

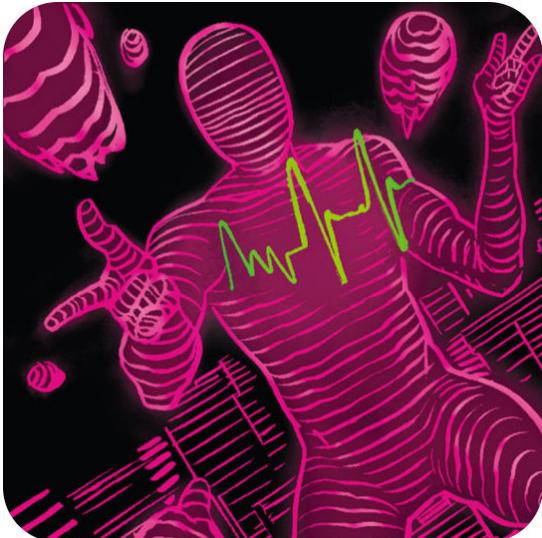
Practical Attacks Against White-Box Crypto Implementations

Cryptography Design for the IoT

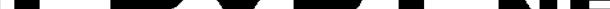
Philippe Teuwen

Quarkslab

15/07/2016



ECRYPT NET



whoami

Philippe Teuwen aka @doegox aka yobibe

- @ **Quarkslab**
- ♥ free software, security,
CTFs, photography
- 웹 <http://wiki.yobi.be>



Notice:

Research presented here was conducted
when I was working for NXP Semiconductors

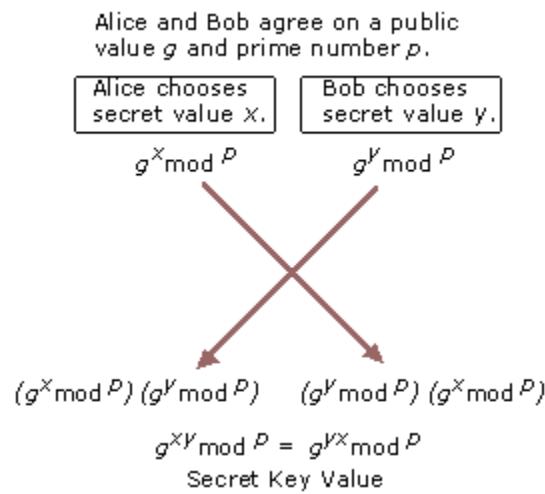


Outline

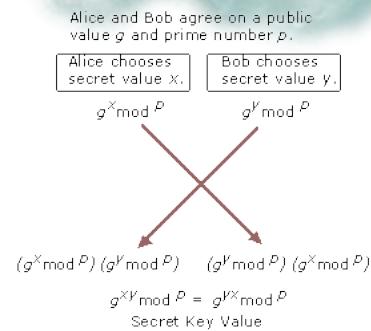
- Introduction to white-box cryptography
- Software execution traces
- Differential Computation Analysis
- Differential Fault Analysis
- Hands-on!

