE) == CRB_START_INVOKE);
tscrequest [ requestcnt ++] = rdtsc () - t;
```





CPU

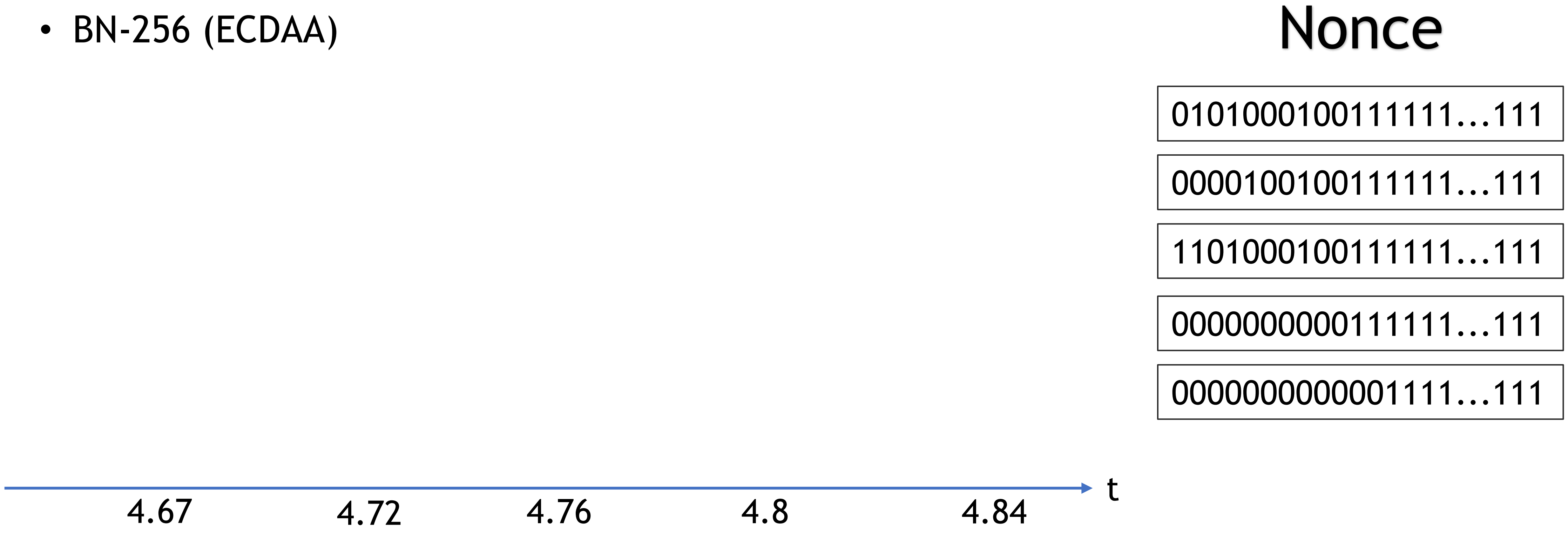
PCH

CSME

fTPM

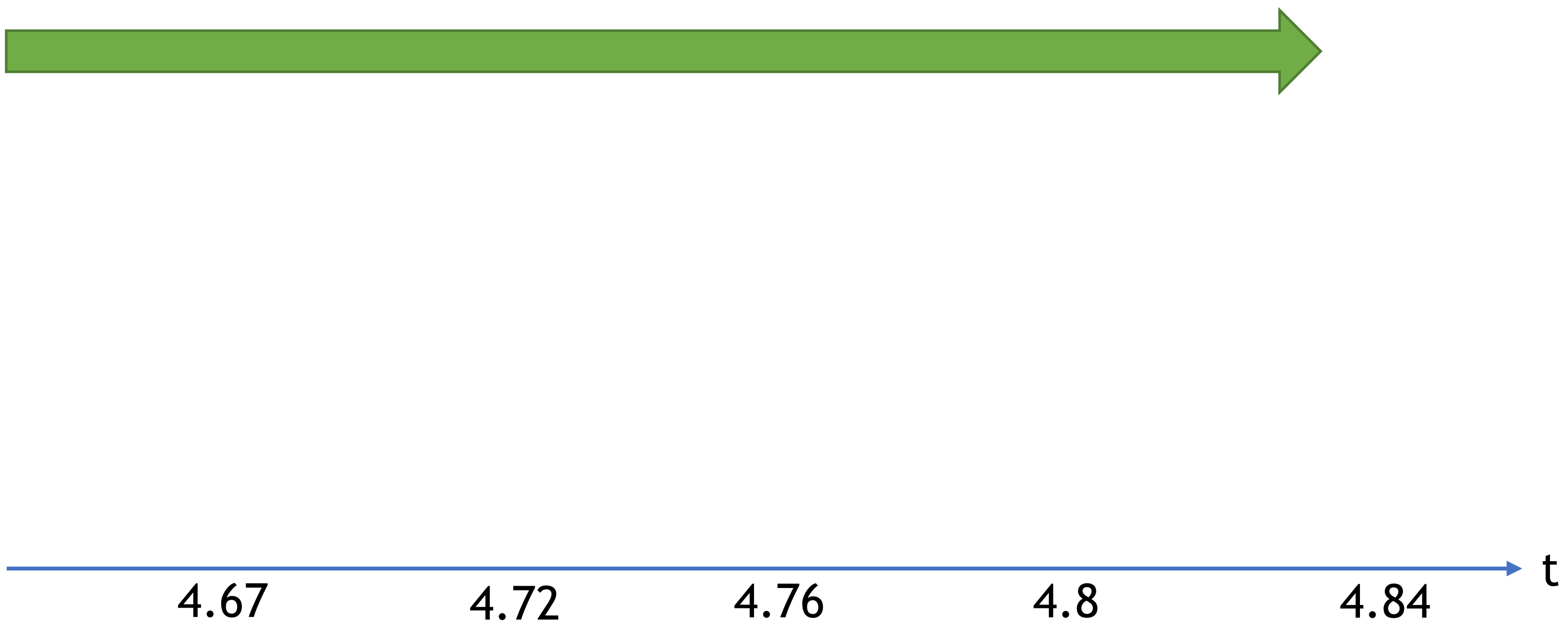
High-resolution Timing Test – ECDSA Nonce

- Intel fTPM: 4-bit Window Nonce Length Leakage
  - ECDSA
  - ECSChnorr
  - BN-256 (ECDAA)



High-resolution Timing Test – ECDSA Nonce

- Intel fTPM: 4-bit Window Nonce Length Leakage
  - ECDSA
  - ECSChnorr
  - BN-256 (ECDAA)



Nonce

0101000100111111...111

0000100100111111...111

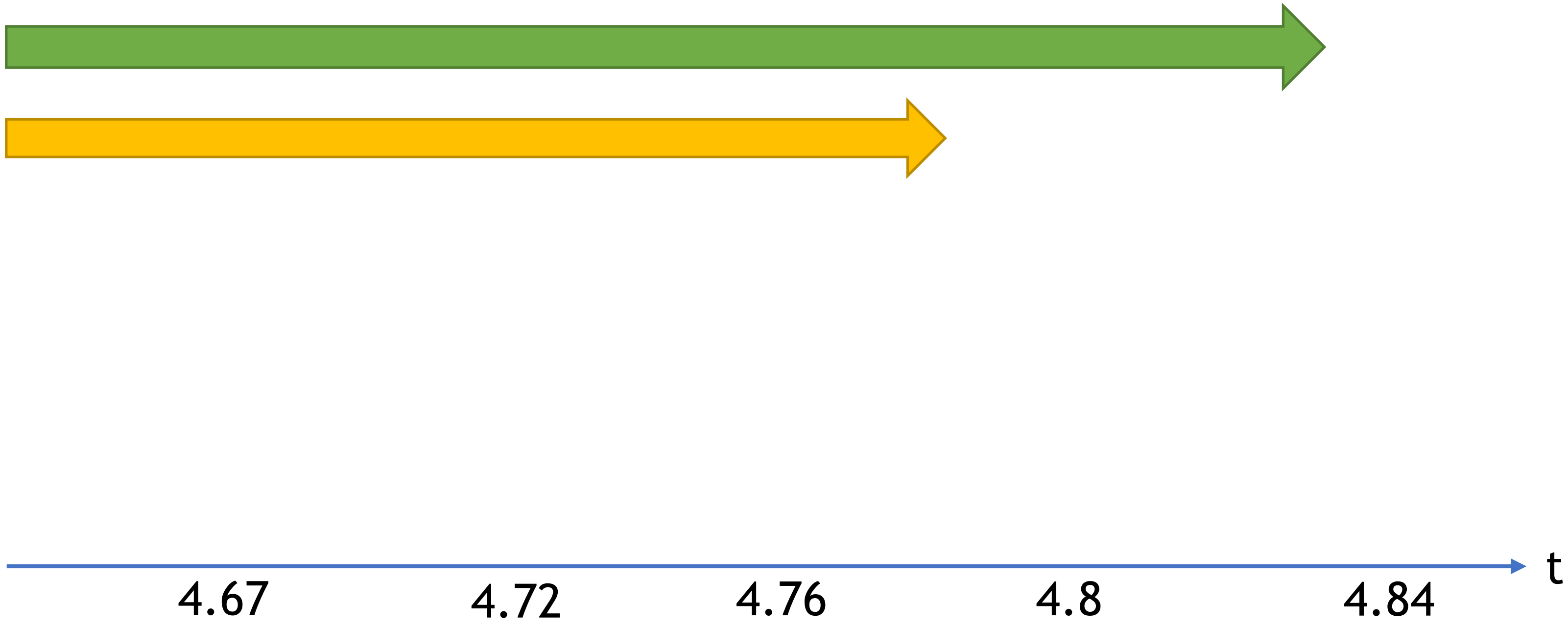
1101000100111111...111

0000000000111111...111

0000000000001111...111

High-resolution Timing Test – ECDSA Nonce

- Intel fTPM: 4-bit Window Nonce Length Leakage
  - ECDSA
  - ECSChnorr
  - BN-256 (ECDAA)

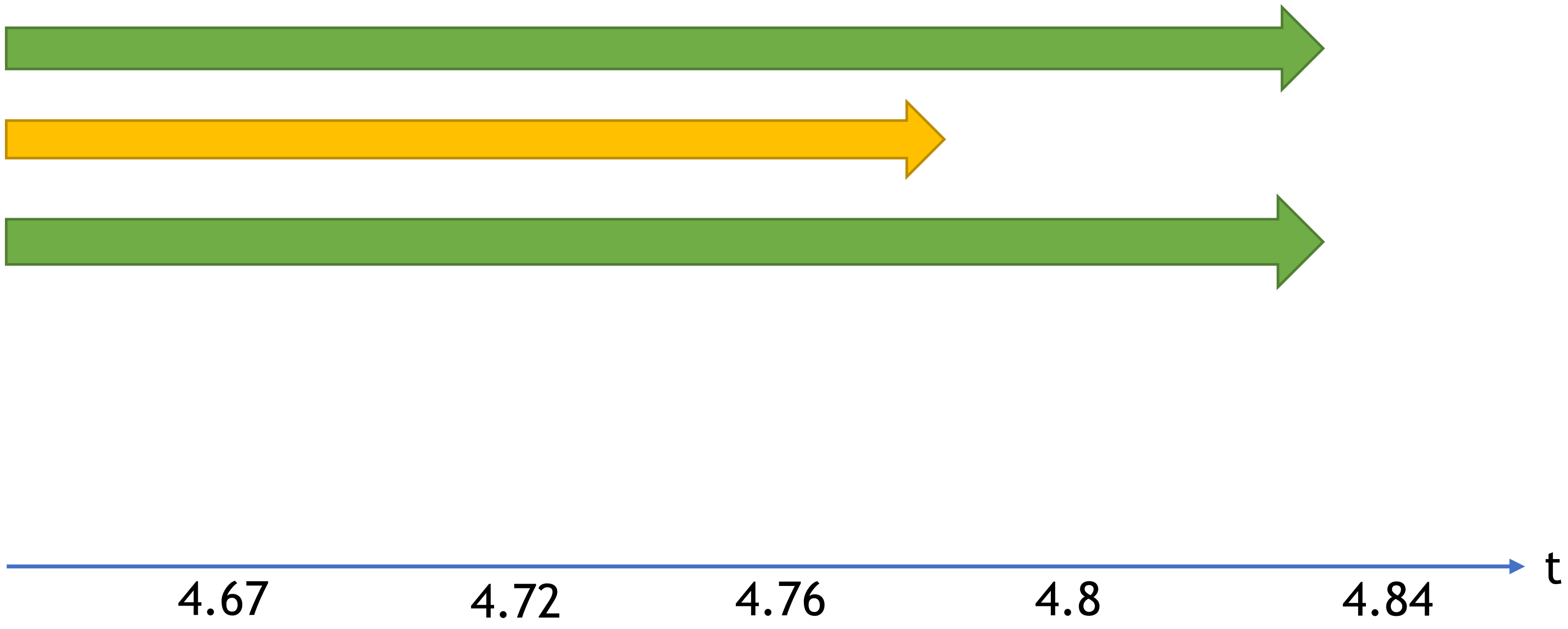


Nonce

0101000100111111...111
0000100100111111...111
1101000100111111...111
0000000000111111...111
0000000000001111...111

High-resolution Timing Test – ECDSA Nonce

- Intel fTPM: 4-bit Window Nonce Length Leakage
  - ECDSA
  - ECSChnorr
  - BN-256 (ECDAA)



Nonce

010100010011111...111

000010010011111...111

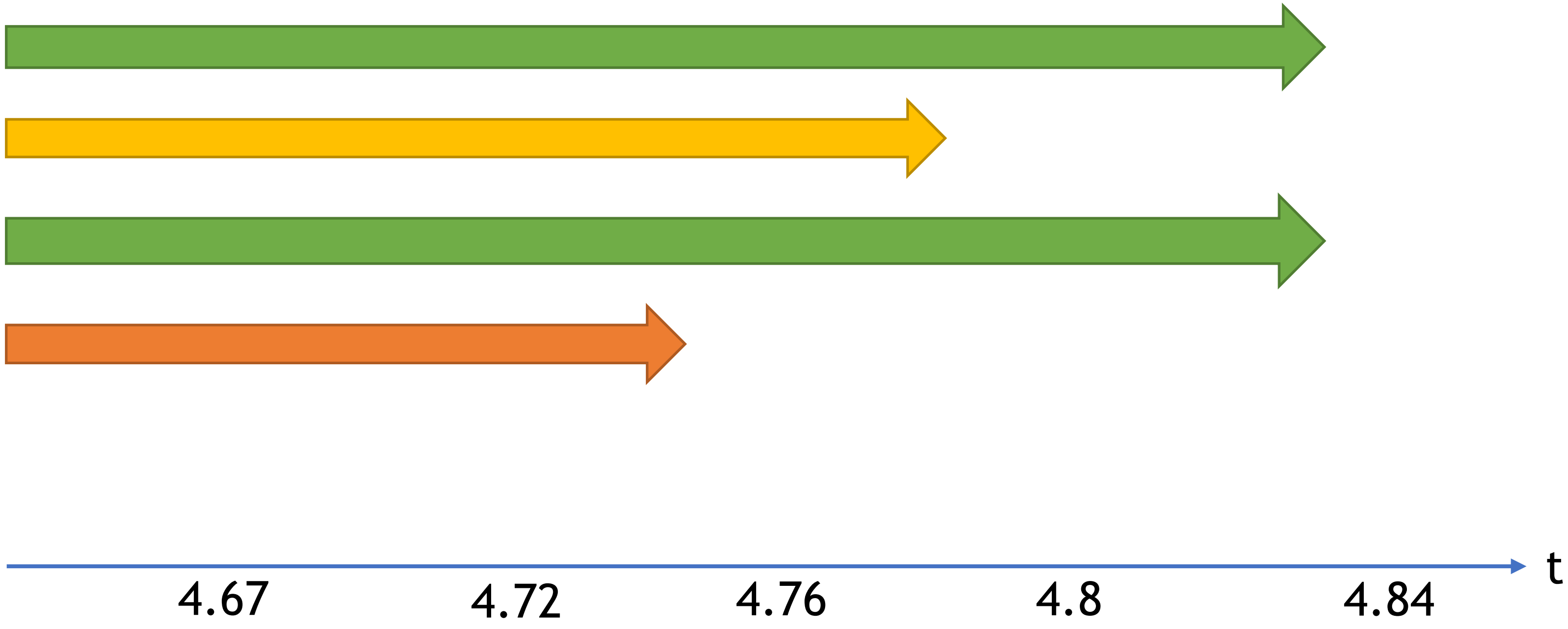
110100010011111...111

000000000111111...111

000000000001111...111

High-resolution Timing Test - ECDSA Nonce

- Intel fTPM: 4-bit Window Nonce Length Leakage
  - ECDSA
  - ECSChnorr
  - BN-256 (ECDAA)



Nonce

01010001001111111...111

00001001001111111...111

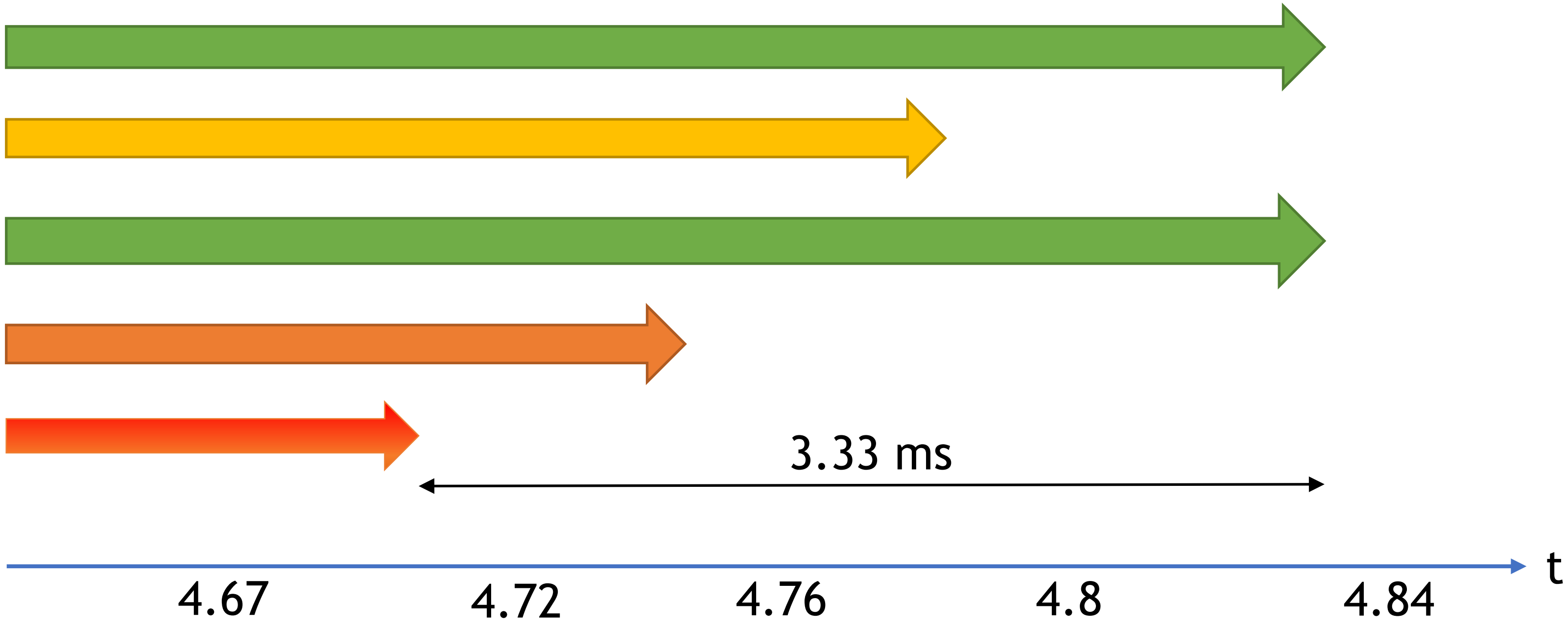
11010001001111111...111

00000000001111111...111

00000000000001111...111

High-resolution Timing Test – ECDSA Nonce