INTRODUCTION TO WHITE-BOX CRYPTOGRAPHY

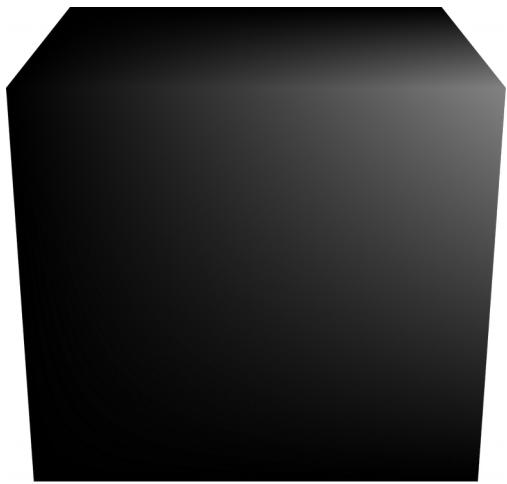
Black box model



Black box model



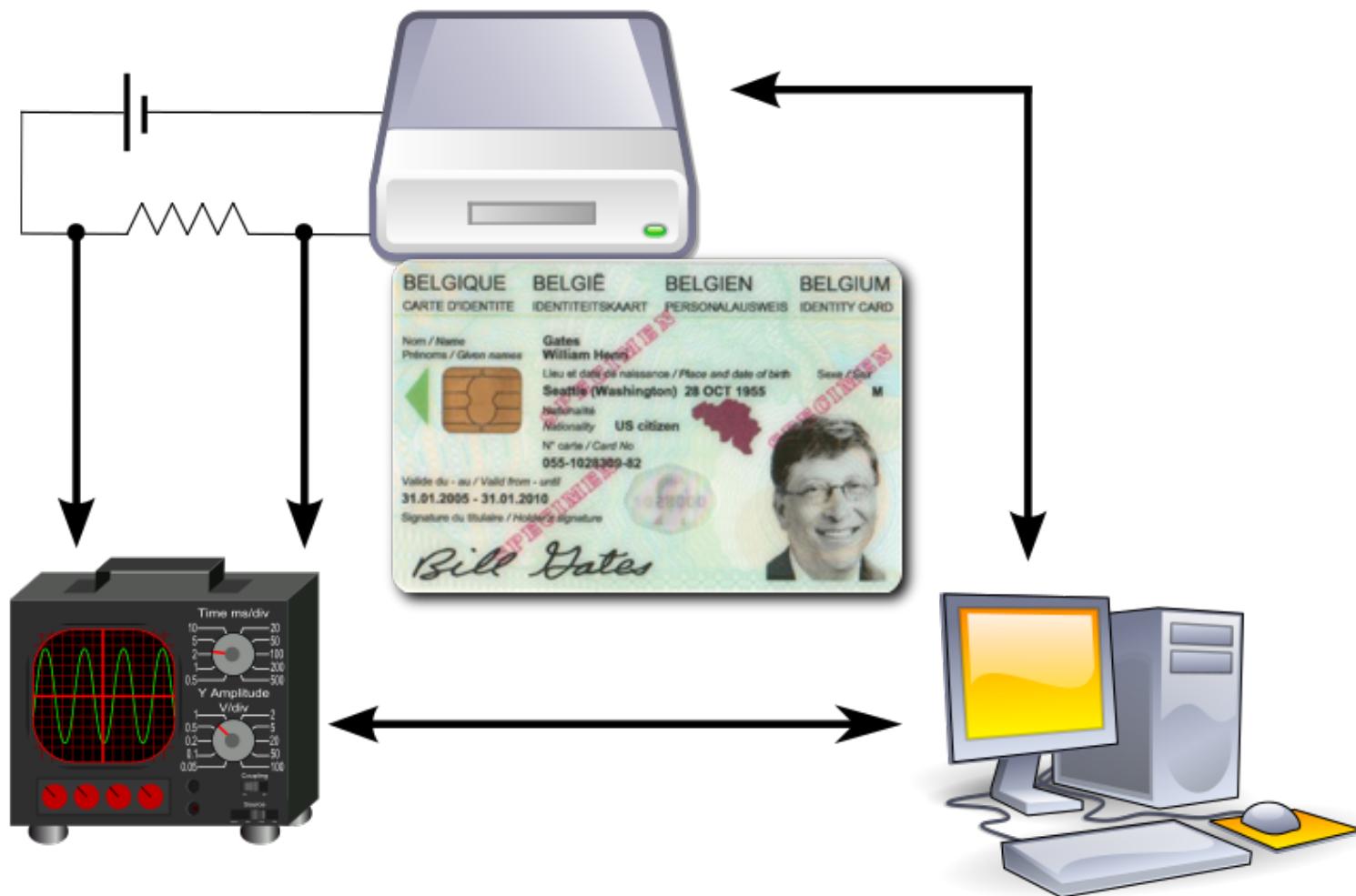
Black box model



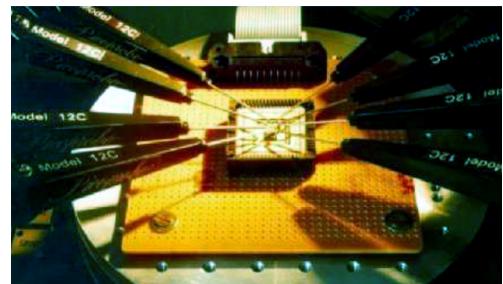
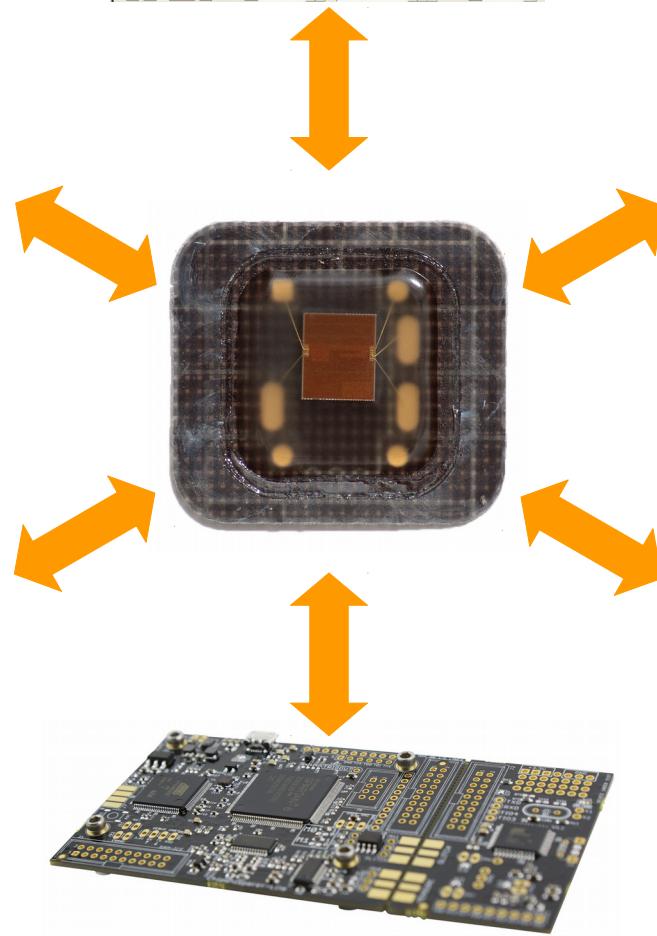
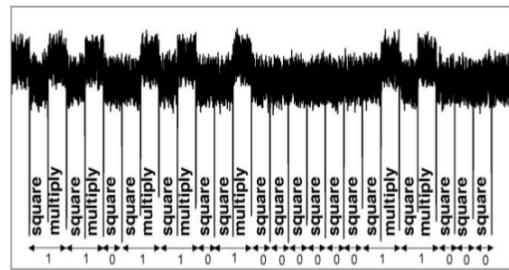
Grey box model



Grey box model



Grey box model



White box model



White box model



Address	Hex dump	ASCII		0012F8C0	0411154	STS empty 17.00000000000000000000000000000000
00457000	00 00 00 00 F1 3D 40 00:.		0012F900	0411154	7FFFD000
00457008	00 00 00 00 00 00 00 00			0012F940	0411154	00000000
00457010	C1 21 40 00 00 00 29 001.0.		0012FB40	0411154	77B1B429
00457018	00 00 00 00 00 00 00 00			0012F980	0411154	RETURN to ntddi.77B1B429
00457020	00 00 00 00 00 00 00 00			0012F9A0	0411154	650F00E8
00457028	00 00 00 00 00 00 00 00			0012F9C0	0411154	00000000
00457030	00 00 00 00 00 00 00 00			0012F9E0	0411154	00000000
00457038	4C 15 00 00 00 00 00 00L..		0012FA00	0411154	7FFF0000
00457040	00 00 00 00 E0 E5 40 00E.....		0012FA40	0411154	00000000
00457048	00 00 00 00 B4 55 40 00B.....		0012FB00	0411154	00000000
0045704A	00 00 00 00 85 55 40 008.....		0012FB40	0411154	00000000
00457050	00 00 00 00 C4 54 40 00C.....		0012FB80	0411154	00000000
00457058	10 00 00 00 C4 54 40 001.....		0012FC00	0411154	00000000
00457060	00 00 00 00 00 00 00 00			0012FC40	0411154	00000000
00457068	11 00 00 00 94 54 40 001.....		0012FC80	0411154	FFFFFFFFFF
00457070	12 00 00 00 00 79 54 401.....		0012FCC0	0411154	End of SEH chain
00457078	13 00 00 00 00 00 00 00			0012FD00	0411154	SE handler
00457080	14 00 00 00 00 00 00 00			0012FD40	0411154	00000000
00457088	15 00 00 00 E4 53 40 001.....		0012FEC0	0411154	00000000
00457090	19 00 00 00 00 00 00 00			0012FEC0	0411154	77B1B2E5C
00457098	1A 00 00 00 HC 53 40 001.....		0012FEC0	0411154	RETURN to ntddi.77B1B2E5C from ntddi.77B1B2E5C

White box model



Sole line of defense:

Implementation

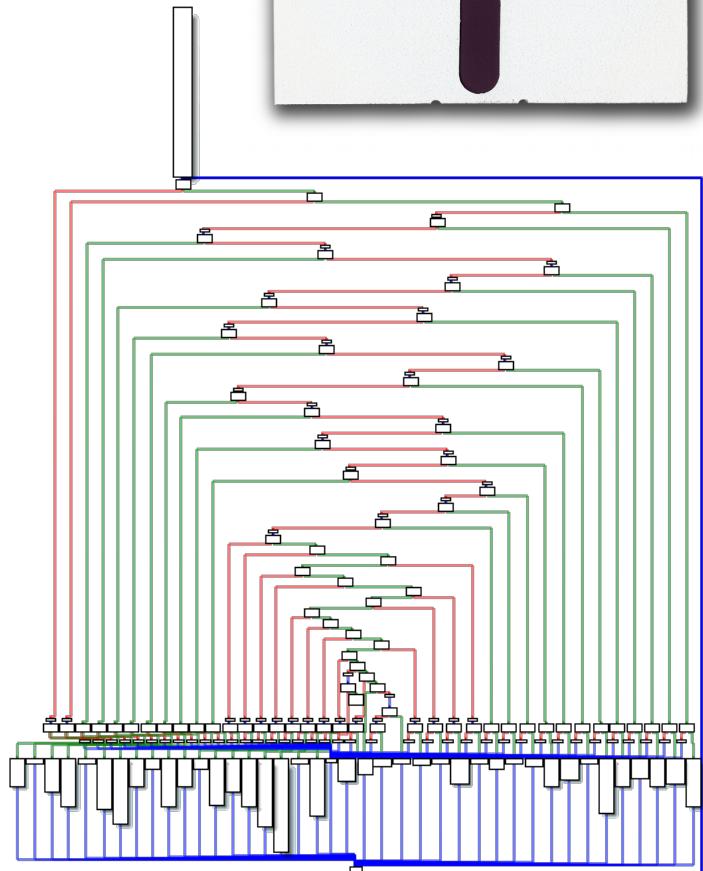
Usual countermeasures

Code obfuscation

Integrity checks

Anti-debug tricks

```
mov eax,0x0          mov ebx,[eax+0x80a051e]    mov edx,0x0
mov ax,[0x80a0451]    mov eax,[ebx]           mov dx,[eax+eax+0x80c0bba]
mov byte [eax+0x80e17bc],0x0  mov edx,0x0
mov al,[eax+0x80e17bc]  mov dx,[eax+eax+0x80c0bba]
mov [0x80a0451],al      mov [ebx],edx
mov eax,[0x80a0556]      mov eax,[0x80a0556]
mov edx,[eax+0x80a058e]  mov ebx,[eax+0x80a051e]
mov eax,[0x80a0451]      mov eax,[ebx]
mov eax,[eax+edx]
mov [0x80a044d],eax      mov edx,0x0
mov eax,[0x80a044d]      mov dx,[eax+eax+0x80c0bba]
mov eax,[eax+0x80a054e]  mov [ebx],edx
mov dword [eax],0x139     mov eax,[0x80a0556]
mov eax,[0x80a044d]      mov ebx,[eax+0x80a051e]
mov eax,[ebx]
```



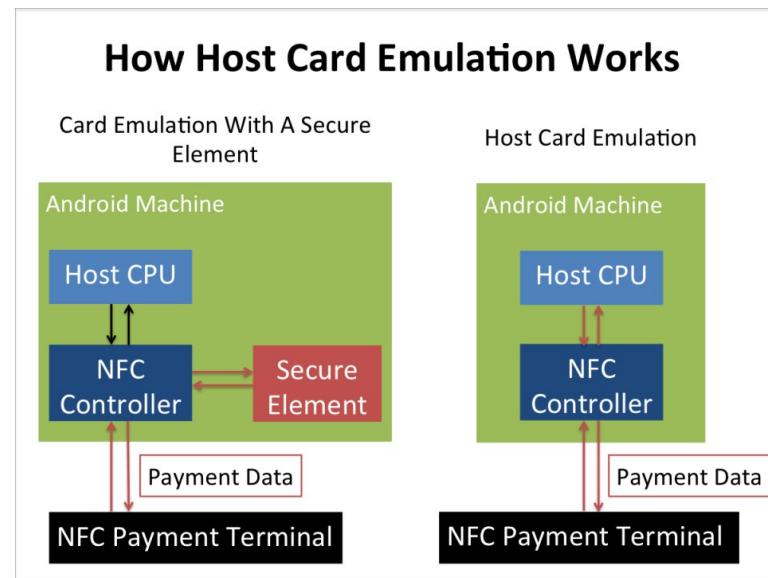
Cryptography under White-box model

What if you need to do some crypto in such hostile environment?

- DRM schemes ↔ criminals users
- Mobile payment, HCE ↔ malwares



Source: "l'industrie du film"



Source: Business Insider

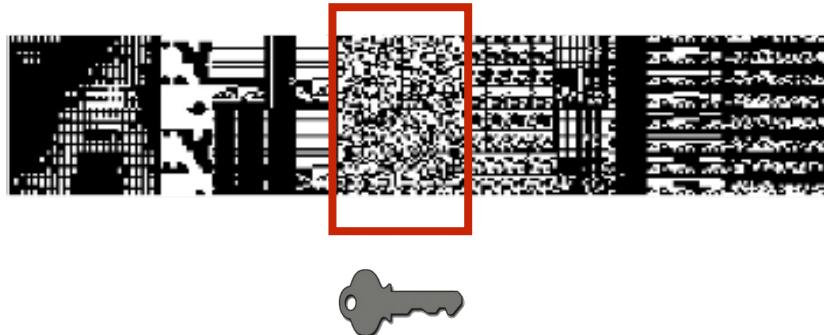
Cryptography under White-box model

What if you need to do some crypto in such hostile environment?

- DRM schemes \leftrightarrow criminals users
- Mobile payment, HCE \leftrightarrow malwares

Obfuscation techniques alone are mostly insufficient

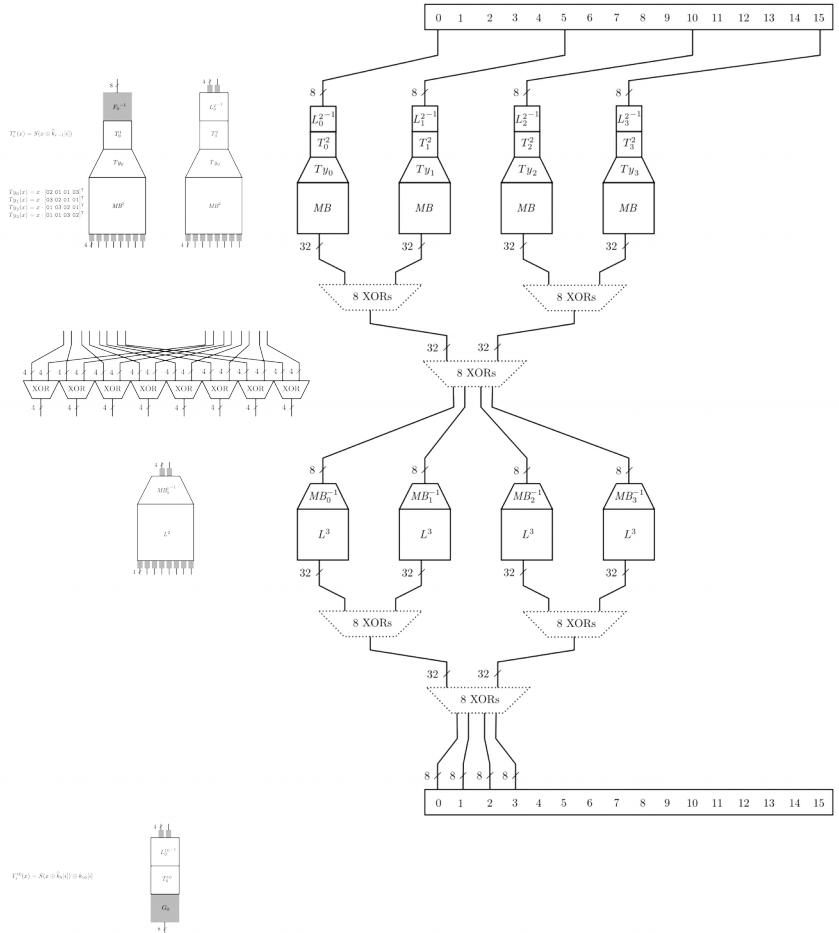
- Obfuscation mainly about securing code but here:
standard crypto algo in need for strong key protection
- E.g. entropy attack on RSA by Shamir and Van Someren (1999)



White-box cryptography

Chow et al. (2002)

- “Ideal” WB AES implementation:
One big lookup table
 4.94×10^{27} TB
- Practical WB AES:
Network of smaller tables
752kB
Encoding on intermediate values



White-box cryptography

History:

- Academic attacks → new designs → ...
- Today, all academic schemes have been broken

White-box cryptography

History:

- Academic attacks → new designs → ...
- Today, all academic schemes have been broken

Industry response:

- Keep white-box designs secret
- Bury white-box implementation under layers of code obfuscation, integrity checks, anti-debug tricks
- Some claim to be equivalent to a Secure Element



“Academic” attacks?

Require reversing of all the obfuscation layers
 Require knowledge on the design
 Then apply attack:

Definition 3. The mapping $\overline{AES}_{enc}^{(r,j)} : (\mathbf{F}_2^8)^4 \rightarrow (\mathbf{F}_2^8)^4$ for $1 \leq r \leq 9$ and $0 \leq j \leq 3$, called an encoded AES subround with byte permutations, is defined by

$$\overline{AES}_{enc}^{(r,j)} = (Q_0^{(r,j)}, Q_1^{(r,j)}, Q_2^{(r,j)}, Q_3^{(r,j)}) \circ \overline{AES}^{(r,j)} \circ (P_0^{(r,j)}, P_1^{(r,j)}, P_2^{(r,j)}, P_3^{(r,j)}) ,$$

where the mapping $\overline{AES}^{(r,j)}$ is defined by

$$\Pi_2^{(r,j)} \circ AES^{(r,\pi^{(r)}(j))} \circ \Pi_1^{(r,j)} = \text{MC}^{(r,j)} \circ (S, S, S, S) \circ \oplus_{[\bar{k}_i^{(r,j)}]_{0 \leq i \leq 3}} ,$$

$$\begin{aligned} \text{with } [\bar{k}_i^{(r,j)}]_{0 \leq i \leq 3} &= (\Pi_1^{(r,j)})^{-1}([k_i^{(r,\pi^{(r)}(j))}]_{0 \leq i \leq 3}) \\ \text{and } \text{MC}^{(r,j)} &= \Pi_2^{(r,j)} \circ \text{MC} \circ \Pi_1^{(r,j)} . \end{aligned}$$

As a result of the algorithm mentioned above, the white-box adversary has black-box access to the following structures of each round $R_r|_{r=1,\dots,10}$:

$$\left\{ \begin{array}{ll} \text{SR} \circ \bigoplus_{K'_{10}} \circ \{S_{10,i}\}_{i=0,\dots,15} \circ \bigoplus_{K_9} \circ A_9'^{-1} & \text{for } R_{10} , \\ \text{MC} \circ \text{SR} \circ A''_r \circ \{S_{r,i}\}_{i=0,\dots,15} \circ \bigoplus_{K_{r-1}} \circ A_{r-1}'^{-1} & \text{for } R_r |_{2 \leq r \leq 9} , \\ \text{MC} \circ \text{SR} \circ A''_1 \circ \{S_{1,i}\}_{i=0,\dots,15} \circ \bigoplus_{K_0} & \text{for } R_1 , \end{array} \right. \quad (5)$$

The second step then implements the function $\tau_{r,i}^k$ in which $\mu_r(n)$ describes the corresponding position of the bit in the output of the t-boxes, and PB is the DES p-box operation:

$$\begin{aligned} \tau_{r,i}^k(x)(L_r^i, R_r'^i) &= \left(\underbrace{\alpha_{r,i}^k(x|_{8\gamma_r(i)}^4, x|_{8\gamma_r(i)+4}, x|_{8\gamma_r(i)+5})}_{\text{depends on } R_{r-1} \text{ only}} \right) \parallel \\ &\quad \left(EP_i \left[PB \left(\underbrace{x|_{\gamma_r(0)}^4 \parallel x|_{\gamma_r(1)}^4 \parallel \dots \parallel x|_{\gamma_r(11)}^4}_{\text{depends on } R_{r-1} \text{ only}} \right) \oplus \left(\underbrace{x|_{\mu_r(0)} \parallel \dots \parallel x|_{\mu_r(32)}}_{\text{depends on } L_{r-1} \text{ only}} \right) \right] \right) \\ \tau_r^k(x) &= \tau_{r,0}^k(x) \parallel \tau_{r,1}^k(x) \parallel \dots \parallel \tau_{r,11}^k(x) \end{aligned}$$