- Intel fTPM: 4-bit Window Nonce Length Leakage
  - ECDSA
  - ECSChnorr
  - BN-256 (ECDAA)



Nonce

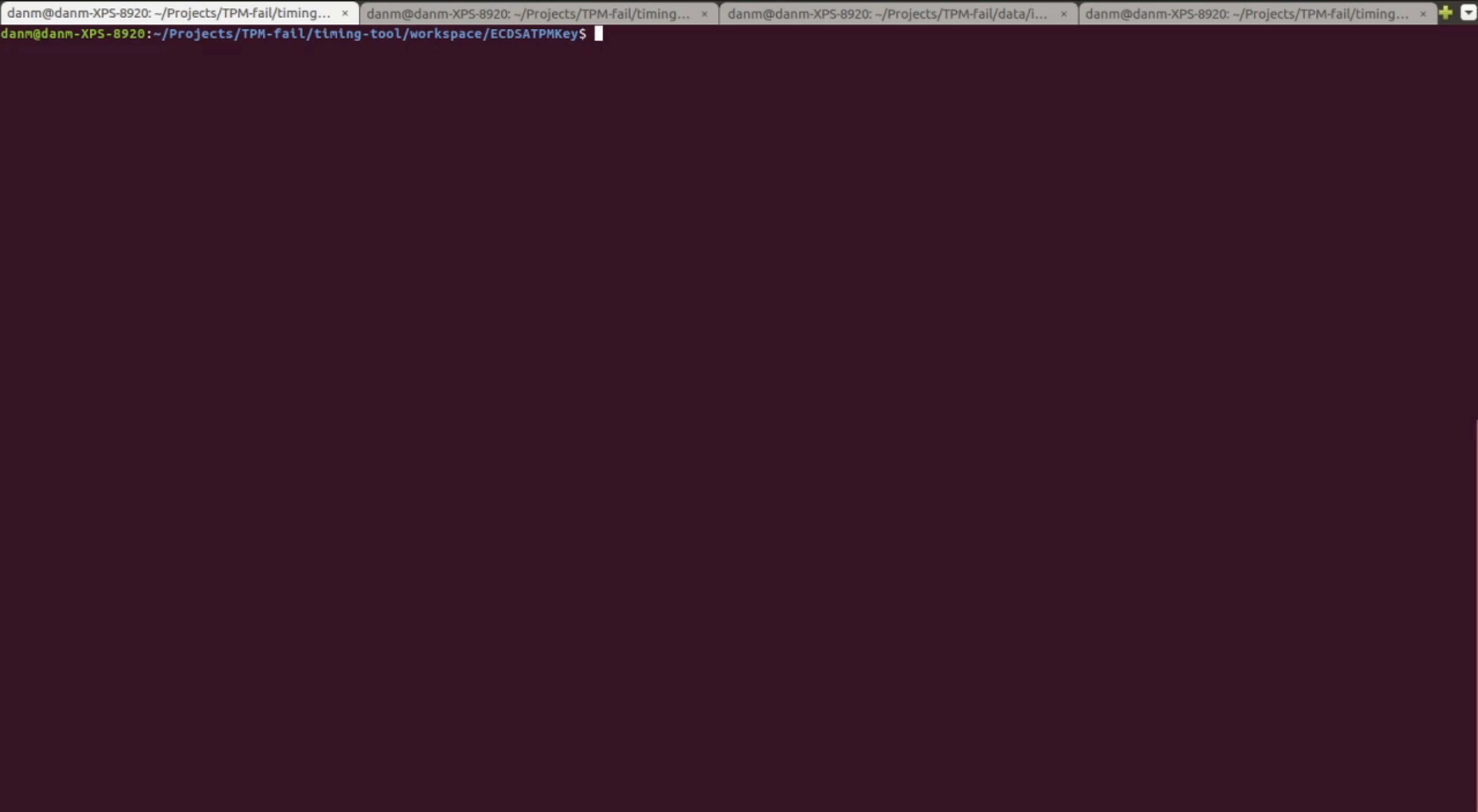
0101000100111111...111

0000100100111111...111

1101000100111111...111

0000000000111111...111

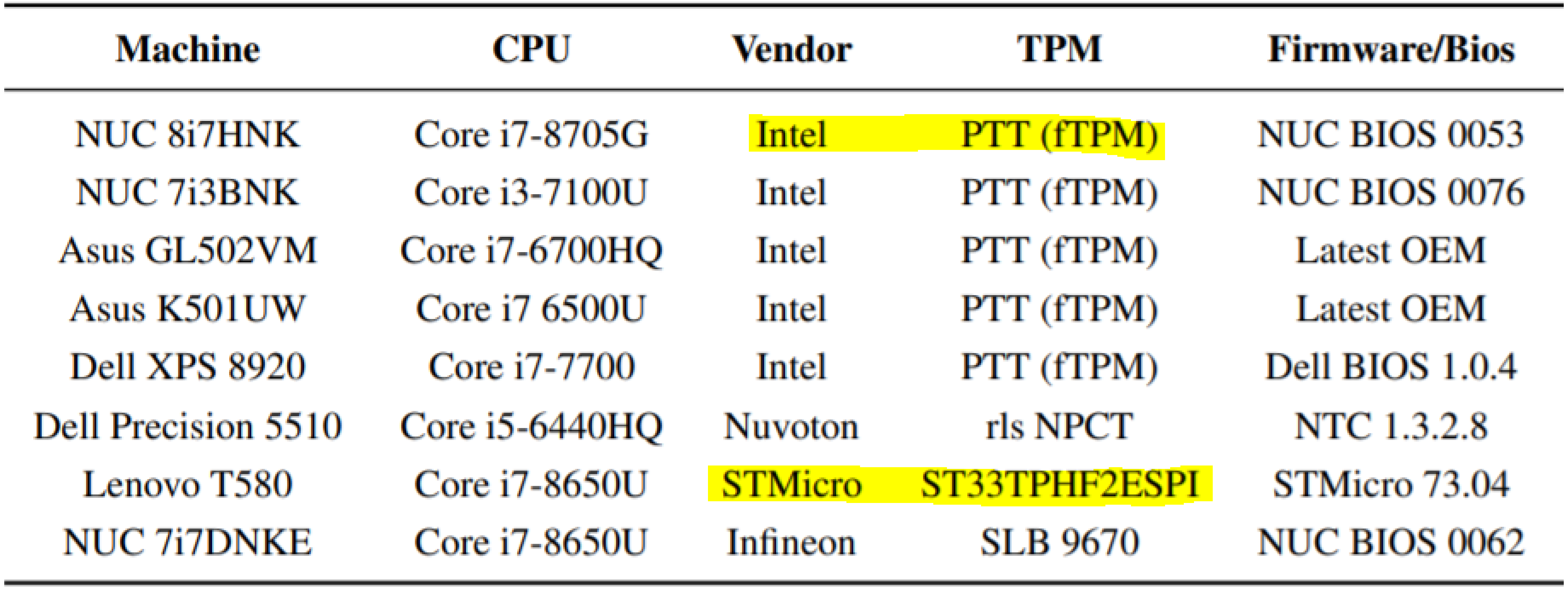
0000000000001111...111


danm@danm-XPS-8920: ~/Projects/TPM-fail/timing...
danm@danm-XPS-8920: ~/Projects/TPM-fail/timing...
danm@danm-XPS-8920: ~/Projects/TPM-fail/data/i...
danm@danm-XPS-8920: ~/Projects/TPM-fail/timing...

```
danm@danm-XPS-8920:~/Projects/TPM-fail/timing-tool/workspace/ECDSATPMKey$ 
```

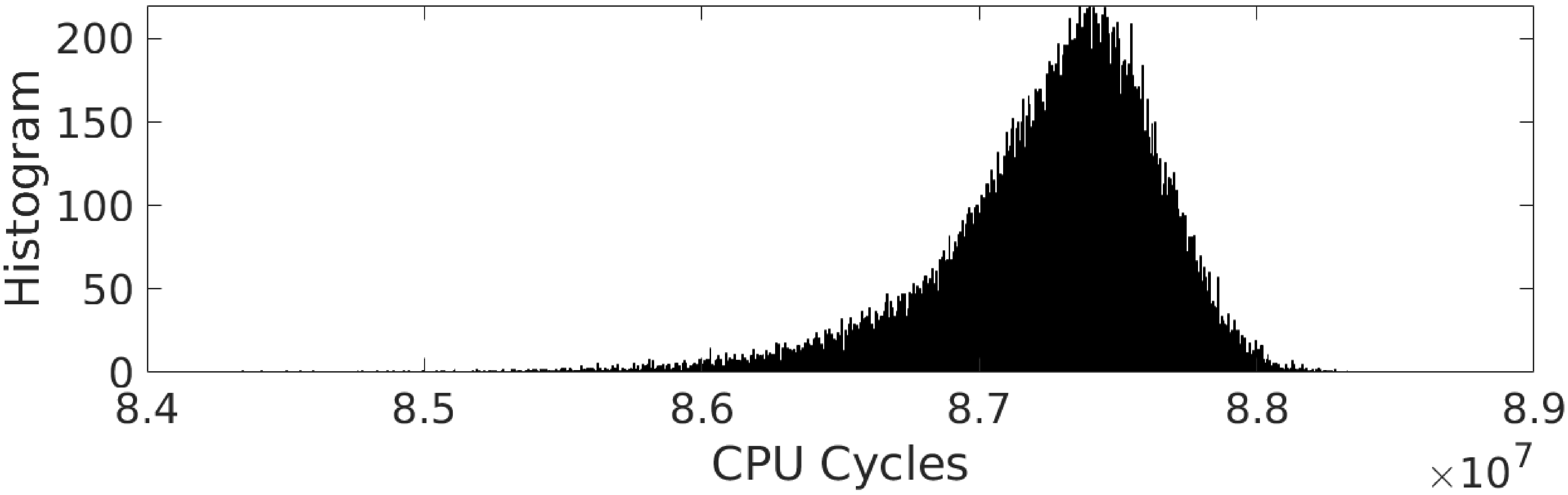
High-resolution Timing Test - Analysis

- RSA and ECDSA timing test on 3 dedicated TPM and Intel fTPM
- Various non-constant behaviour for both RSA and ECDSA



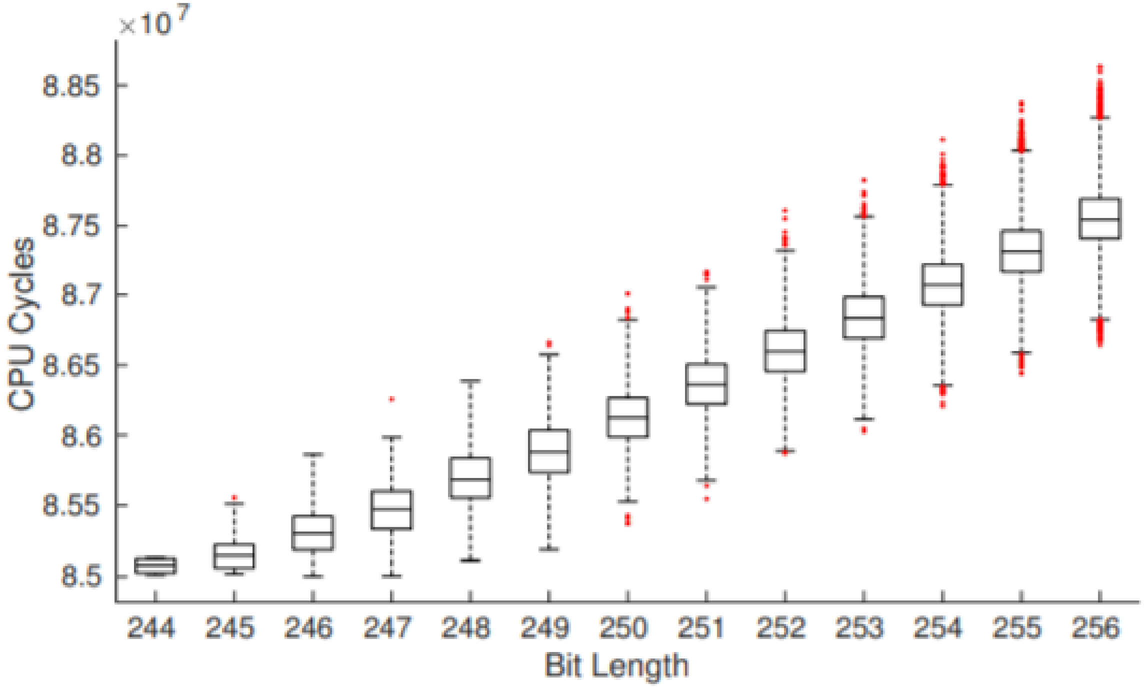
Machine	CPU	Vendor	TPM	Firmware/Bios
NUC 8i7HNK	Core i7-8705G	Intel	PTT (fTPM)	NUC BIOS 0053
NUC 7i3BNK	Core i3-7100U	Intel	PTT (fTPM)	NUC BIOS 0076
Asus GL502VM	Core i7-6700HQ	Intel	PTT (fTPM)	Latest OEM
Asus K501UW	Core i7 6500U	Intel	PTT (fTPM)	Latest OEM
Dell XPS 8920	Core i7-7700	Intel	PTT (fTPM)	Dell BIOS 1.0.4
Dell Precision 5510	Core i5-6440HQ	Nuvoton	rls NPCT	NTC 1.3.2.8
Lenovo T580	Core i7-8650U	STMicro	ST33TPHF2ESPI	STMicro 73.04
NUC 7i7DNKE	Core i7-8650U	Infineon	SLB 9670	NUC BIOS 0062

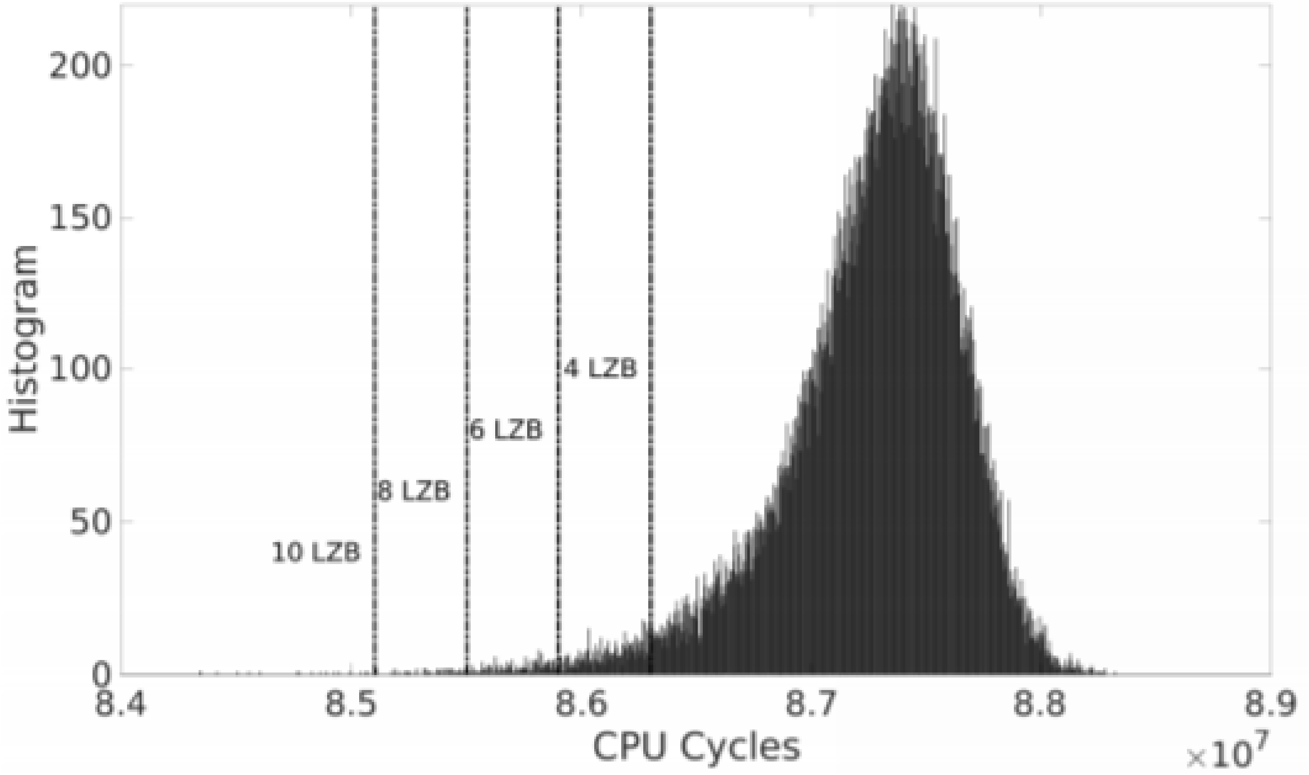
STMicro - ECDSA



High-resolution Timing Test – ECDSA Nonce

- STMicro TPM: Bit-by-Bit Nonce Length Leakage





TPM-Fail – Recovering Private ECDSA Key

- TPM is programmed with an unknown key
- We already have a template for t_i.