ψ_r and ϕ_r are different non-linear bijective encodings on 4-bit blocks, and δ_r

$$\delta_r(L, R') = \gamma_r(\mu_r((L|0^{24}), R'))$$

$$\begin{aligned} \mu_r(x_0x_1\dots x_{47}, y_0\dots y_{47}) &= y_0\dots y_5x_{\mu_r^{-1}(0)}x_{\mu_r^{-1}(1)}y_6\dots y_{11}x_{\mu_r^{-1}(2)}x_{\mu_r^{-1}(3)}\dots y_{42}\dots y_{47}x_{\mu_r^{-1}(22)}x_{\mu_r^{-1}(23)}\dots x_{\mu_r^{-1}(47)} \\ \gamma_r(z_0z_1\dots z_{95}) &= z_{\gamma_r^{-1}(0)}\dots z_{(\gamma_r^{-1}(0)+5)}z_6z_7\dots z_{\gamma_r^{-1}(11)}\dots z_{(\gamma_r^{-1}(11)+5)}z_{94}z_{95} \end{aligned}$$

The obfuscated t-box is

$$T_r^k(x) = (\phi_r T_r^k \psi_{r-1}^{-1})(x).$$

Hence the transformed function is:

$$E^k(x) = \left[(\lambda^{-1} \delta_n^{-1} \psi_n^{-1}) \cdot (\psi_n \delta_n \tau_n^k \phi_n^{-1}) \cdot (\phi_n T_n^k \psi_{n-1}^{-1}) \cdot \dots \cdot (\psi_1 \delta_1 \tau_1^k \phi_1^{-1}) \cdot (\phi_1 T_1^k \psi_0^{-1}) \cdot (\psi_0 \delta_0 \beta \lambda) \right] (x)$$

with

$$\beta(L, R) = L \parallel EP(R)$$

By setting

$$\tau_r'^k = \begin{cases} \psi_0 \delta_0 \beta \lambda & r = 0 \\ \psi_r \delta_r \tau_r^k \phi_r^{-1} & r = 1, \dots, n \\ \lambda^{-1} \delta_n^{-1} \psi_n^{-1} & r = n + 1 \end{cases}$$

the resulting encryption operation is

$$E^k(x) = \left[\tau_{n+1}^k \cdot (\tau_n'^k \cdot T_n^k) \cdot \dots \cdot (\tau_1'^k \cdot T_1^k) \cdot \tau_0'^k \right] (x)$$

Excerpts:

- “Two Attacks on a White-Box AES”
- “Cryptanalysis of a Perturbated White-Box AES Implementation”
- “Attacking an obfuscated cipher by injecting faults”

“Academic” attacks?

= a lot of effort
then, anyway, for me:

Definition 3. The mapping $\overline{AES}_{enc}^{(r,j)} : (\mathbf{F}_2^8)^4 \rightarrow (\mathbf{F}_2^8)^4$ for $1 \leq r \leq 9$ and $0 \leq j \leq 3$, called an encoded AES subround with byte permutations, is defined by

$$\overline{AES}_{enc}^{(r,j)} = (Q_0^{(r,j)}, Q_1^{(r,j)}, Q_2^{(r,j)}, Q_3^{(r,j)}) \circ \overline{AES}^{(r,j)} \circ (P_0^{(r,j)}, P_1^{(r,j)}, P_2^{(r,j)}, P_3^{(r,j)}) ,$$

where the mapping $\overline{AES}^{(r,j)}$ is defined by

$$\begin{aligned} \Pi_2^{(r,j)} &= \Sigma^{(r,\pi^{(r)}(j))} \circ \Pi_1^{(r,j)} = \Sigma^{(r,j)} \circ (S, S, S, S) \circ \oplus_{[\bar{k}_i^{(r,j)}]_{0 \leq i \leq 3}} \\ &= \Sigma^{(r,j)} \circ \Pi_1^{(r,j)}. \end{aligned}$$

As a result, the algorithm mentioned above can be white-box adversarial, if it has black-box access to the following structures of each round $R_r |_{r=1, \dots, 10}$:

$$\left\{ \begin{array}{ll} \text{SR} \circ \bigoplus_{K'_{10}} \circ \{S_{10,i}\}_{i=0, \dots, 15} \circ \bigoplus_{K_9} \circ A_9'^{-1} & \text{for } R_{10}, \\ \text{MC} \circ \text{SR} \circ A_r'' \circ \{S_{r,i}\}_{i=0, \dots, 15} \circ \bigoplus_{K_{r-1}} \circ A_{r-1}'^{-1} & \text{for } R_r |_{2 \leq r \leq 9}, \\ \text{MC} \circ \text{SR} \circ A_1'' \circ \{S_{1,i}\}_{i=0, \dots, 15} \circ \bigoplus_{K_0} & \text{for } R_1, \end{array} \right. \quad (5)$$

The second step then implements the function $\tau_{r,i}^k$ in which $\mu_r(n)$ describes the corresponding position of the bit in the output of the t-boxes, and PB is the DES p-box operation:

$$\begin{aligned} \tau_{r,i}^k(x)(L_r^i, R_r'^i) &= \left(\underbrace{\alpha_{r,i}^k(x|_{8\gamma_r(i)}^4, x|_{8\gamma_r(i)+4}, x|_{8\gamma_r(i)+5})}_{\text{depends on } R_{r-1} \text{ only}} \right) \parallel \\ &\quad \left(EP_i \left[PB \left(\underbrace{x|_{\gamma_r(0)}^4 \parallel x|_{\gamma_r(1)}^4 \parallel \dots \parallel x|_{\gamma_r(11)}^4}_{\text{depends on } R_{r-1} \text{ only}} \right) \oplus \left(\underbrace{x|_{\mu_r(0)} \parallel \dots \parallel x|_{\mu_r(32)}}_{\text{depends on } L_r \text{ only}} \right) \right) \right) \\ \tau_r^k(x) &= \tau_{r,0}^k(x) \parallel \tau_{r,1}^k(x) \parallel \dots \parallel \tau_{r,11}^k(x) \end{aligned}$$

ψ_r and ϕ_r are different non-linear bijective encodings on 4-bit blocks, and δ_r

$$\begin{aligned} \mu_r(L|0^{24}) &= \gamma_r(\mu_r((L|0^{24}), \\ \mu_r(z_{\gamma_r^{-1}(1)} \dots z_{\gamma_r^{-1}(47)}) &= y_0 \dots y_5 x_{\mu_r^{-1}(0)} x_{\mu_r^{-1}(1)} \dots x_{\mu_r^{-1}(2)} x_{\mu_r^{-1}(3)} \dots y_{42} \dots z_{\gamma_r^{-1}(12)} \dots z_{\gamma_r^{-1}(13)} \dots x_{\mu_r^{-1}(47)} \\ \gamma_r(z_{\gamma_r^{-1}(1)} \dots z_{\gamma_r^{-1}(47)}) &= z_{\gamma_r^{-1}(0)} \dots z_{\gamma_r^{-1}(46)} \dots z_{\gamma_r^{-1}(11)} \dots z_{\gamma_r^{-1}(11)+5} z_{94} z_{95} \end{aligned}$$

The operation $\beta(L, R)$ is

$$T_r^k(x) = \phi_r(\beta(\psi_r^{-1}(L), \psi_r^{-1}(R)))$$

the transformed function is:

$$E^k(x) = \left[(\lambda^{-1} \delta_n^{-1} \psi_n^{-1}) \cdot (\psi_n \delta_n \tau_n^k \phi_n^{-1}) \cdot (\phi_n T_n^k \psi_{n-1}^{-1}) \cdot \dots \cdot (\psi_1 \delta_1 \tau_1^k \phi_1^{-1}) \cdot (\phi_1 T_1^k \psi_0^{-1}) \cdot (\psi_0 \delta_0 \beta \lambda) \right] (x)$$

with

$$\beta(L, R) = L \parallel EP(R)$$

By setting

$$\tau_r'^k = \begin{cases} \psi_0 \delta_0 \beta \lambda & r = 0 \\ \psi_r \delta_r \tau_r^k \phi_r^{-1} & r = 1, \dots, n \\ \lambda^{-1} \delta_n^{-1} \psi_n^{-1} & r = n + 1 \end{cases}$$