1. Collect list of signatures (r_i, s_i) and timing samples t_i.
2. Filter signatures based on t_i and keeps (r_i, s_i) with a known bias.
3. Lattice-based attack to recover private key d, from signatures with biased nonce k_i.

Lattice and Hidden Number Problem

- $s = k^{-1}(z + dr) \; mod \; n$

Lattice and Hidden Number Problem

- $s = k^{-1}(z + dr)\ mod\ n \rightarrow\ k_i^{-1}\ - s_i^{-1}r_i d - s_i^{-1}z \equiv 0\ mod\ n$

Lattice and Hidden Number Problem

- $s = k^{-1}(z + dr)\ mod\ n \rightarrow\ k_i^{-1}\ - s_i^{-1}r_i d - s_i^{-1}z \equiv 0\ mod\ n$
- $A_i = -s_i^{-1}r_i, B_i = -s_i^{-1}z\ \rightarrow k_i + A_i d + B_i = 0$

Lattice and Hidden Number Problem

- $s = k^{-1}(z + dr)\ mod\ n \rightarrow\ k_i^{-1}\ - s_i^{-1} r_i d - s_i^{-1} z \equiv 0\ mod\ n$
- $A_i = -s_i^{-1} r_i, B_i = -s_i^{-1} z\ \rightarrow k_i + A_i d + B_i = 0$
- Let X be the upper bound on $\mathrm{k_i}$ and $(d, k_{0,} k_1 \ldots, k_n)$ is unknown

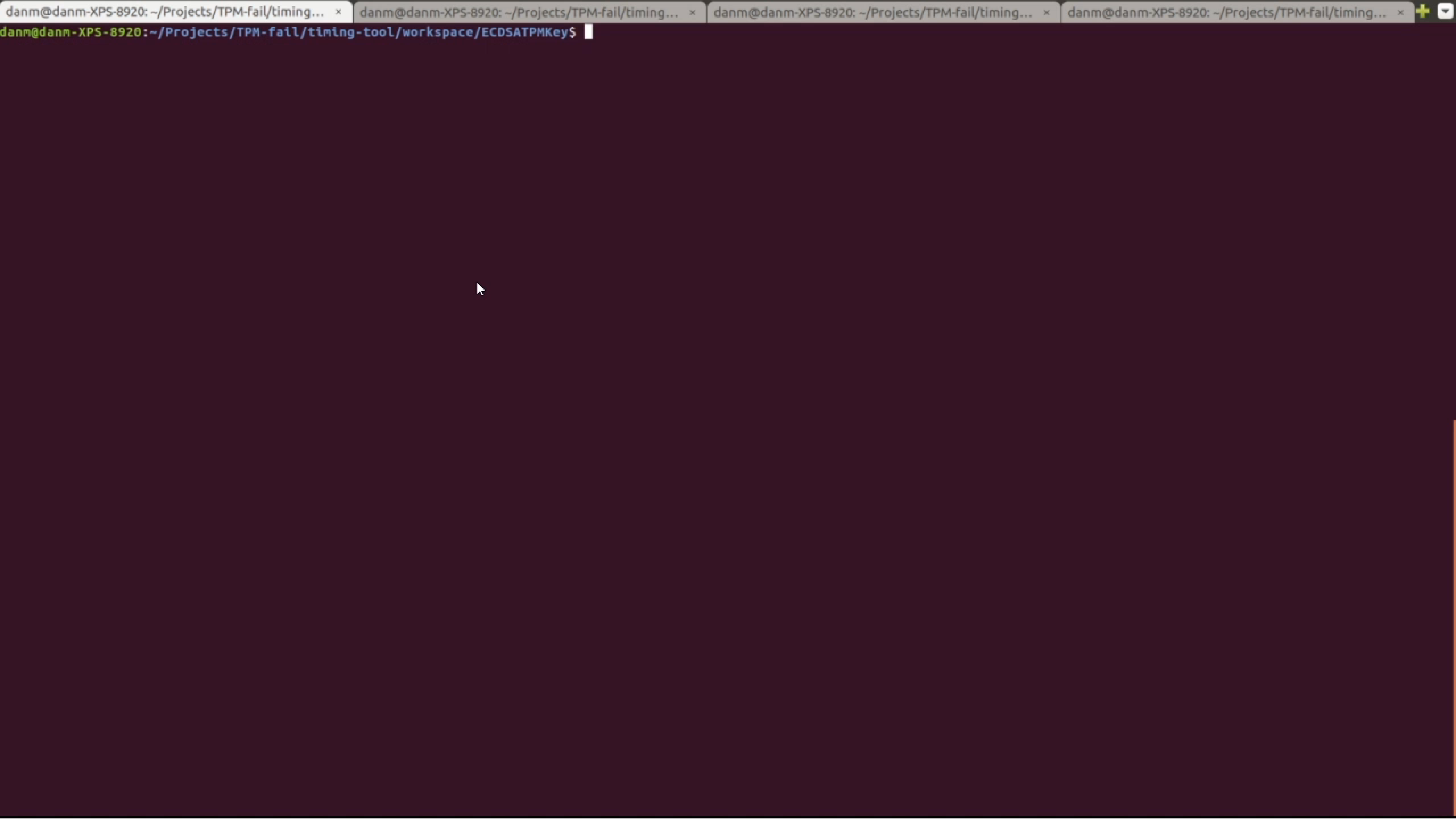
Boneh and Venkatesan[1]

[1] Dan Boneh and Ramarathnam Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes

Lattice and Hidden Number Problem

- $s = k^{-1}(z + dr) \, mod \, n \rightarrow k_i^{-1} - s_i^{-1} r_i d - s_i^{-1} z \equiv 0 \, mod \, n$
- $A_i = -s_i^{-1} r_i, B_i = -s_i^{-1} z \rightarrow k_i + A_i d + B_i = 0$
- Let X be the upper bound on $\mathrm{k_i}$ and $(d, k_{0,} k_1 \dots, k_n)$ is unknown
- Lattice Construction:

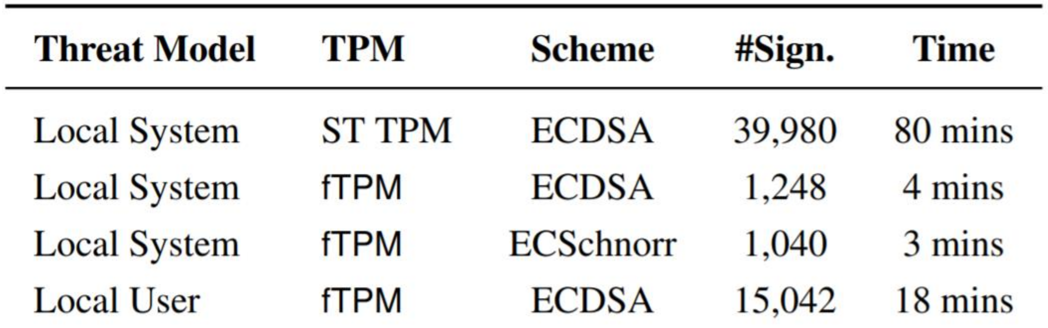
$$\begin{bmatrix} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ A_1 & A_2 & \dots & A_t & \frac{X}{n} & \\ B_1 & B_2 & \dots & B_t & & X \end{bmatrix}$$


danm@danm-XPS-8920: ~/Projects/TPM-fail/timing...
danm@danm-XPS-8920: ~/Projects/TPM-fail/timing...
danm@danm-XPS-8920: ~/Projects/TPM-fail/timing...
danm@danm-XPS-8920: ~/Projects/TPM-fail/timing...

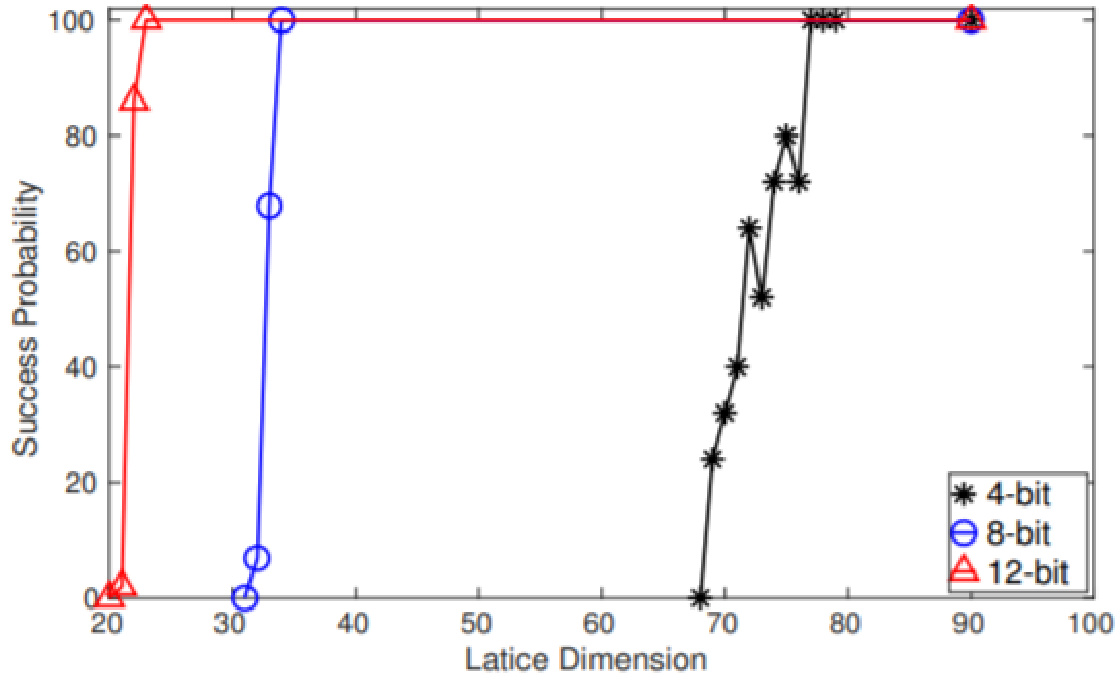
```
danm@danm-XPS-8920:~/Projects/TPM-fail/timing-tool/workspace/ECDSATPMKey$ 
```

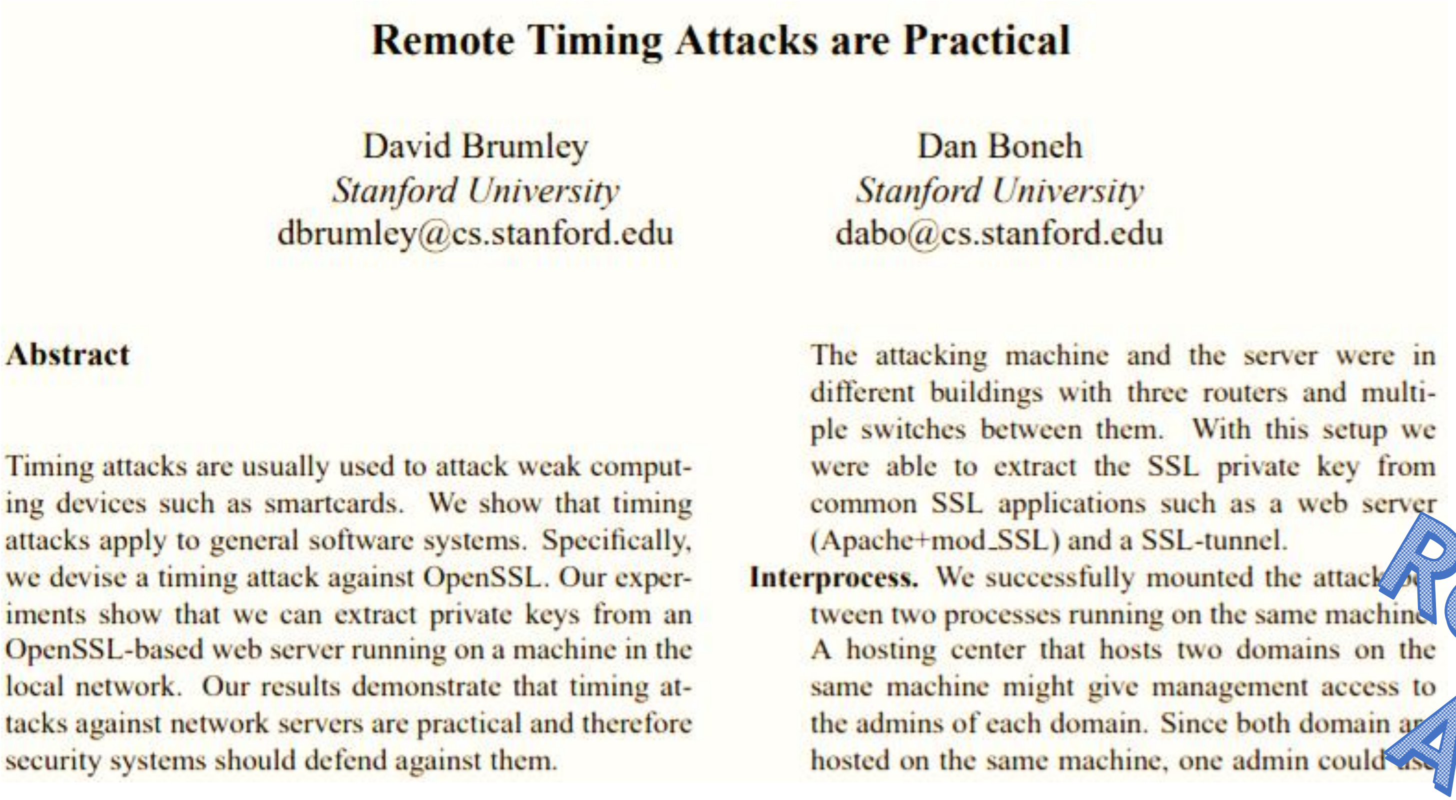
TPM-Fail – Key Recovery Results

- Intel fTPM
  - ECDSA, ECSchnorr and BN-256 (ECDAA)
  - Three different threat model System, User, Network
- STMicroelectronics TPM
  - CC EAL4+ Certified
  - Give you the key in 80 minutes



Threat Model	TPM	Scheme	#Sign.	Time
Local System	ST TPM	ECDSA	39,980	80 mins
Local System	fTPM	ECDSA	1,248	4 mins
Local System	fTPM	ECSchnorr	1,040	3 mins
Local User	fTPM	ECDSA	15,042	18 mins





Remote Timing Attacks are Practical

David Brumley
Stanford University
dbrumley@cs.stanford.edu

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Abstract

Timing attacks are usually used to attack weak computing devices such as smartcards. We show that timing attacks apply to general software systems. Specifically, we devise a timing attack against OpenSSL. Our experiments show that we can extract private keys from an OpenSSL-based web server running on a machine in the local network. Our results demonstrate that timing attacks against network servers are practical and therefore security systems should defend against them.

The attacking machine and the server were in different buildings with three routers and multiple switches between them. With this setup we were able to extract the SSL private key from common SSL applications such as a web server (Apache+mod_SSL) and a SSL-tunnel.

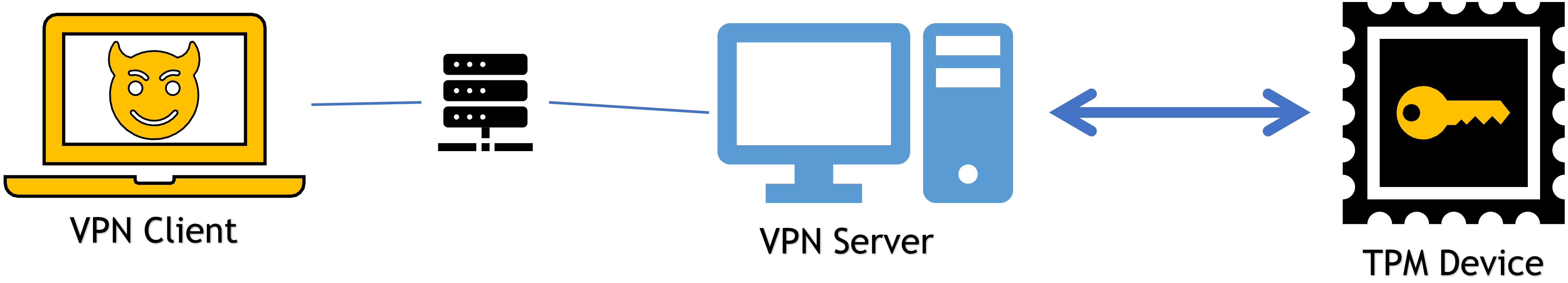
Interprocess. We successfully mounted the attack between two processes running on the same machine. A hosting center that hosts two domains on the same machine might give management access to the admins of each domain. Since both domain are hosted on the same machine, one admin could use

TPMs are extremely slow

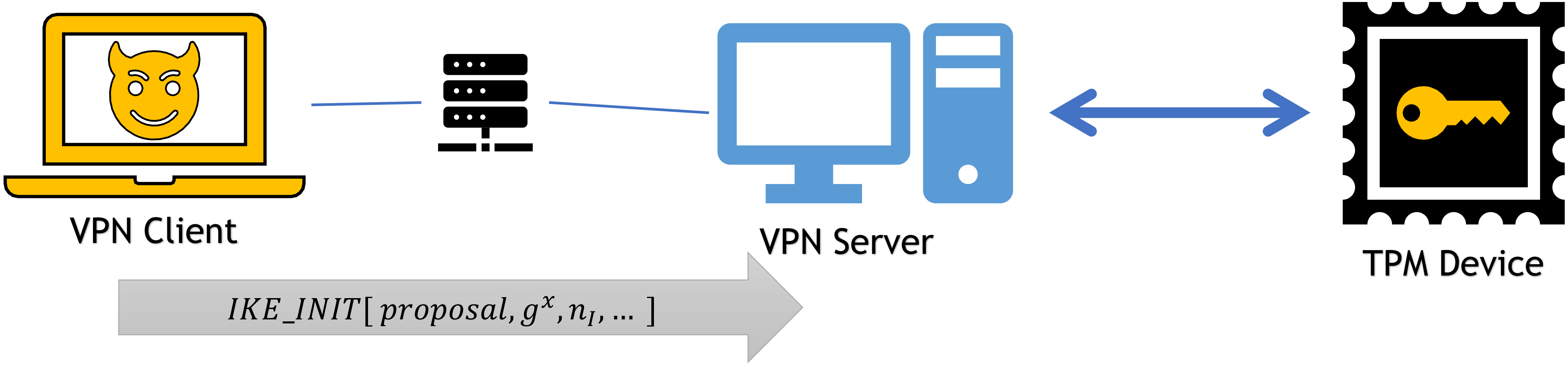
Remote Timing Attacks are Practical!!

Timing difference for each window	(4.76e8 - 4.72e8)/3600e6 * 1000 = **1.11 ms**
ping 192.168.1.x	average rtt **0.713 ms**
ping 1.1.1.1 (Cloudflare DNS)	average rtt **19.312 ms**

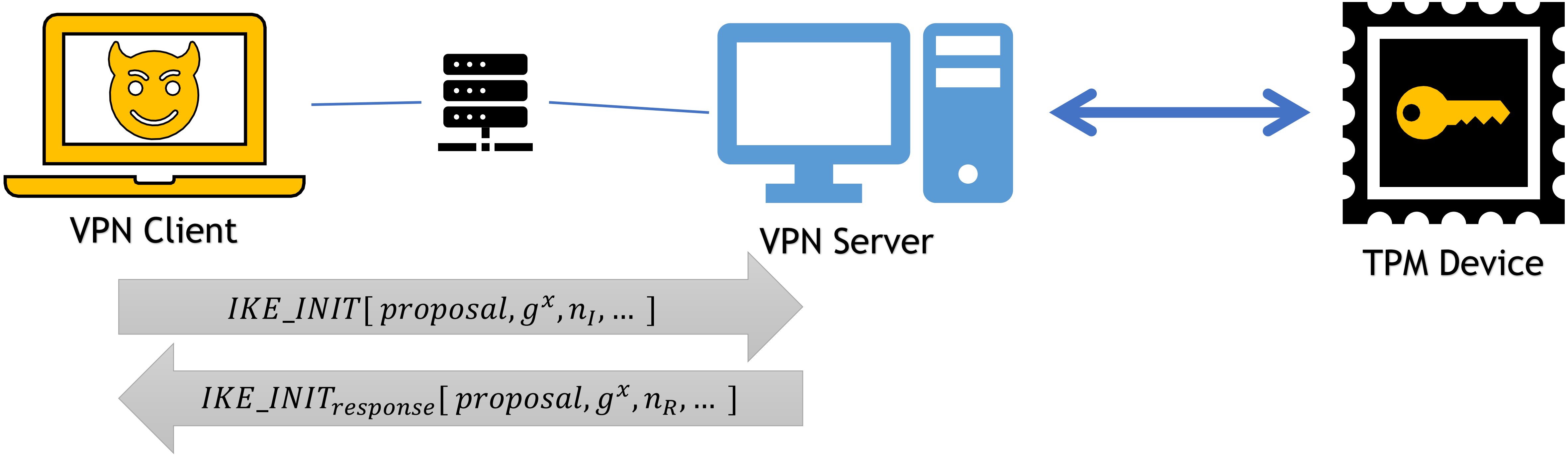
TPM-Fail Case Study: StrongSwan VPN



TPM-Fail Case Study: StrongSwan VPN

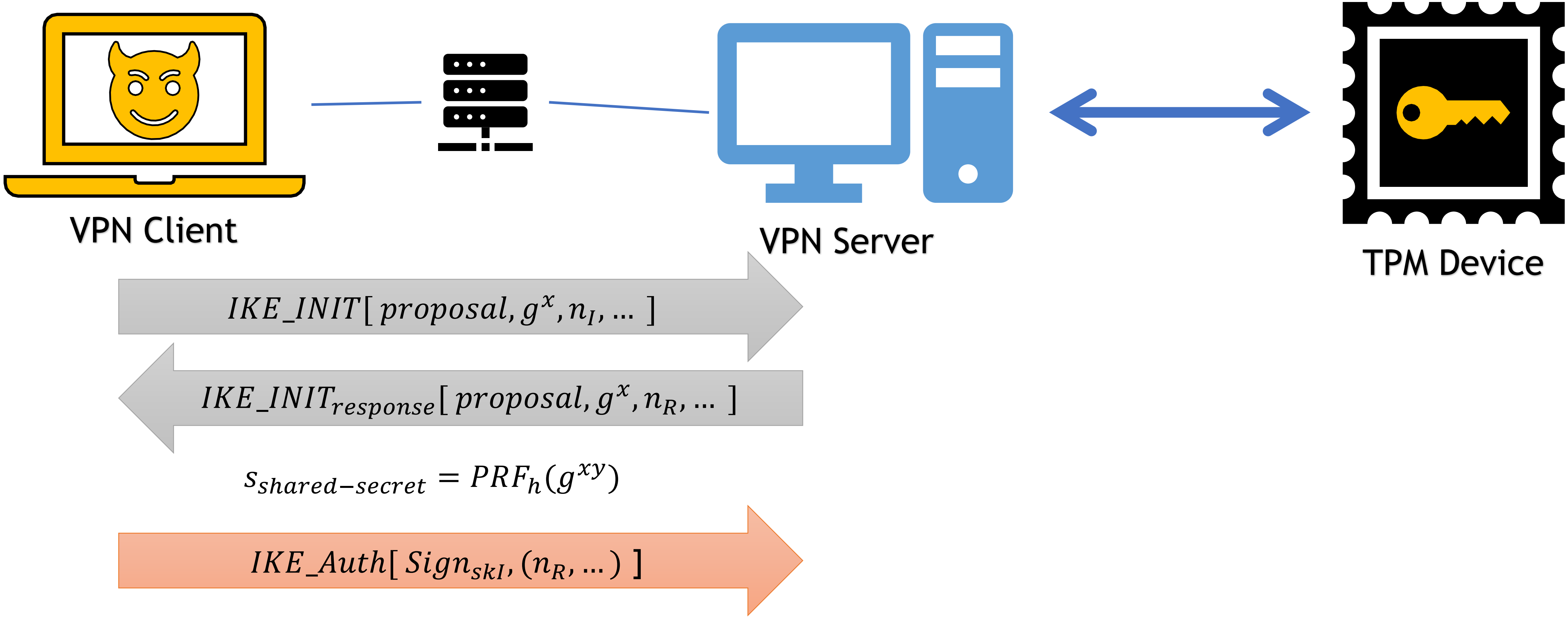


TPM-Fail Case Study: StrongSwan VPN


VPN Client
VPN Server
TPM Device
$IKE\_INIT[\,proposal, g^x, n_I, \dots\,]$
$IKE\_INIT_{response}[\,proposal, g^x, n_R, \dots\,]$

$$s_{shared-secret} = PRF_h(g^{xy})$$

TPM-Fail Case Study: StrongSwan VPN



VPN Client

VPN Server

TPM Device

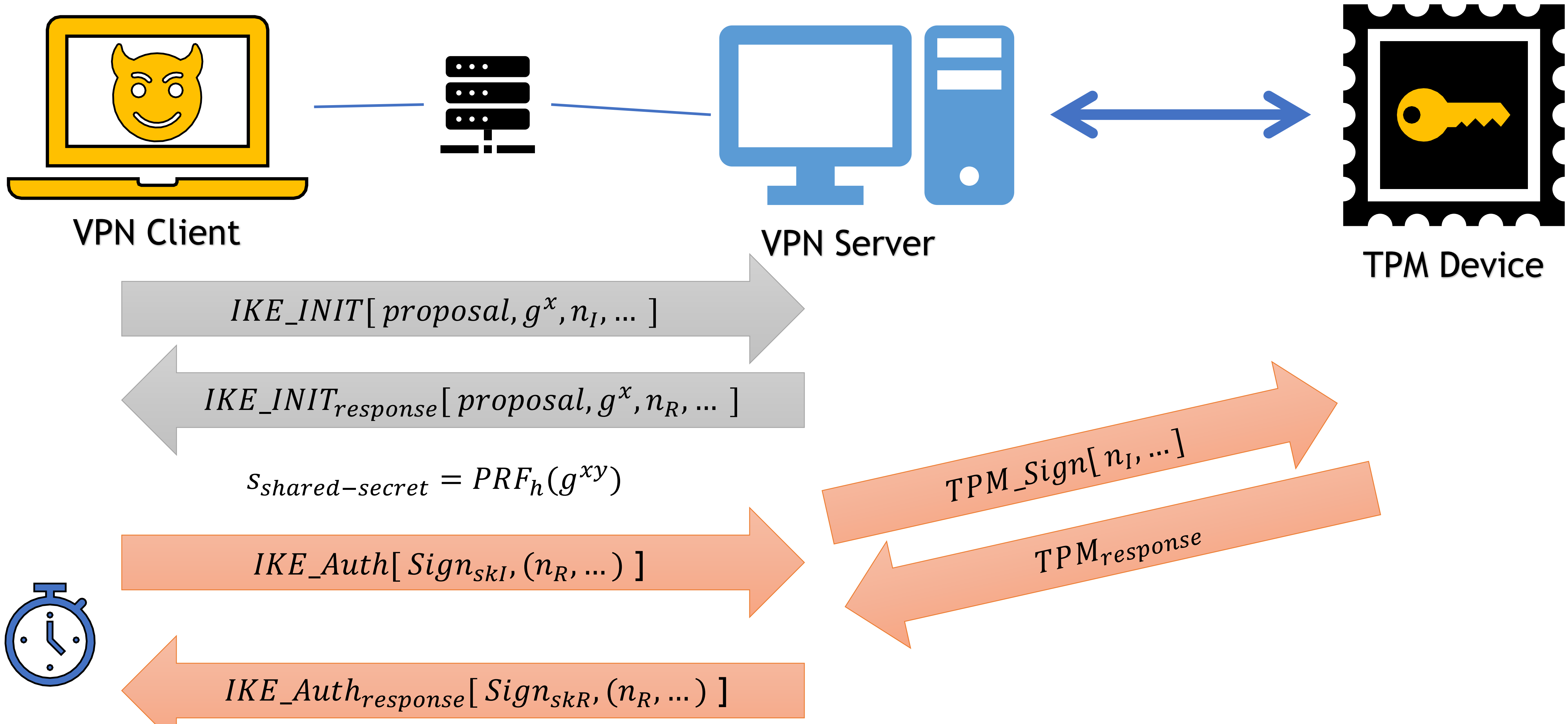
$IKE\_INIT[\,proposal, g^{x}, n_I, \ldots\,]$

$IKE\_INIT_{response}[\,proposal, g^{x}, n_R, \ldots\,]$

$s_{shared-secret} = PRF_h(g^{xy})$

$IKE\_Auth[\,Sign_{skI}, (n_R, \ldots)\,]$

TPM-Fail Case Study: StrongSwan VPN



VPN Client

VPN Server

TPM Device