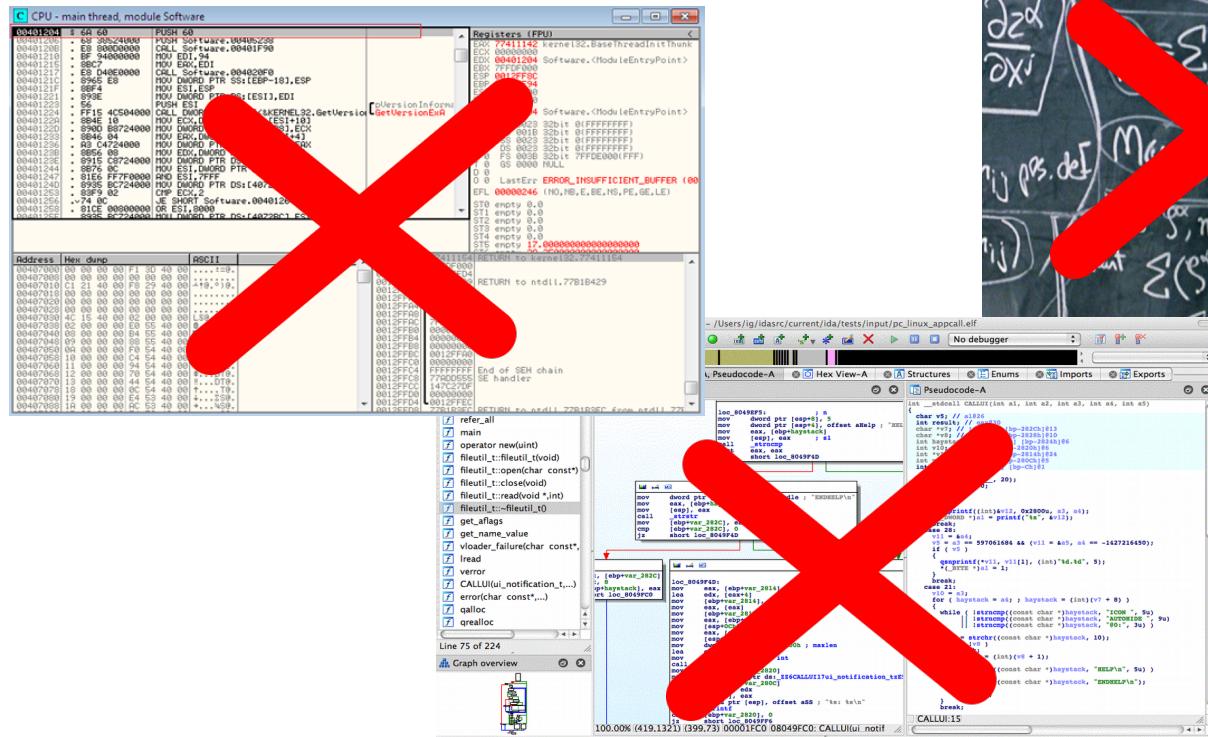
the resulting encryption operation is

$$E^k(x) = \left[\tau_{n+1}^k \cdot (\tau_n'^k \cdot T_n^k) \cdot \dots \cdot (\tau_1'^k \cdot T_1^k) \cdot \tau_0'^k \right] (x)$$

Our goal

Recover white-box keys

- without much reverse-engineering effort
 - without much intellectual effort ^^



SOFTWARE EXECUTION TRACES

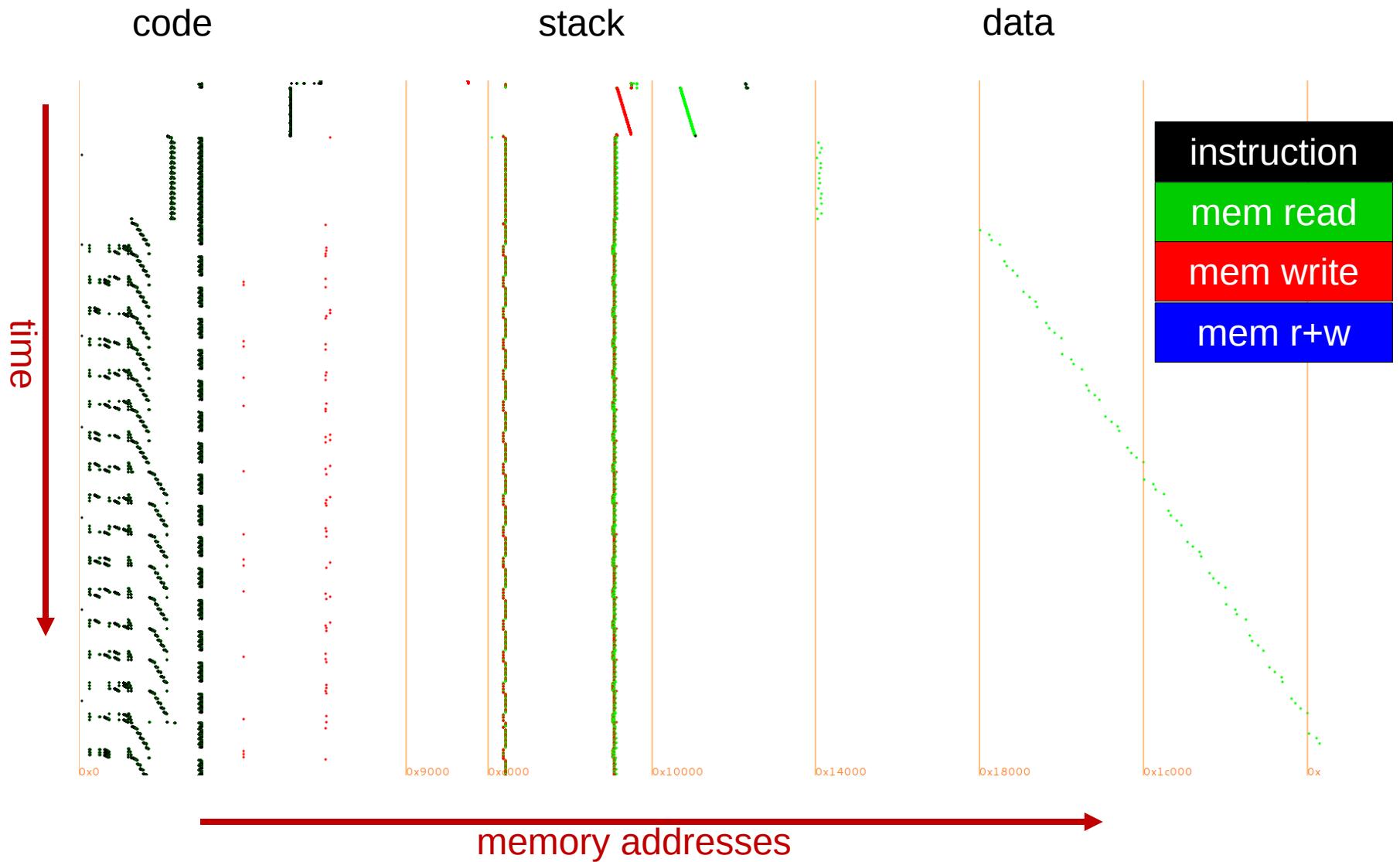
Tracing binaries

Record all instructions and memory accesses

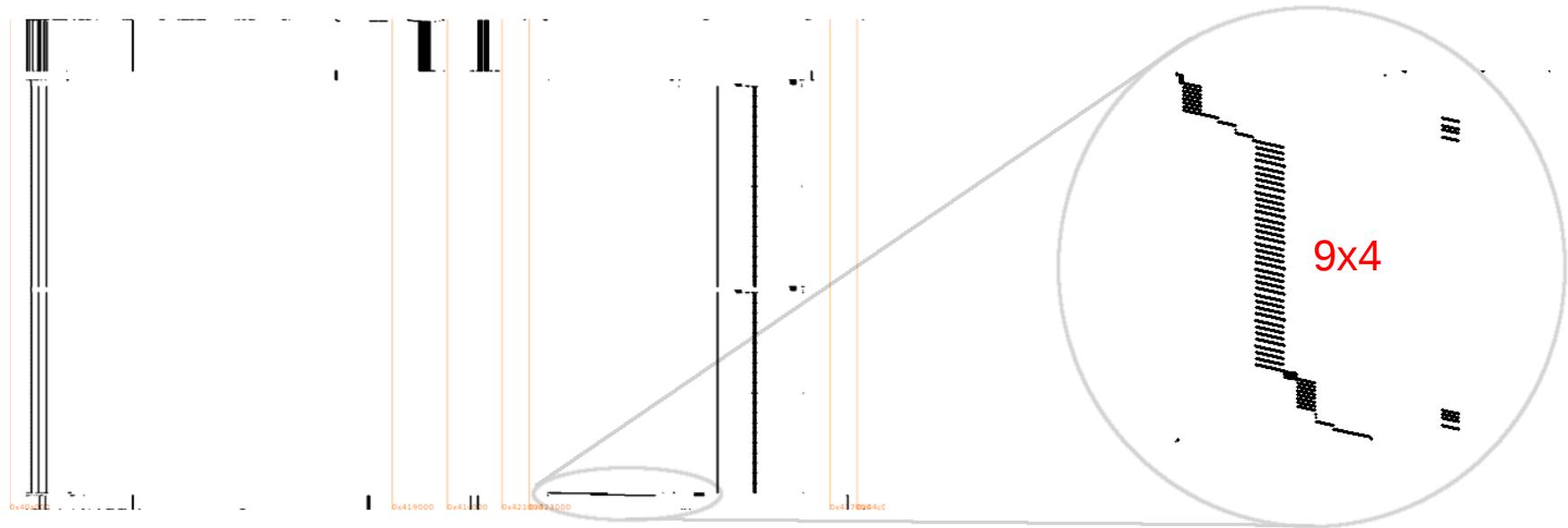
Examples:

- Intel PIN (x86, x86-64, Linux, Windows, Wine/Linux)
- Valgrind (idem+ARM, Android)
- Add hooks to VM (Java, Python,...)
- Add hooks to emulators

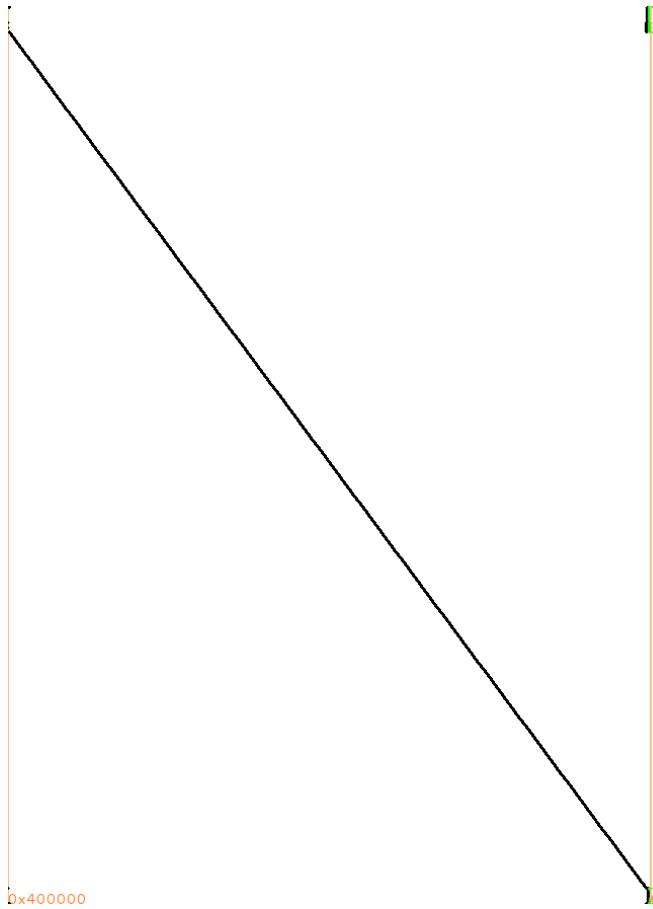
Trace convention: Quarkslab's pTra waterfall