$IKE\_INIT[\,proposal, g^x, n_I, \dots\,]$

$IKE\_INIT_{response}[\,proposal, g^x, n_R, \dots\,]$

$TPM\_Sign[\,n_I, \dots]$

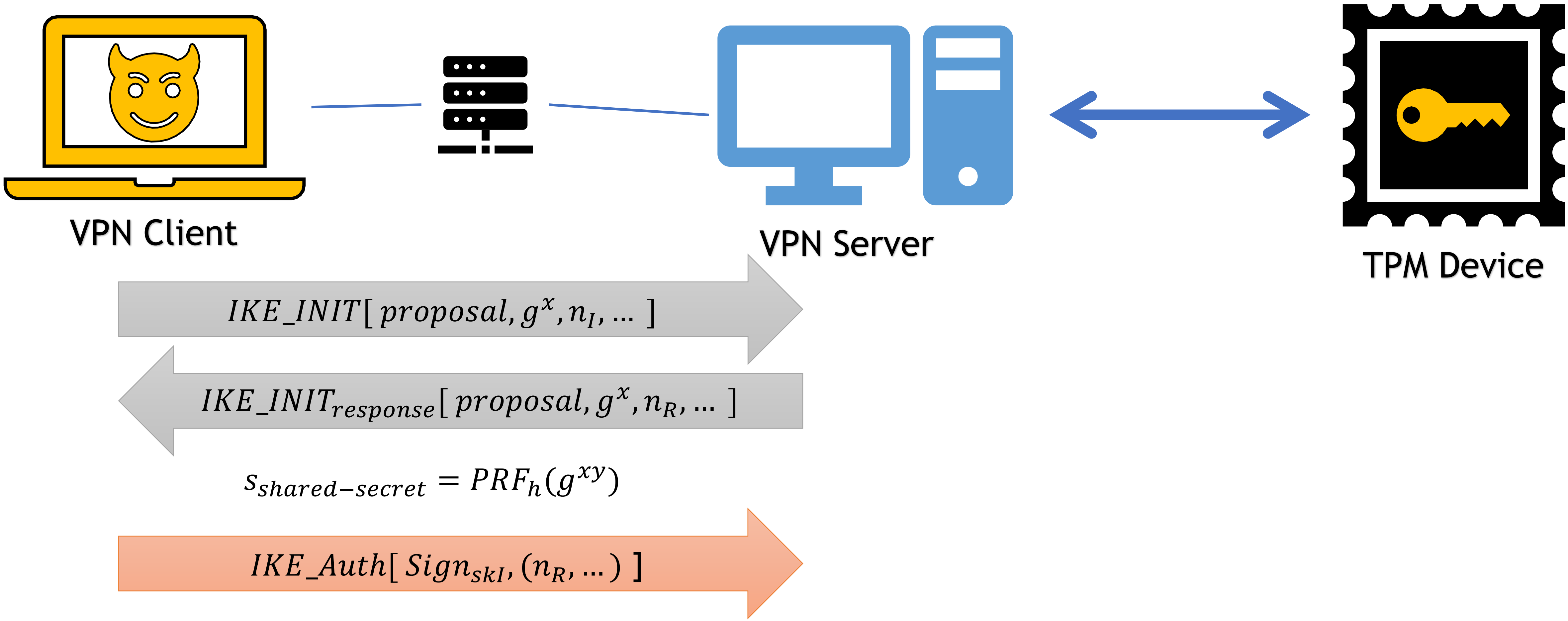
$s_{shared-secret} = PRF_h(g^{xy})$

$TPM_{response}$

$IKE\_Auth[\,Sign_{skI}, (n_R, \dots)\,]$

$IKE\_Auth_{response}[\,Sign_{skR}, (n_R, \dots)\,]$

TPM-Fail Case Study: StrongSwan VPN



VPN Client

VPN Server

TPM Device

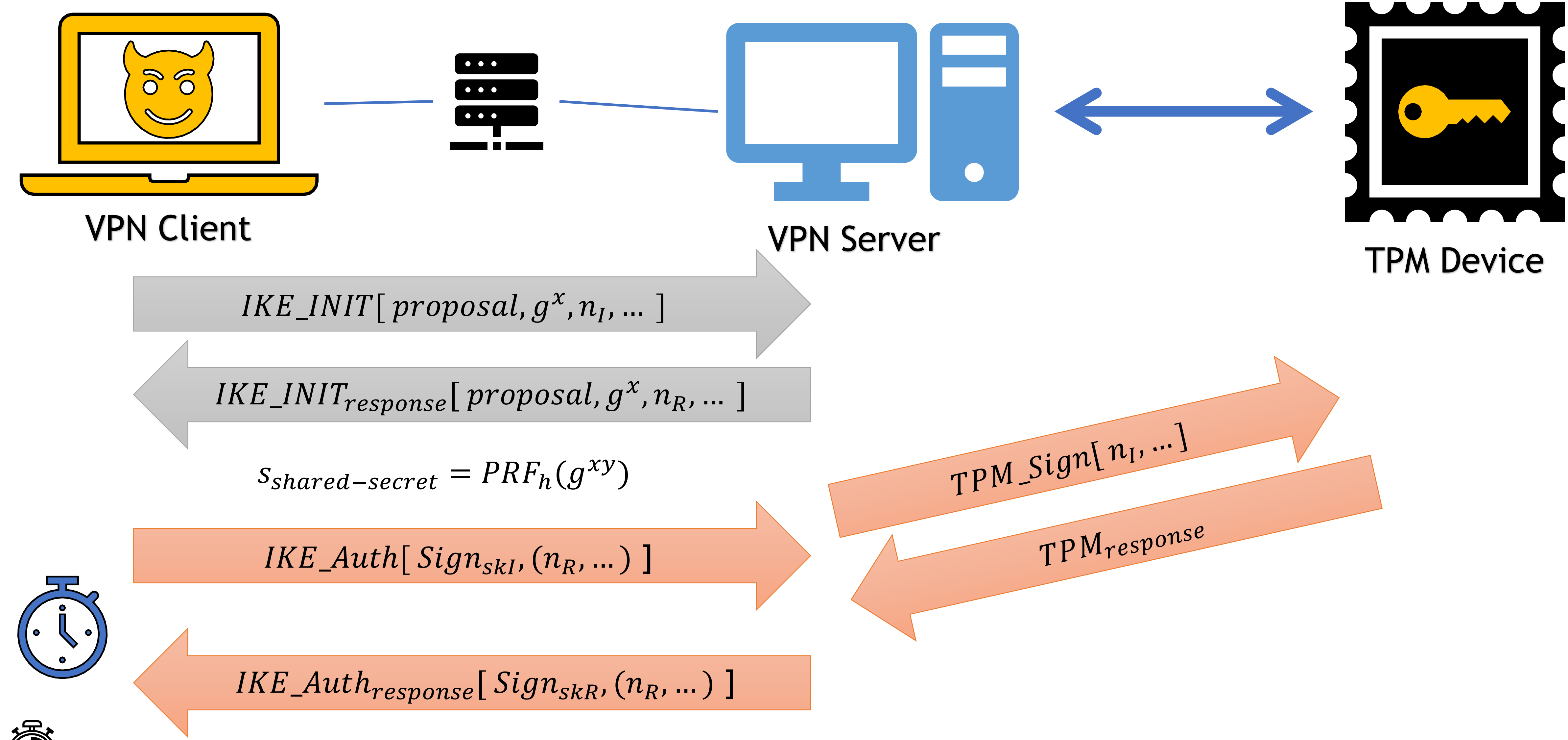
$IKE\_INIT[\,proposal, g^{x}, n_{I}, \dots\,]$

$IKE\_INIT_{response}[\,proposal, g^{x}, n_{R}, \dots\,]$

$$s_{shared-secret} = PRF_{h}(g^{xy})$$

$IKE\_Auth[\,Sign_{skI}, (n_{R}, \dots)\,]$

TPM-Fail Case Study: StrongSwan VPN



VPN Client

VPN Server

TPM Device

$IKE\_INIT[\,proposal, g^x, n_I, \dots\,]$

$IKE\_INIT_{response}[\,proposal, g^x, n_R, \dots\,]$

$s_{shared-secret} = PRF_h(g^{xy})$

$IKE\_Auth[\,Sign_{skI}, (n_R, \dots)\,]$

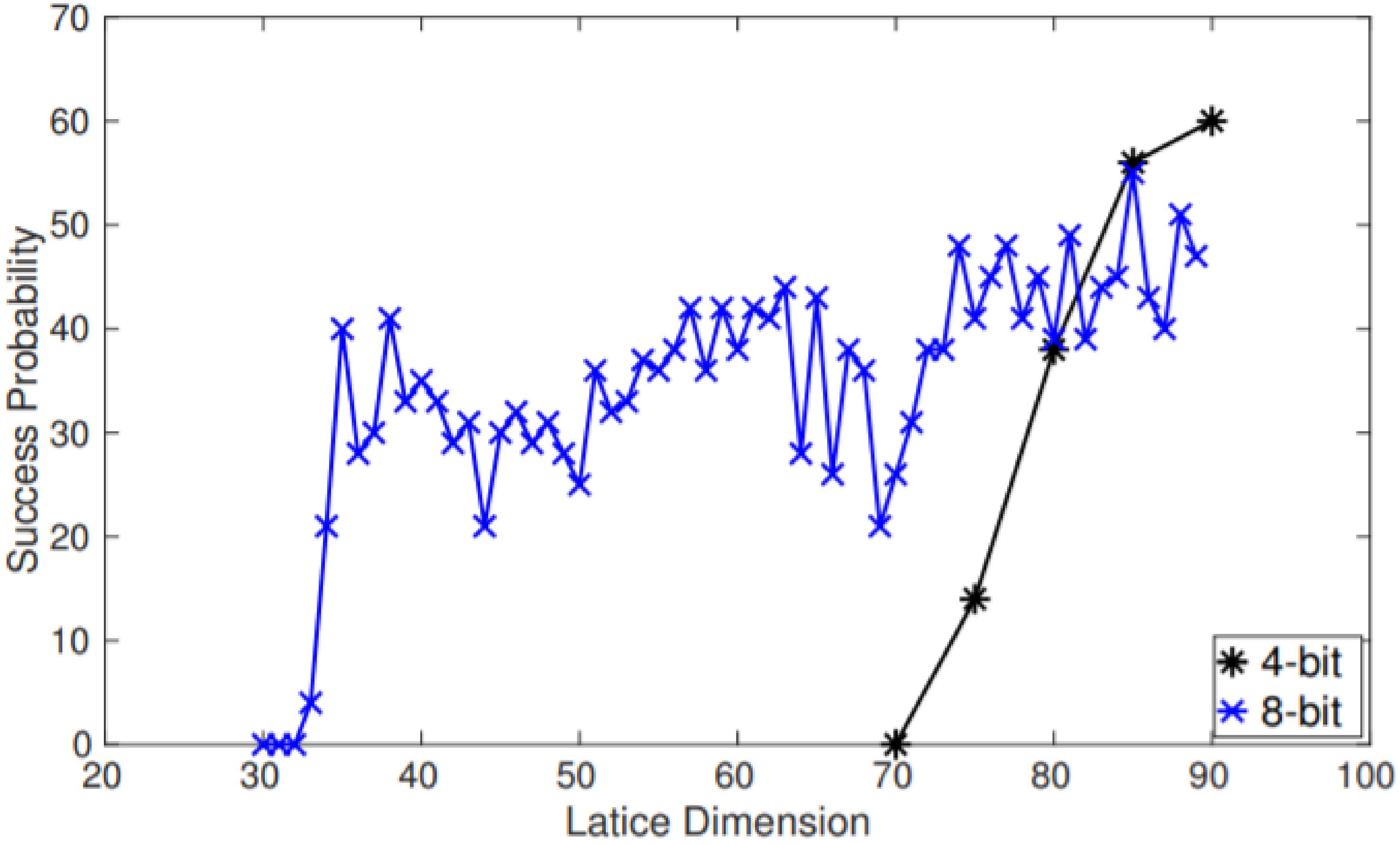
$TPM\_Sign[\,n_I, \dots\,]$

$TPM_{response}$

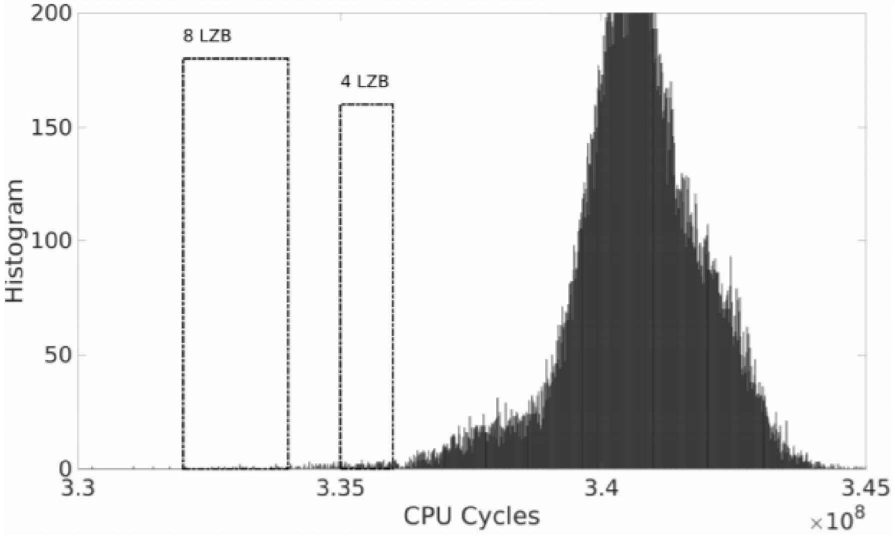
$IKE\_Auth_{response}[\,Sign_{skR}, (n_R, \dots)\,]$

TPM-Fail Case Study: StrongSwan VPN Key Recovery

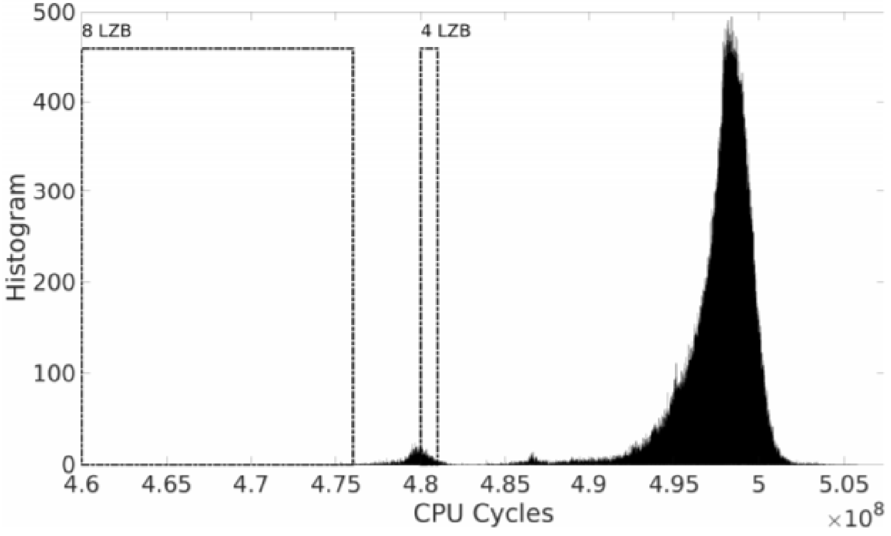
- Remote Key Recovery after about 44,000 handshake ~= 5 hours

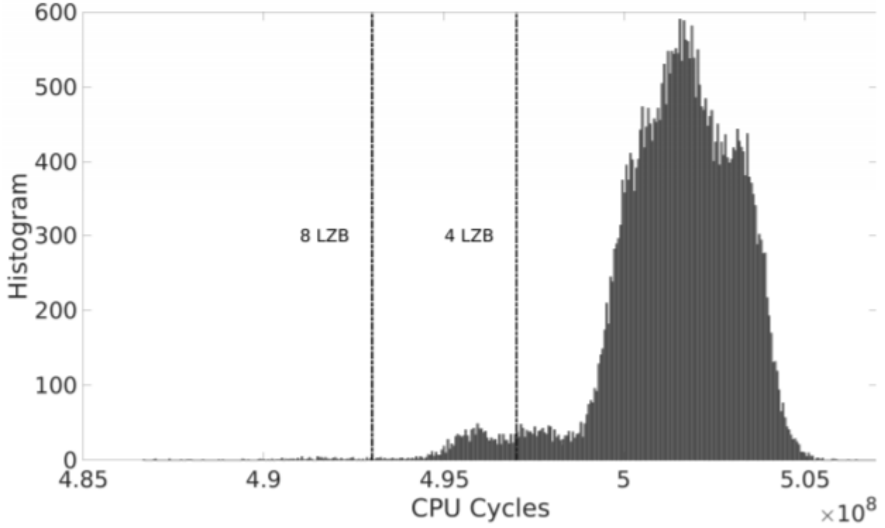


Remote StrongSwan VPN

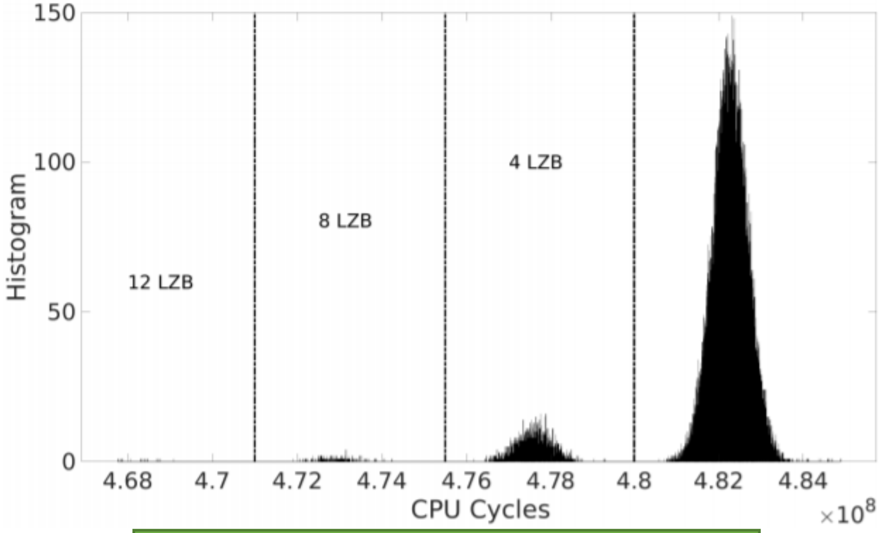


User Adversary



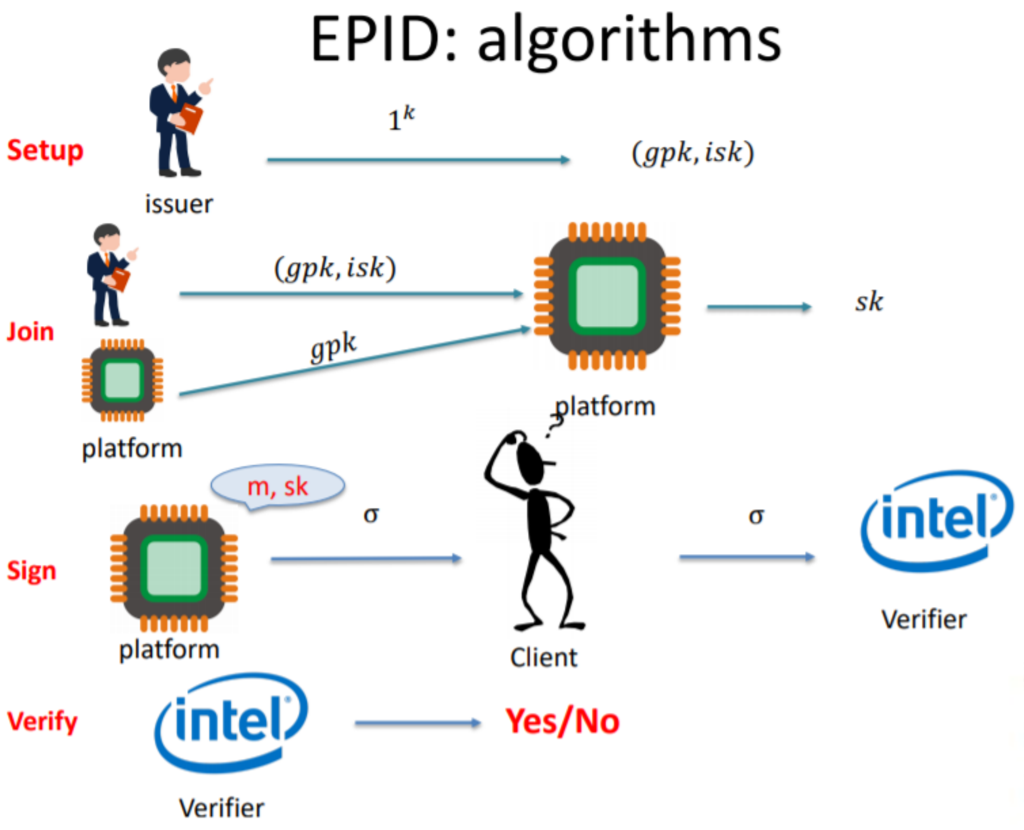


Remote Sample UDP App

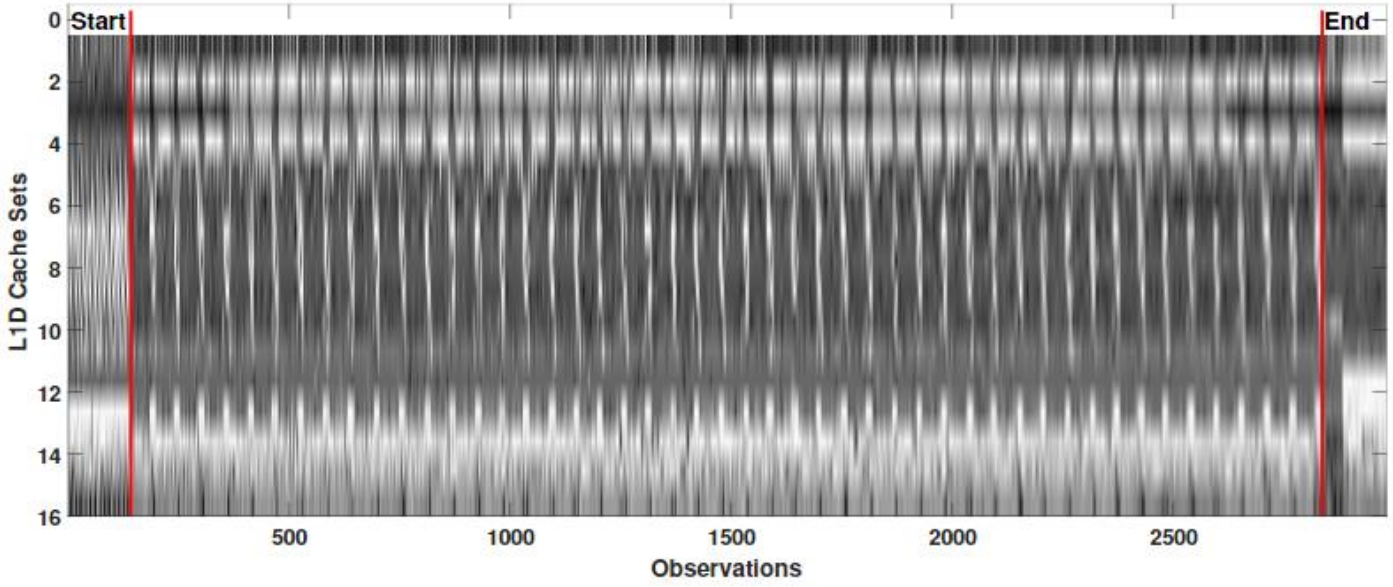


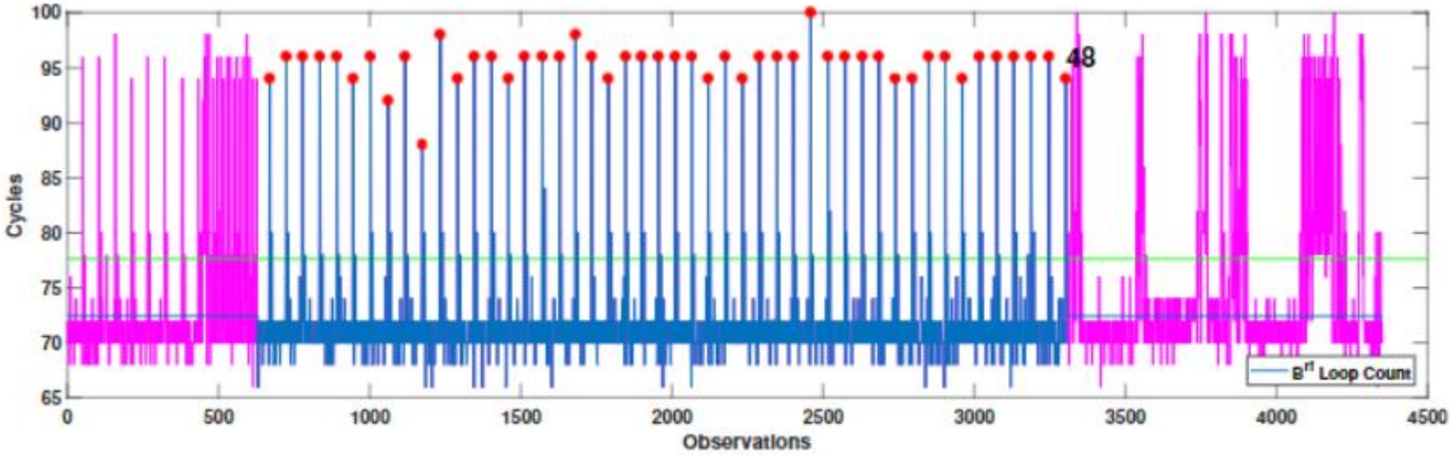
System Adversary

CacheQuote [2]



EPID: algorithms



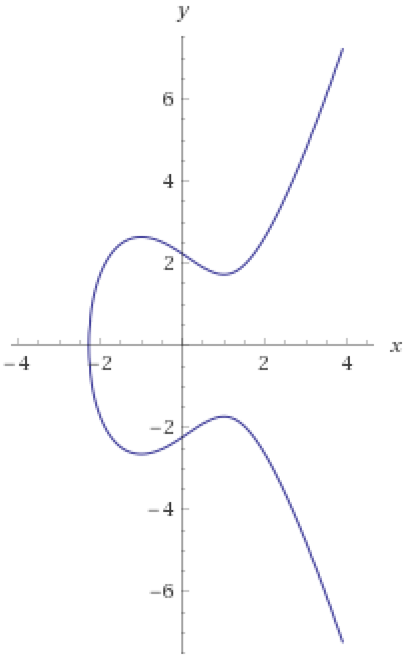


[2] F Dall, G De Micheli, T Eisenbarth, D Genkin, N Heninger, A Moghimi, Y Yarom. CacheQuote: Efficiently Recovering Long-term Secrets of SGX EPID via Cache Attacks

Cryptographic Implementation is Hard - ECDSA

$ECDSA\ Sign$:

$(x_1, y_1) = k_i \times G$
$r_i = x_1 \bmod n$
$s_i = {k_i}^{-1}(z + r_i d) \bmod n$

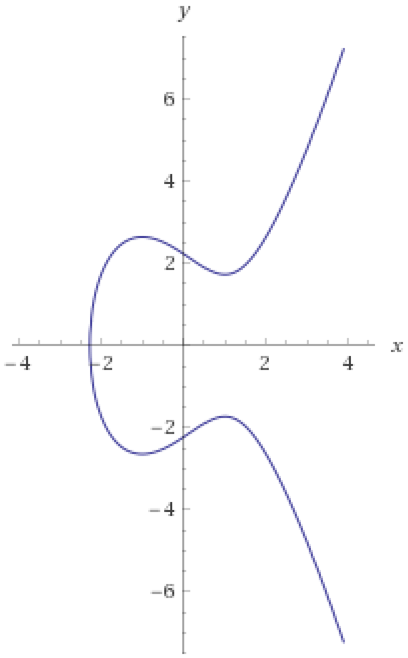


$y^2 = x^3 - 3x + 5$

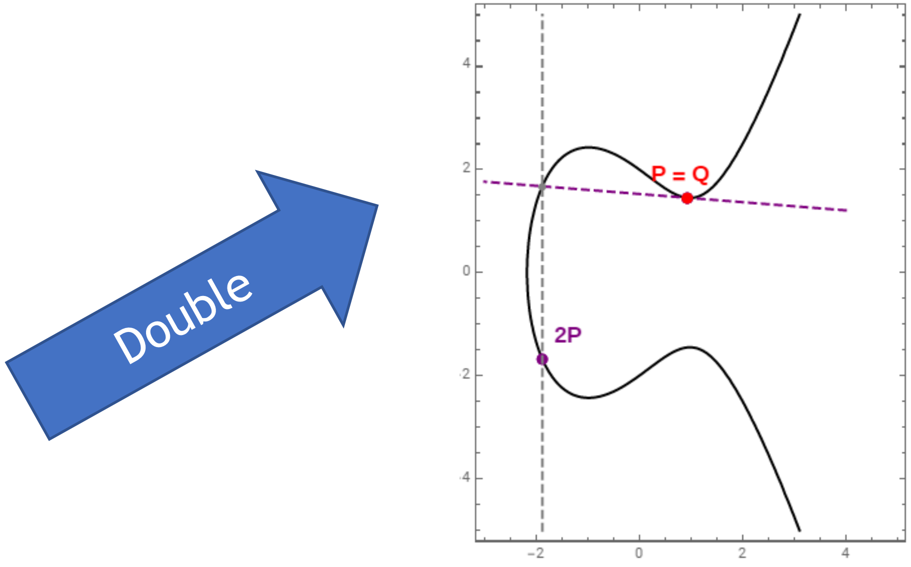
Cryptographic Implementation is Hard - ECDSA

$ECDSA\ Sign$:

$$(x_1, y_1) = k_i \times G$$
$$r_i = x_1 \bmod n$$
$$s_i = {k_i}^{-1}(z + r_i d) \bmod n$$



$$y^2 = x^3 - 3x + 5$$

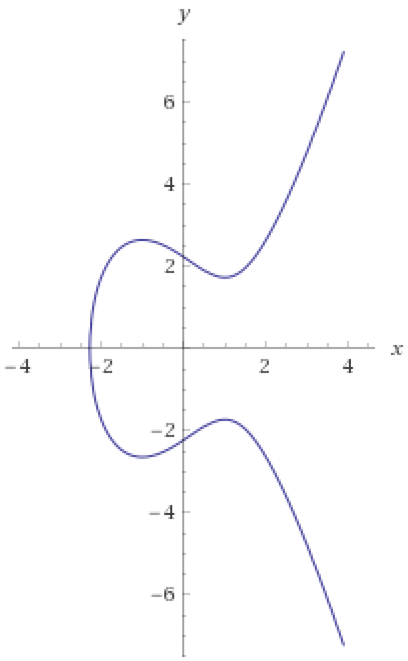


Cryptographic Implementation is Hard - ECDSA

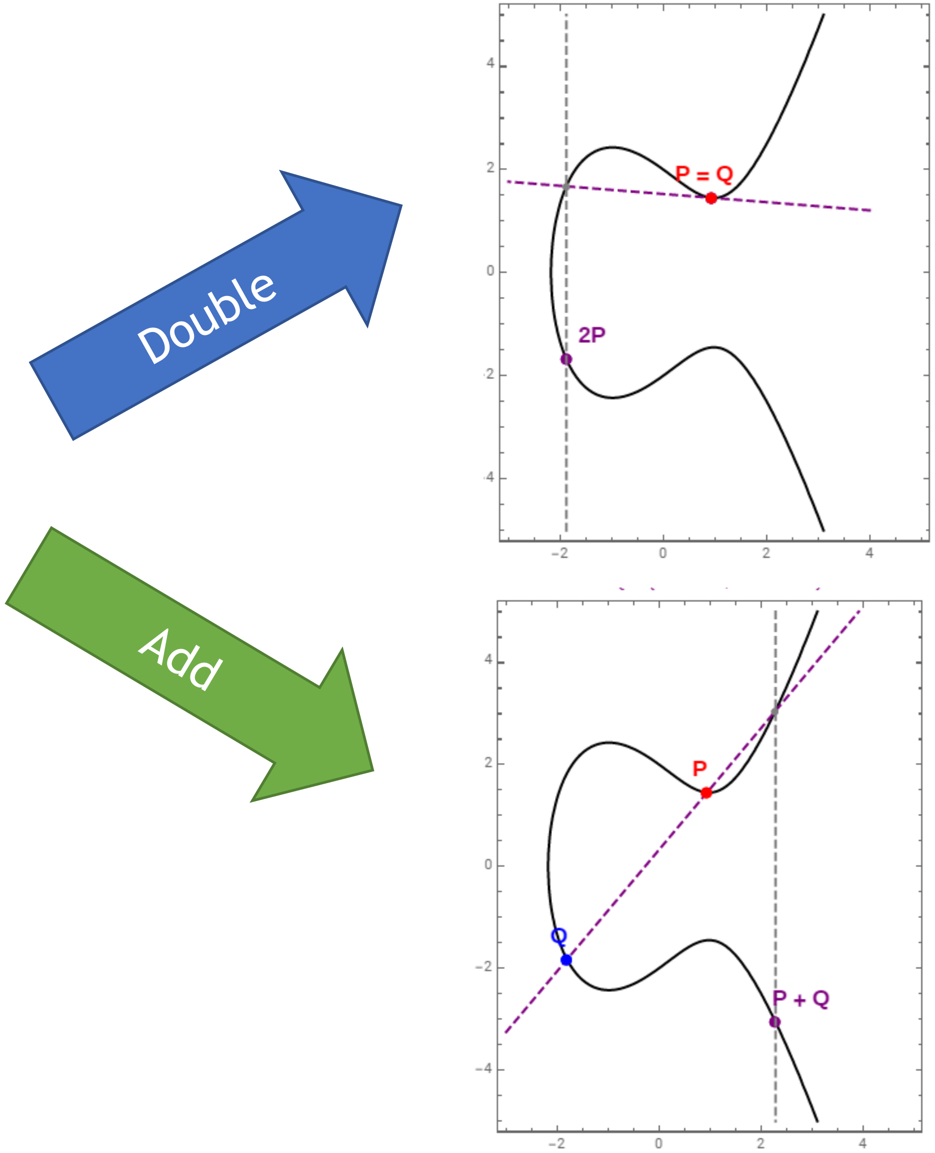
$ECDSA\ Sign$:

$$(x_1, y_1) = k_i \times G$$
$$r_i = x_1 \ mod\ n$$
$$s_i = {k_i}^{-1}(z + r_i d) \ mod\ n$$

$$k_i = 3 \rightarrow \ 3 \times G = \ 2G + G$$



$$y^2 = x^3 - 3x + 5$$

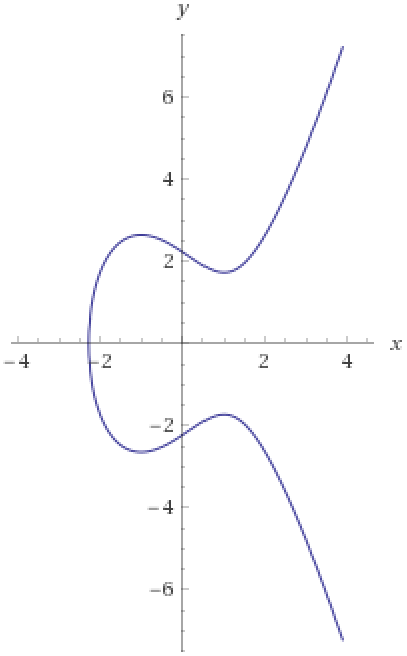


Cryptographic Implementation is Hard - ECDSA

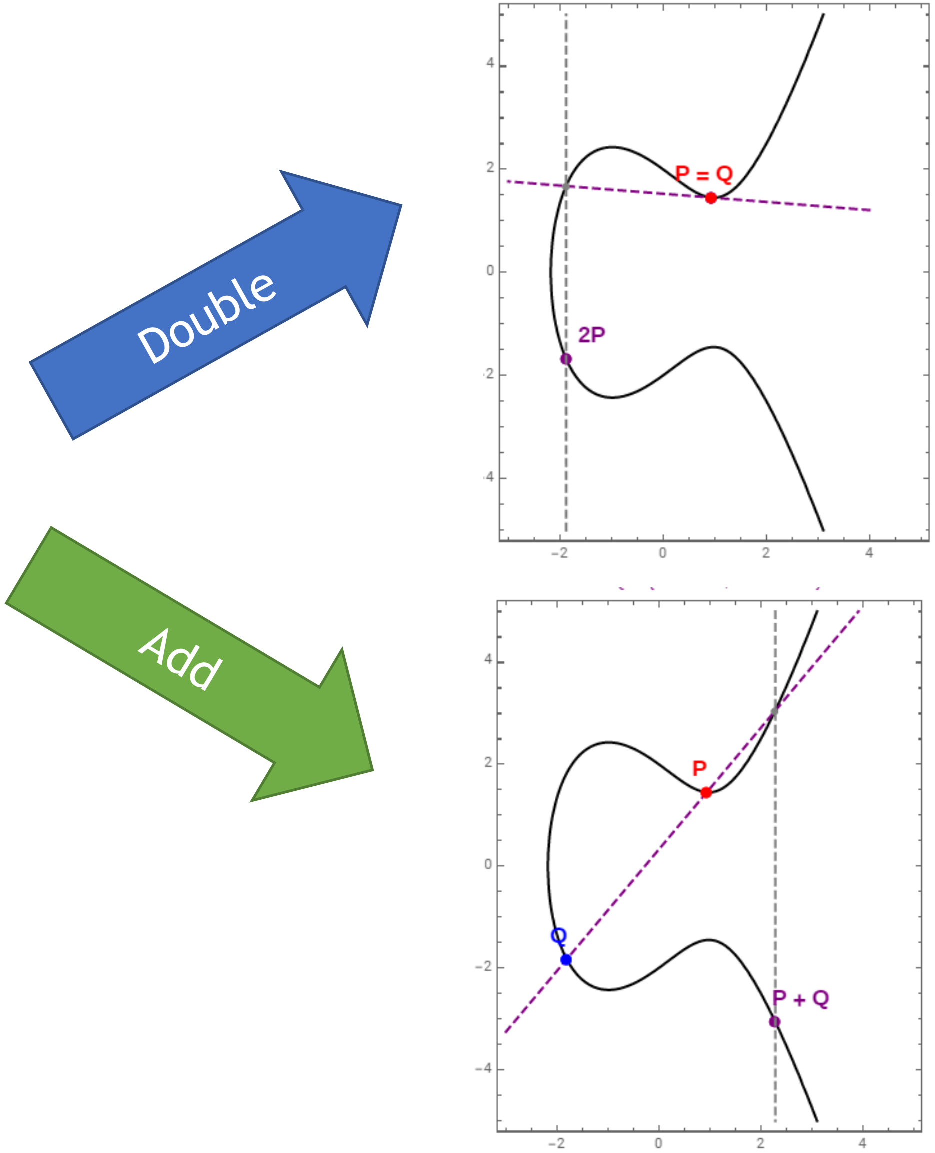
$ECDSA\ Sign$:

$$(x_1, y_1) = k_i \times G$$
$$r_i = x_1 \bmod n$$
$$s_i = k_i^{-1}(z + r_i d) \bmod n$$