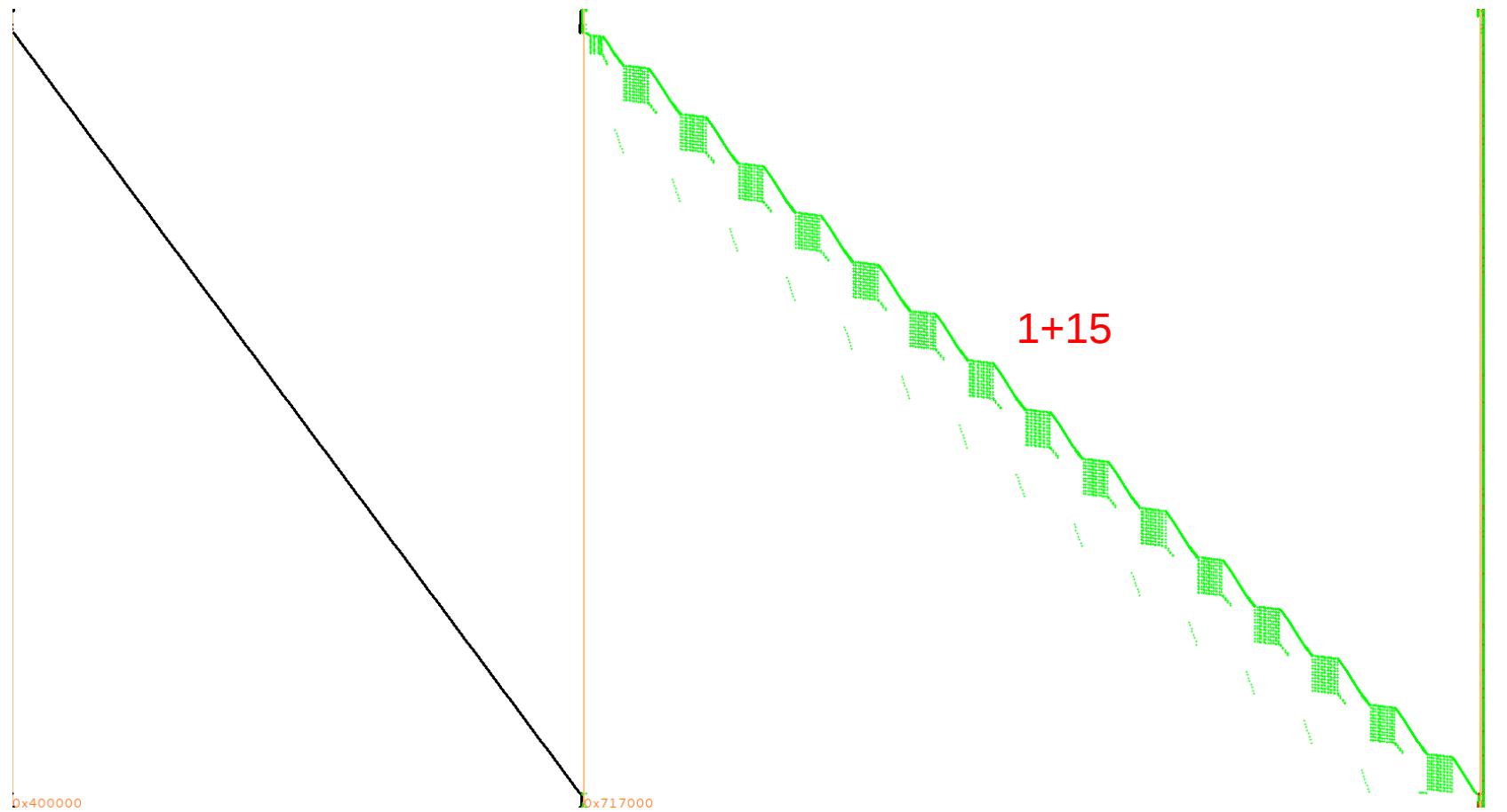
Visual crypto identification: code



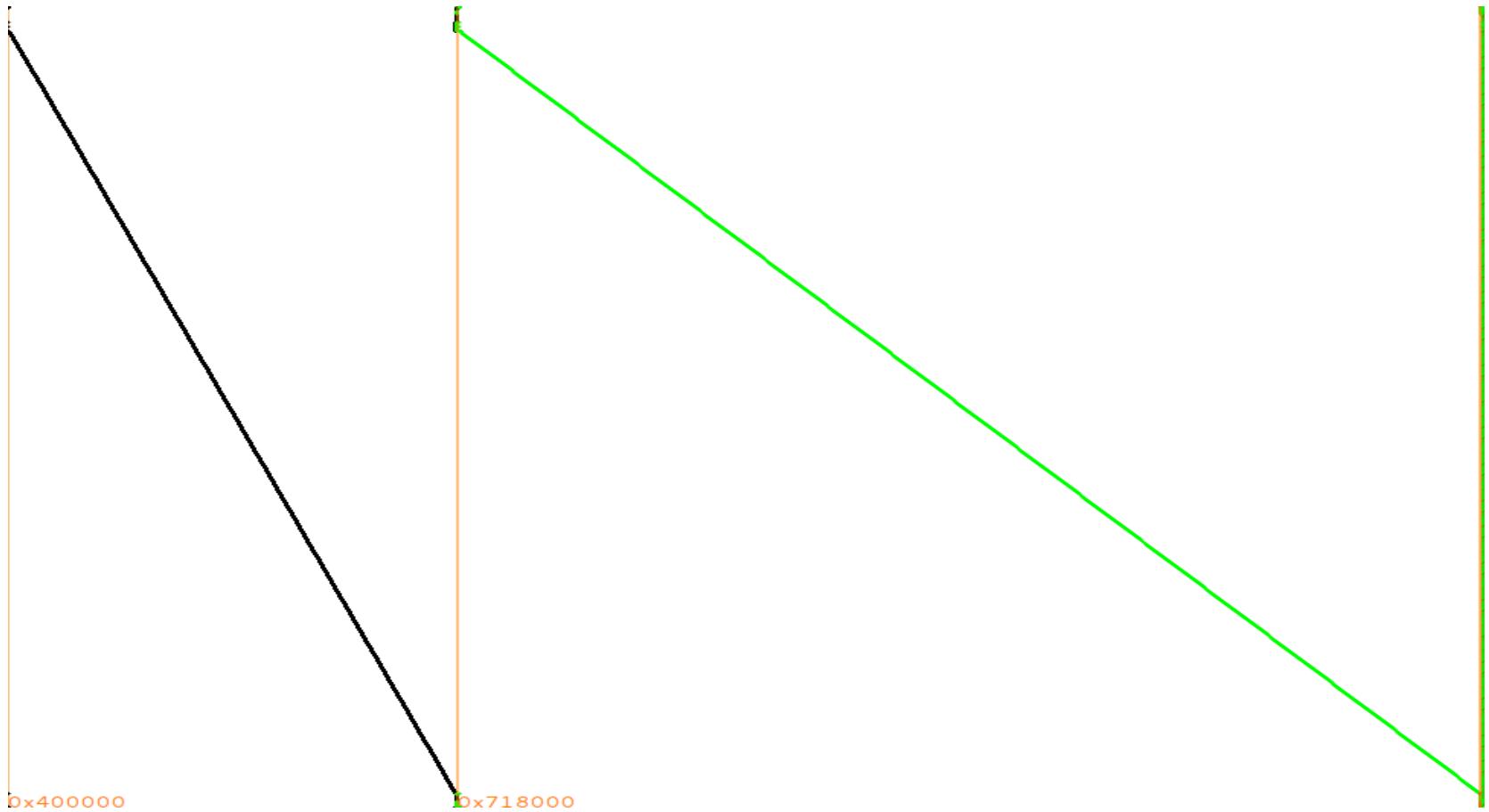
Visual crypto identification: code?



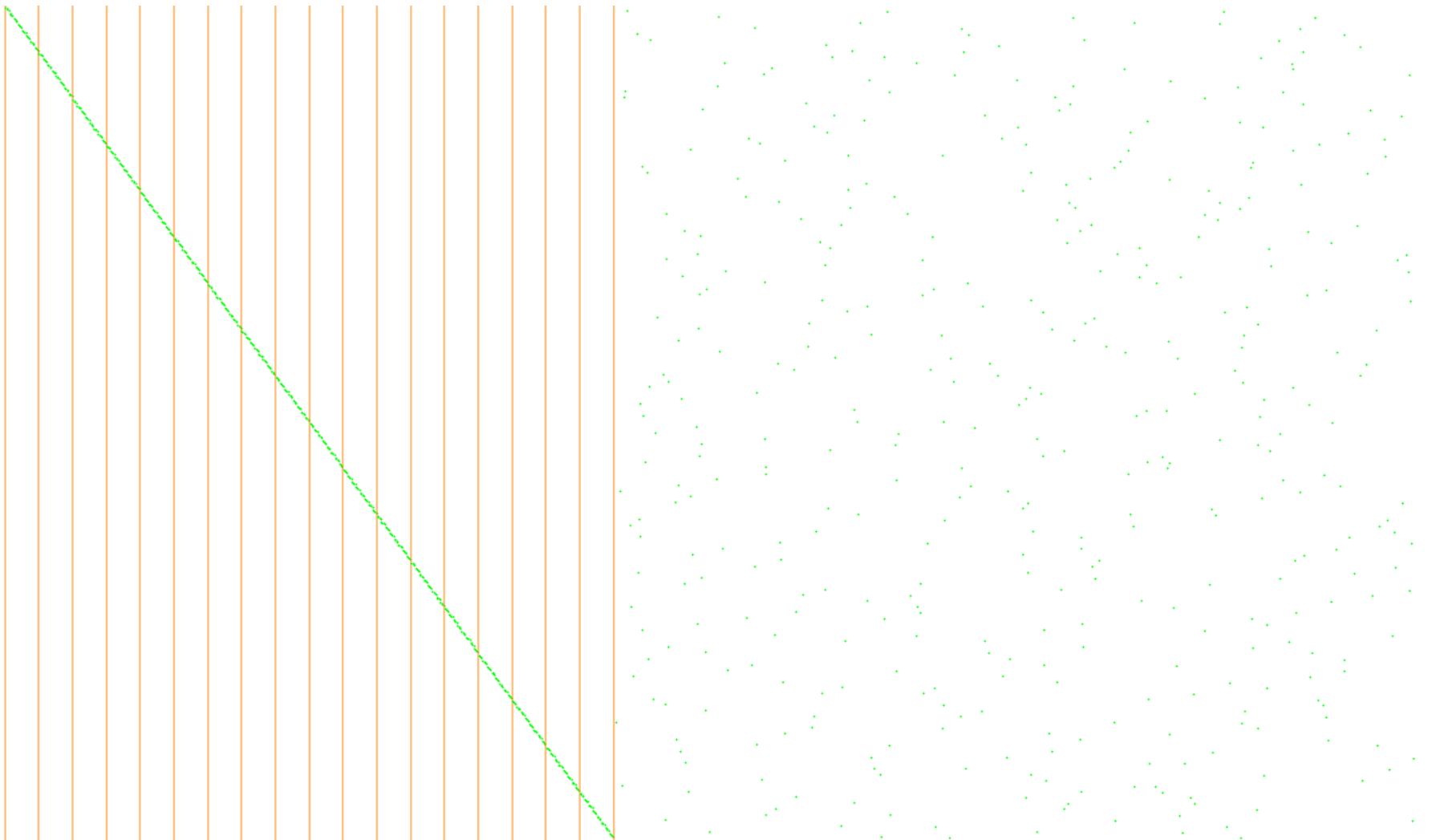
Visual crypto identification: code? data!



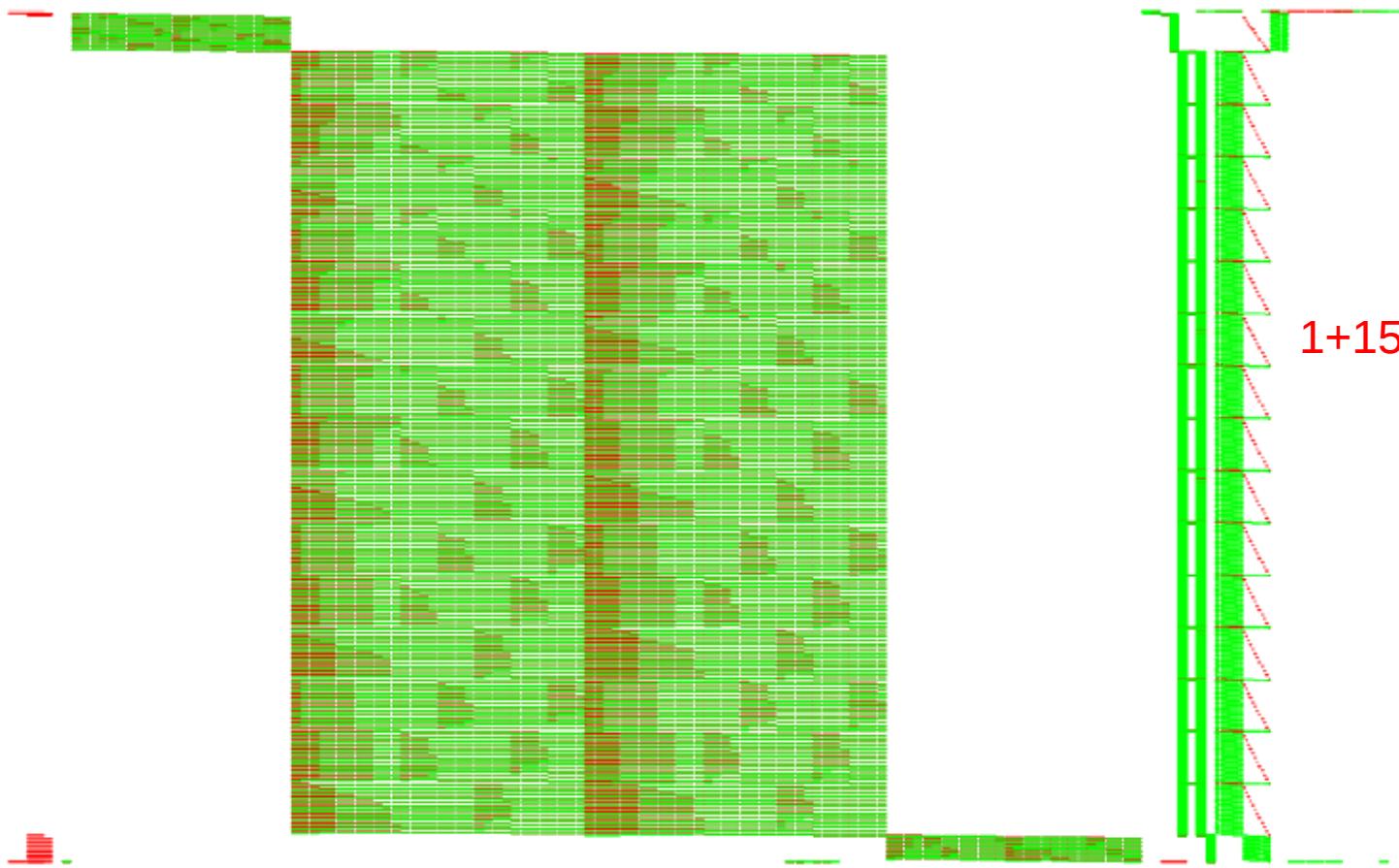
Visual crypto identification: code? data?