$$k_i = 3 \rightarrow 3 \times G = 2G + G$$
$$k_i = 7 \rightarrow 7 \times G = 2(2G) + 2G + G$$



$$y^2 = x^3 - 3x + 5$$

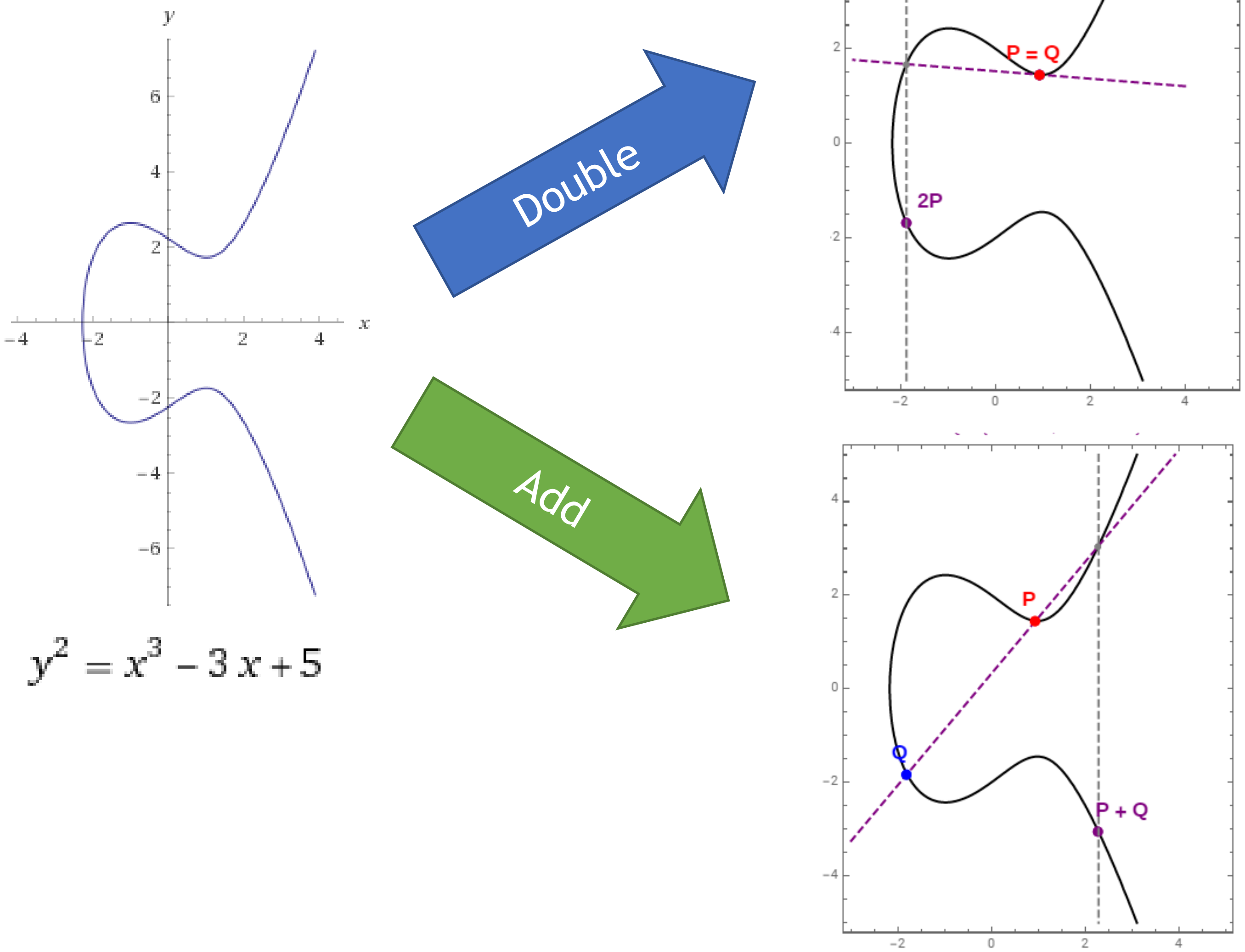


Cryptographic Implementation is Hard - ECDSA

$ECDSA\ Sign$:

$$(x_1, y_1) = k_i \times G$$
$$r_i = x_1\ mod\ n$$
$$s_i = {k_i}^{-1}(z + r_i d)\ mod\ n$$

$$k_i = 3 \rightarrow\ 3 \times G =\ 2G + G$$
$$k_i = 7 \rightarrow 7 \times G = 2(2G) + 2G + G$$
$$k_i = 7 \rightarrow 23 \times G$$
$$= 2(2(2(2G) + G) + G) + G$$



$$y^2 = x^3 - 3x + 5$$

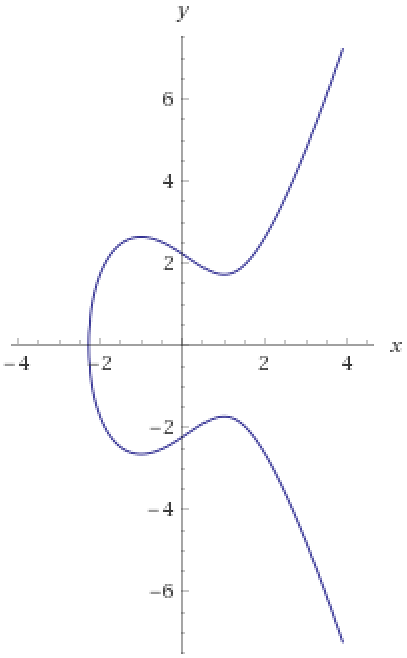
Cryptographic Implementation is Hard - ECDSA

$ECDSA\ Sign$:

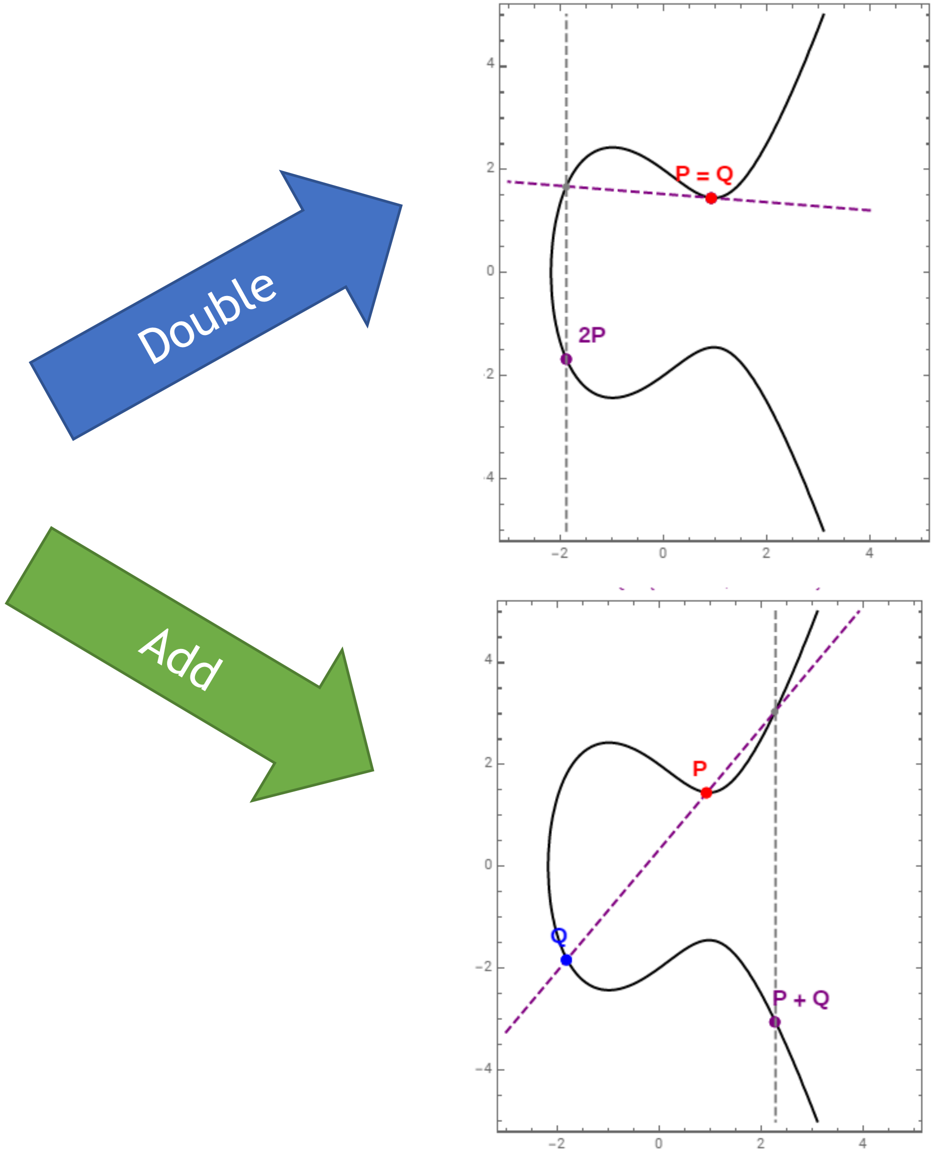
$$(x_1, y_1) = k_i \times G$$
$$r_i = x_1\ mod\ n$$
$$s_i = k_i^{-1}(z + r_i d)\ mod\ n$$

$$k_i = 3 \rightarrow\ 3 \times G =\ 2G + G$$
$$k_i = 7 \rightarrow 7 \times G = 2(2G) + 2G + G$$
$$k_i = 7 \rightarrow 23 \times G$$
$$= 2(2(2(2G) + G) + G) + G$$

```
//Scalar Mul: Add & Double
Q = 0
R = G
for k_b in k:
    if k_b == 1:
        Q = add(Q, R)
    R = double(R)
return Q
```



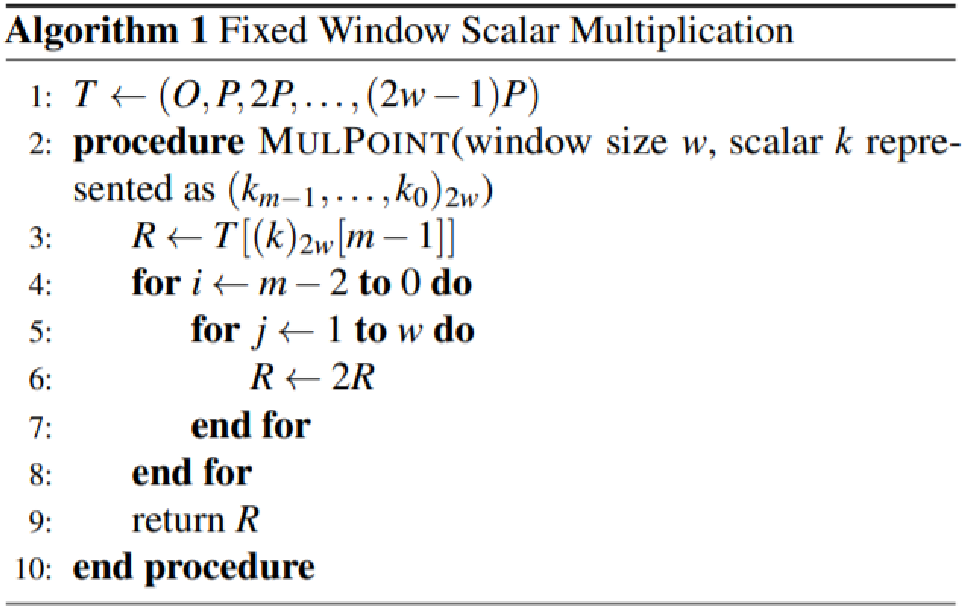
$$y^2 = x^3 - 3x + 5$$



Cryptographic Implementation is Hard

- Many Algorithms to do the same thing
  - Scalar Multiplication
    - Double-Add Algorithm
    - Montgomery Double-Add
    - Sliding Window
    - Fixed Window

- Unclear Threat Model
  - What is a side channel?
  - Power Analysis, Timing, Cache?



Algorithm 1 Fixed Window Scalar Multiplication

```
$T \leftarrow (O, P, 2P, \ldots, (2w-1)P)$
procedure MULPOINT(window size $w$, scalar $k$ represented as $(k_{m-1}, \ldots, k_0)_{2w}$)
    $R \leftarrow T[(k)_{2w}[m-1]]$
    for $i \leftarrow m-2$ to $0$ do
        for $j \leftarrow 1$ to $w$ do
            $R \leftarrow 2R$
        end for
    end for
    return $R$
end procedure
```

Software Leakages

- Secret Dependent Control Flow

```
for(int i = 0; i < Bitlength(key); ++i)
```

- Secret Dependent Memory Access Pattern

```
state[i] = state[i] ^ sbox[roundKey[i]]
```

- Secret Dependent Timing, *e.g: ARM Cortext-M3* umull

MicroWalk Goal

- Automated Analysis
- Dynamic Approach
- Binary-level Analysis:
  - Leakages introduced by compilation
  - Closed-source libraries
- Locate leakage source at Instruction Level

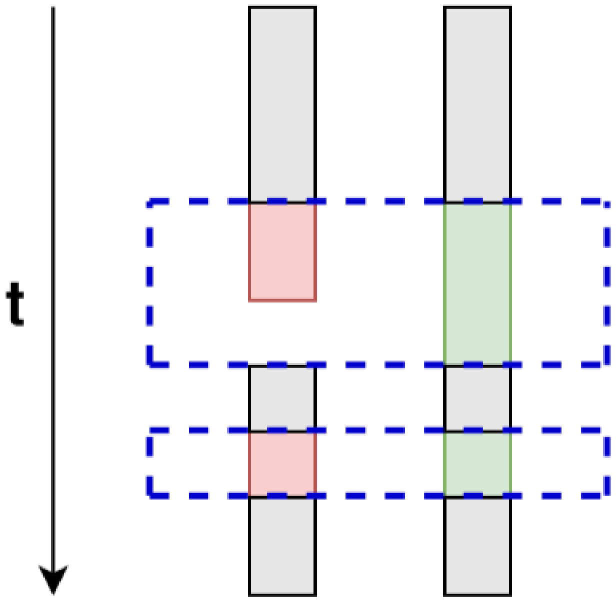
MicroWalk Model

- In practice: Attacker measures
  - Execution time for `(int i = 0; i < bitlength(key); ++i)`
  - Memory usage pattern `state[i] = state[i] ^ sbox[roundKey[i]]`

- In theory: Attacker gets access to execution trace with
  - Executed instructions
  - Branch targets
  - Memory access offsets

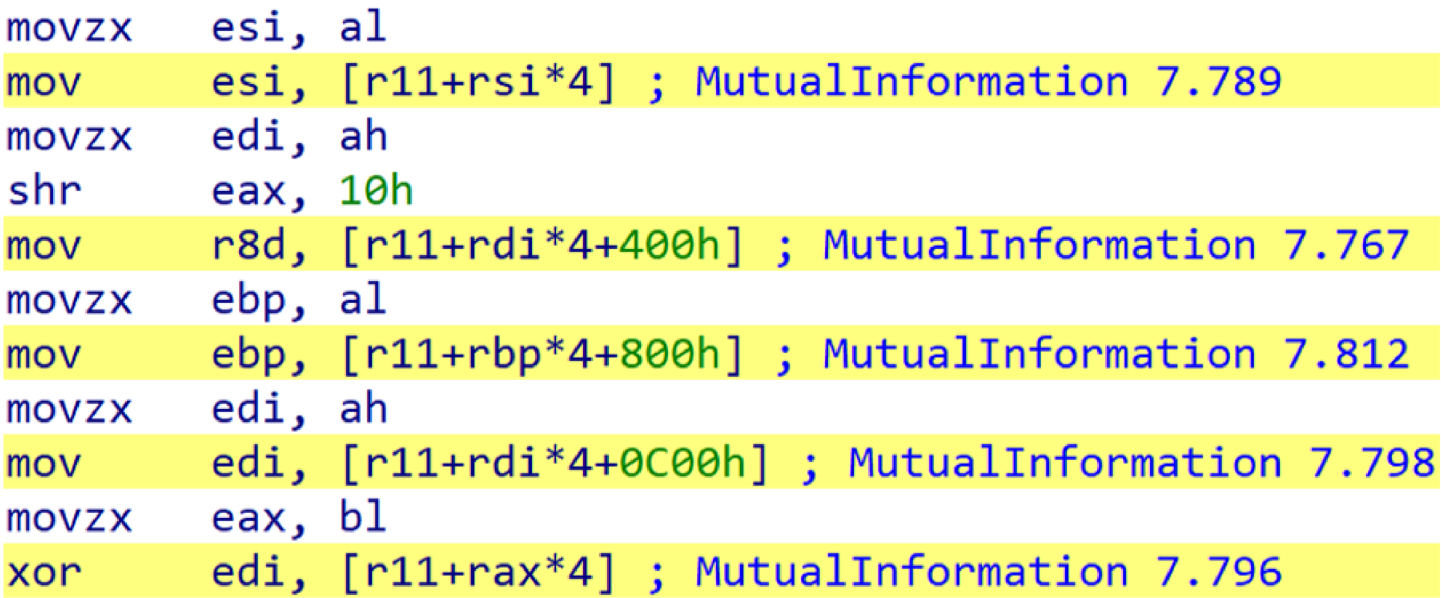
MicroWalk Approach

- Generate set of random test cases and capture execution traces
- Analysis A: Compute pairwise diffs



MicroWalk Approach

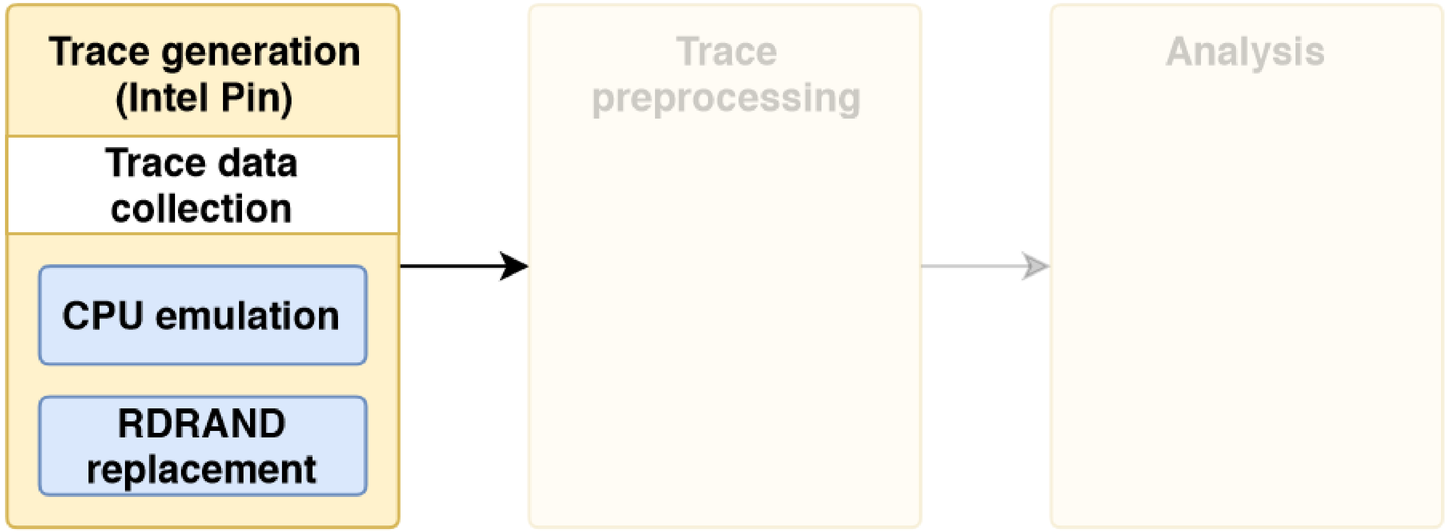
- Generate set of random test cases and capture execution traces
- Analysis A: Compute pairwise diffs
- Analysis B: Compute mutual information between execution trace and input



```
movzx   esi, al
mov     esi, [r11+rsi*4] ; MutualInformation 7.789
movzx   edi, ah
shr     eax, 10h
mov     r8d, [r11+rdi*4+400h] ; MutualInformation 7.767
movzx   ebp, al
mov     ebp, [r11+rbp*4+800h] ; MutualInformation 7.812
movzx   edi, ah
mov     edi, [r11+rdi*4+0C00h] ; MutualInformation 7.798
movzx   eax, bl
xor     edi, [r11+rax*4] ; MutualInformation 7.796
```

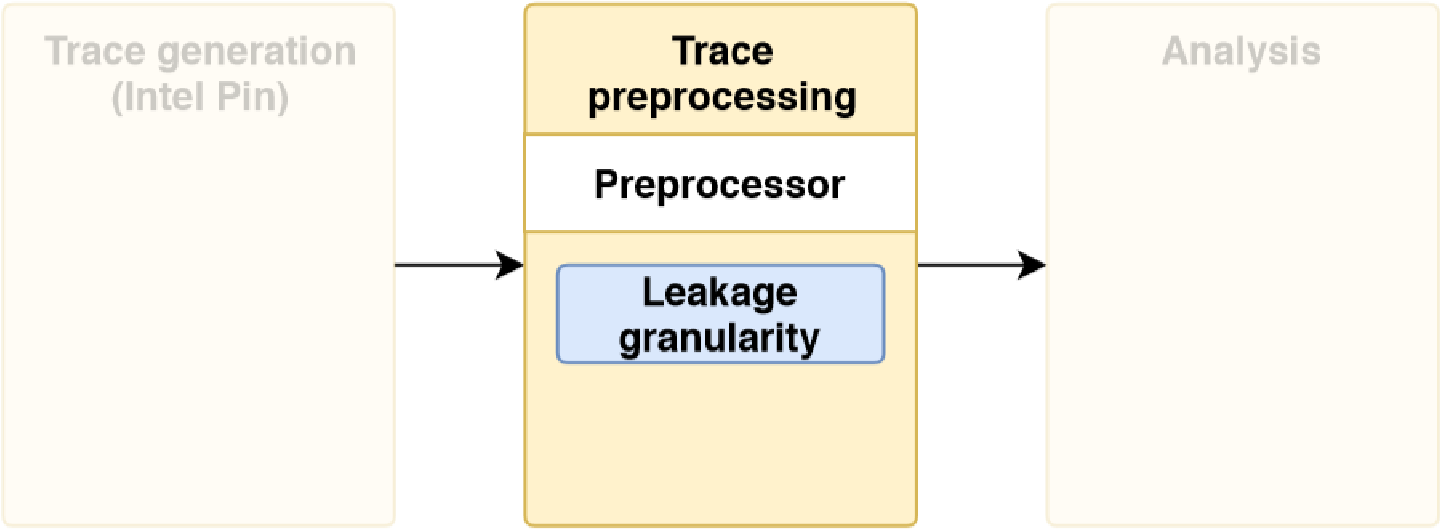
MicroWalk Implementation

- Dynamic binary instrumentation using Intel Pin
- Collect traces while program runs
- Modules:
  - Emulate other CPUs or disable certain capabilities (e.g. AES-NI)
  - Modify RDRAND output



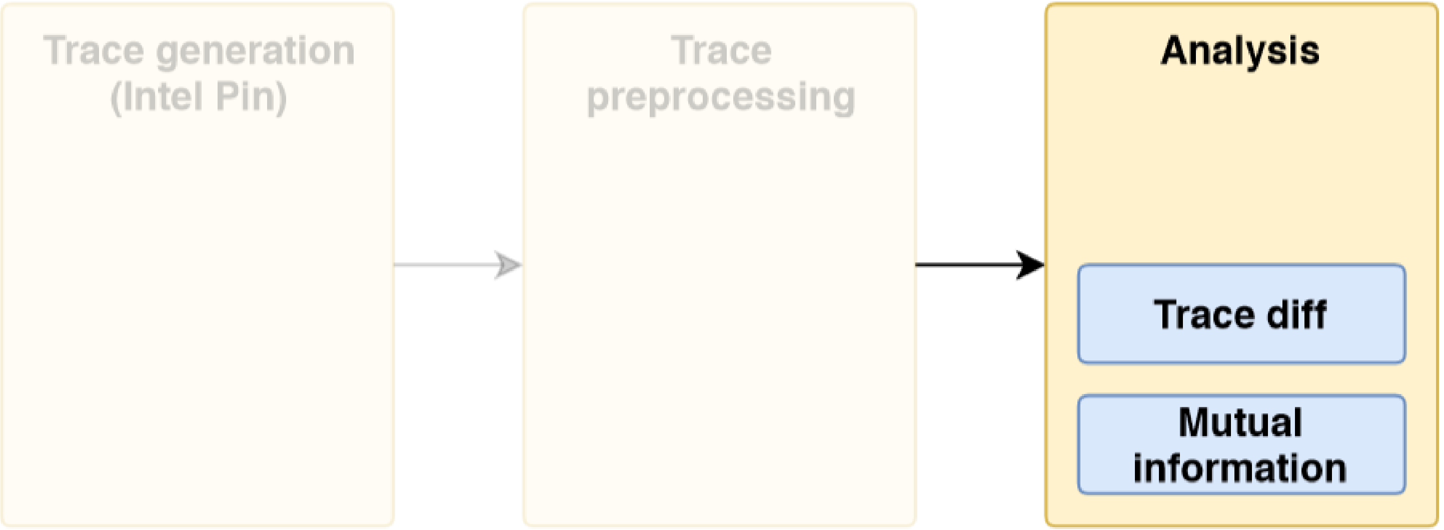
MicroWalk Implementation

- Raw traces only contain absolute addresses of memory accesses
  0x1111107A → sbox+0x7A
- Removal of uninteresting trace entries → considerable size reduction
- Modules:
  - Configure memory address leakage granularity 0x156F → 0x1540



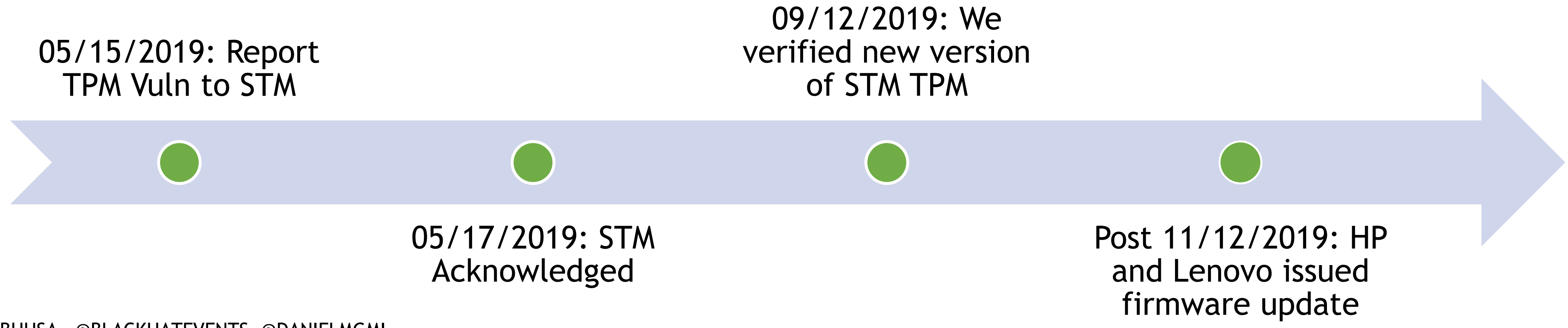
MicroWalk Implementation

- Load and analyze preprocessed traces
- Optionally pass results to visualization stage
- Modules:
  - Compute pairwise trace diffs
  - Calculate mutual information for each memory accessing instruction



Coordinated Disclosure - STMicroelectronics

- STMicroelectronics (CVE-2019-16863)
  - 05/15/2019: Reported to ST
  - 05/17/2019: Acknowledged
  - Lots of calls/emails to clarify the disclosure process
  - 09/12/2019: Verified new version of STM TPM firmware
  - After 11/12/2019:
    - HP and Lenovo have issued firmware updates.
    - ST released a list of affected devices.



05/15/2019: Report TPM Vuln to STM

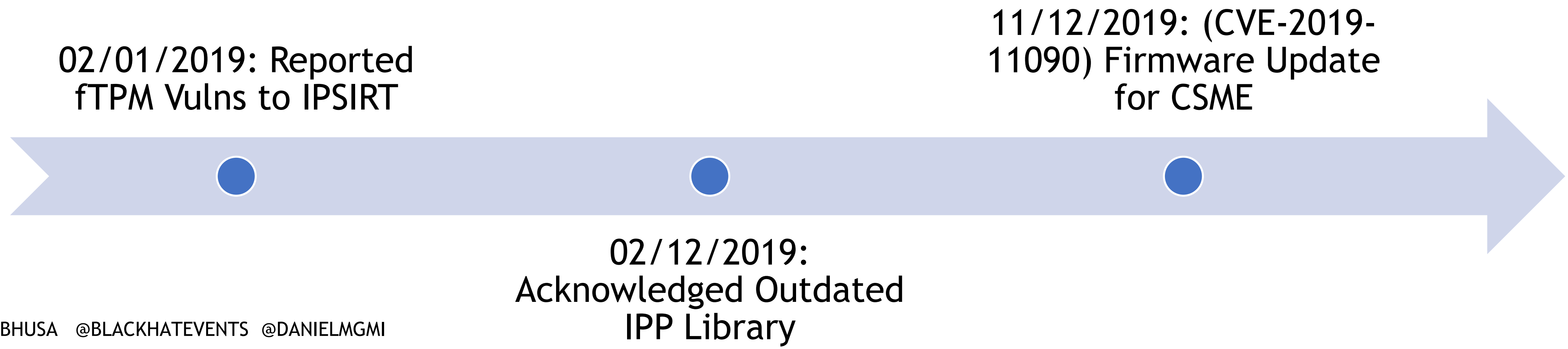
05/17/2019: STM Acknowledged

09/12/2019: We verified new version of STM TPM

Post 11/12/2019: HP and Lenovo issued firmware update

Coordinated Disclosure - Intel

- Intel (CVE-2019-11090)
  - 02/01/2019: Reported to IPSIRT
  - 02/12/2019: Acknowledged (Outdated Intel IPP Crypto library)
  - 11/12/2019: Firmware Update for Intel Management Engine

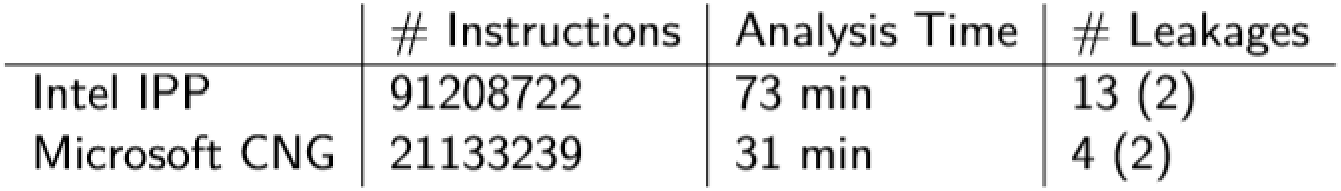

02/01/2019: Reported fTPM Vulns to IPSIRT

11/12/2019: (CVE-2019-11090) Firmware Update for CSME

02/12/2019: Acknowledged Outdated IPP Library

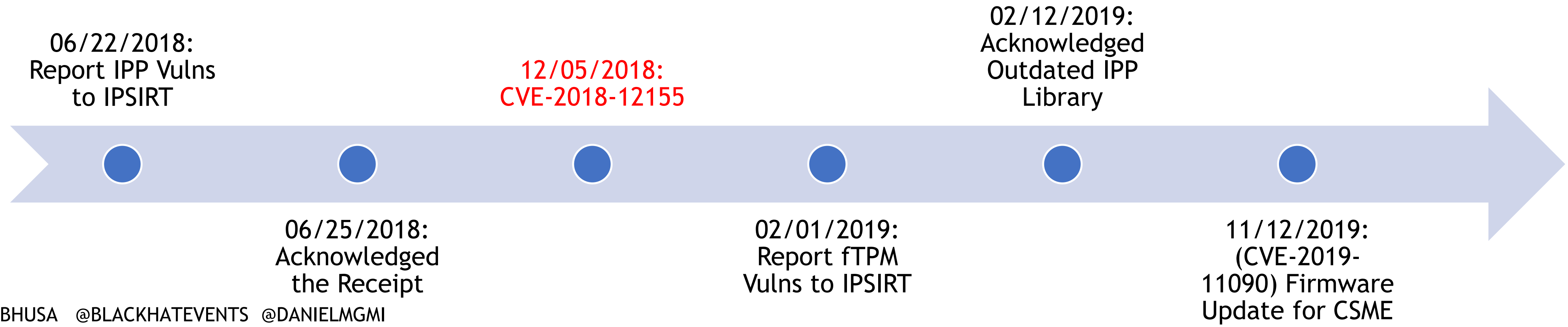
MicroWalk Analysis Results

- Rigorous Analysis of two Closed-source Libraries



	# Instructions	Analysis Time	# Leakages
Intel IPP	91208722	73 min	13 (2)
Microsoft CNG	21133239	31 min	4 (2)

- Intel IPP CVEs
    - CVE-2018-12155
    - CVE-2018-12156



06/22/2018: Report IPP Vulns to IPSIRT

06/25/2018: Acknowledged the Receipt

12/05/2018: CVE-2018-12155

02/01/2019: Report fTPM Vulns to IPSIRT

02/12/2019: Acknowledged Outdated IPP Library

11/12/2019: (CVE-2019-11090) Firmware Update for CSME

Questions?!

Daniel Moghimi

@danielmgmi

TPM-FAIL

https://tpm.fail/

https://github.com/VernamLab/TPM-Fail

29TH USENIX SECURITY SYMPOSIUM

https://www.usenix.org/conference/usenixsecurity20/presentation/moghimi

https://github.com/UzL-ITS/Microwalk