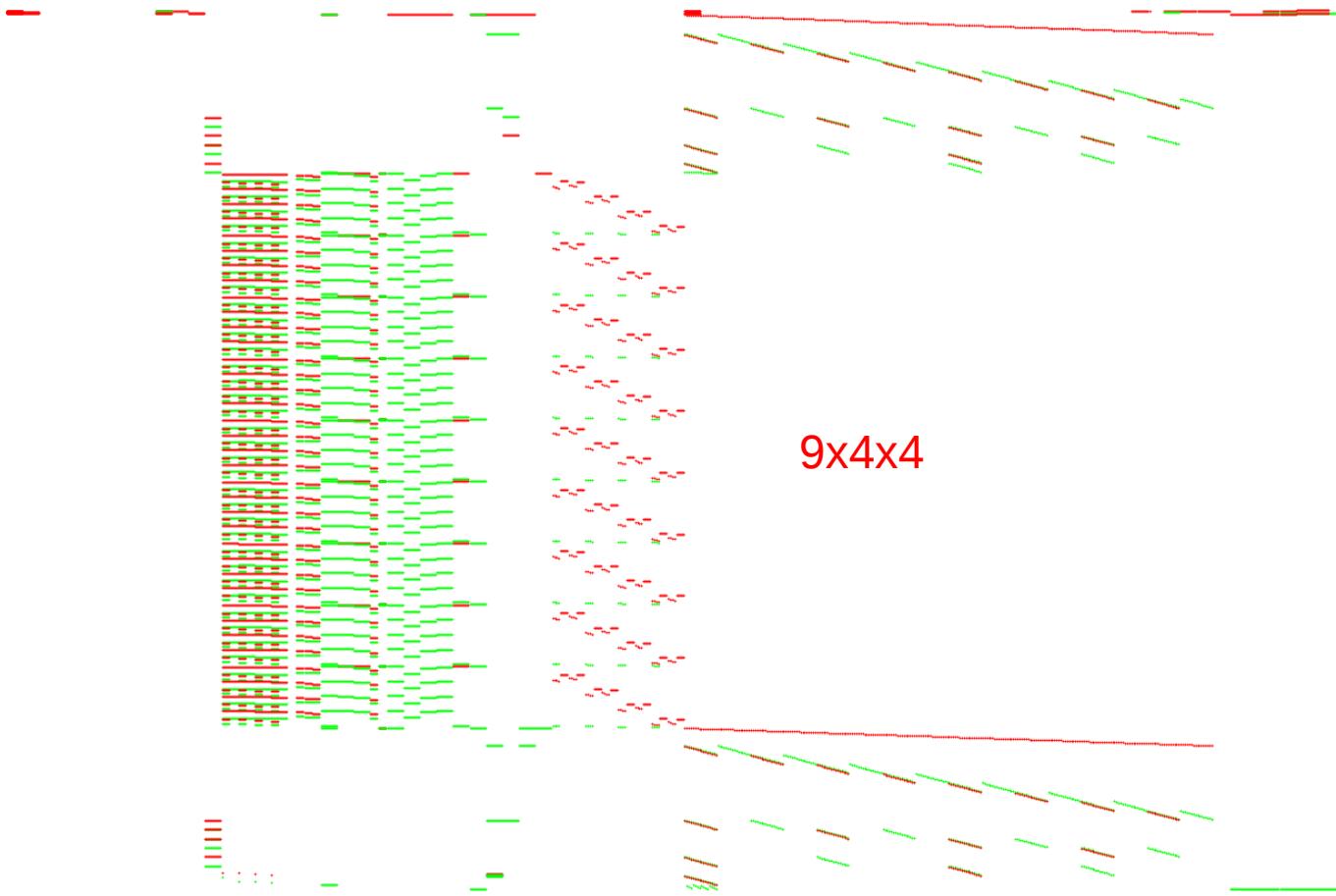
Visual crypto identification: data?



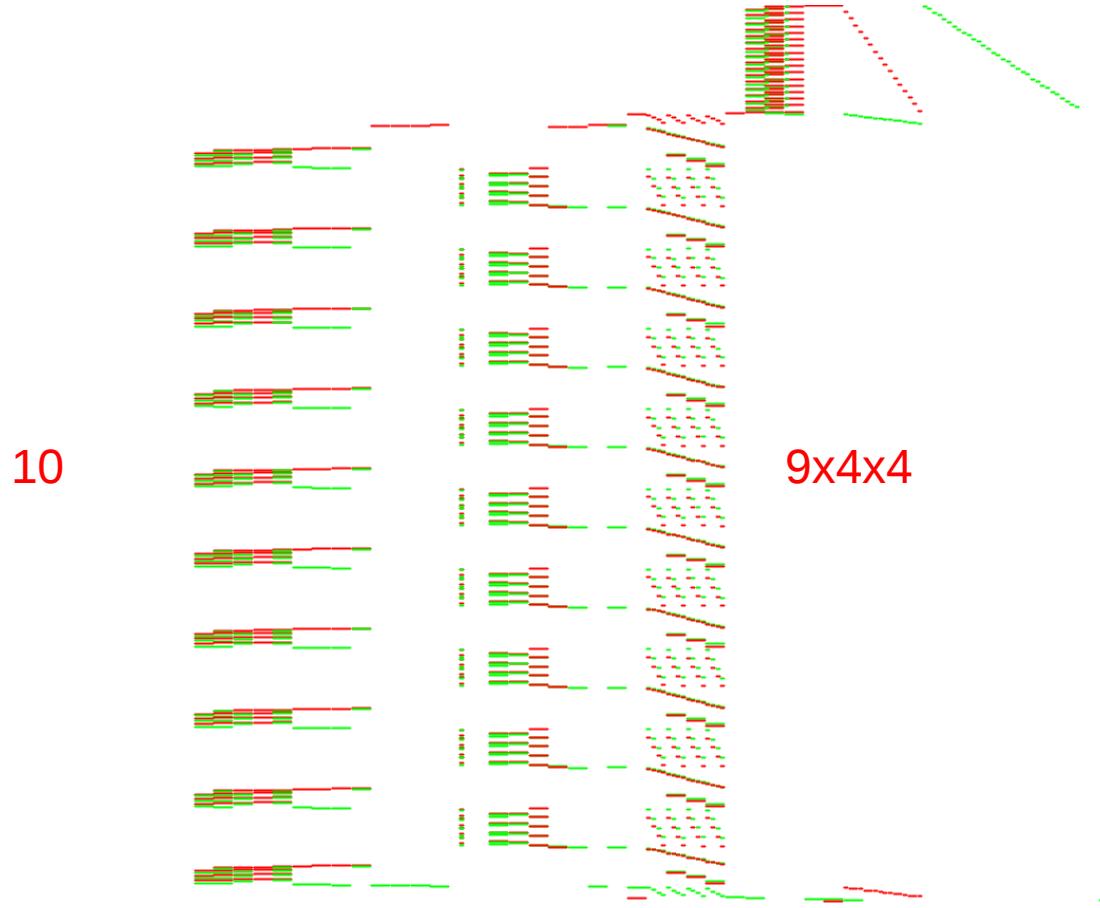
Visual crypto identification: stack!



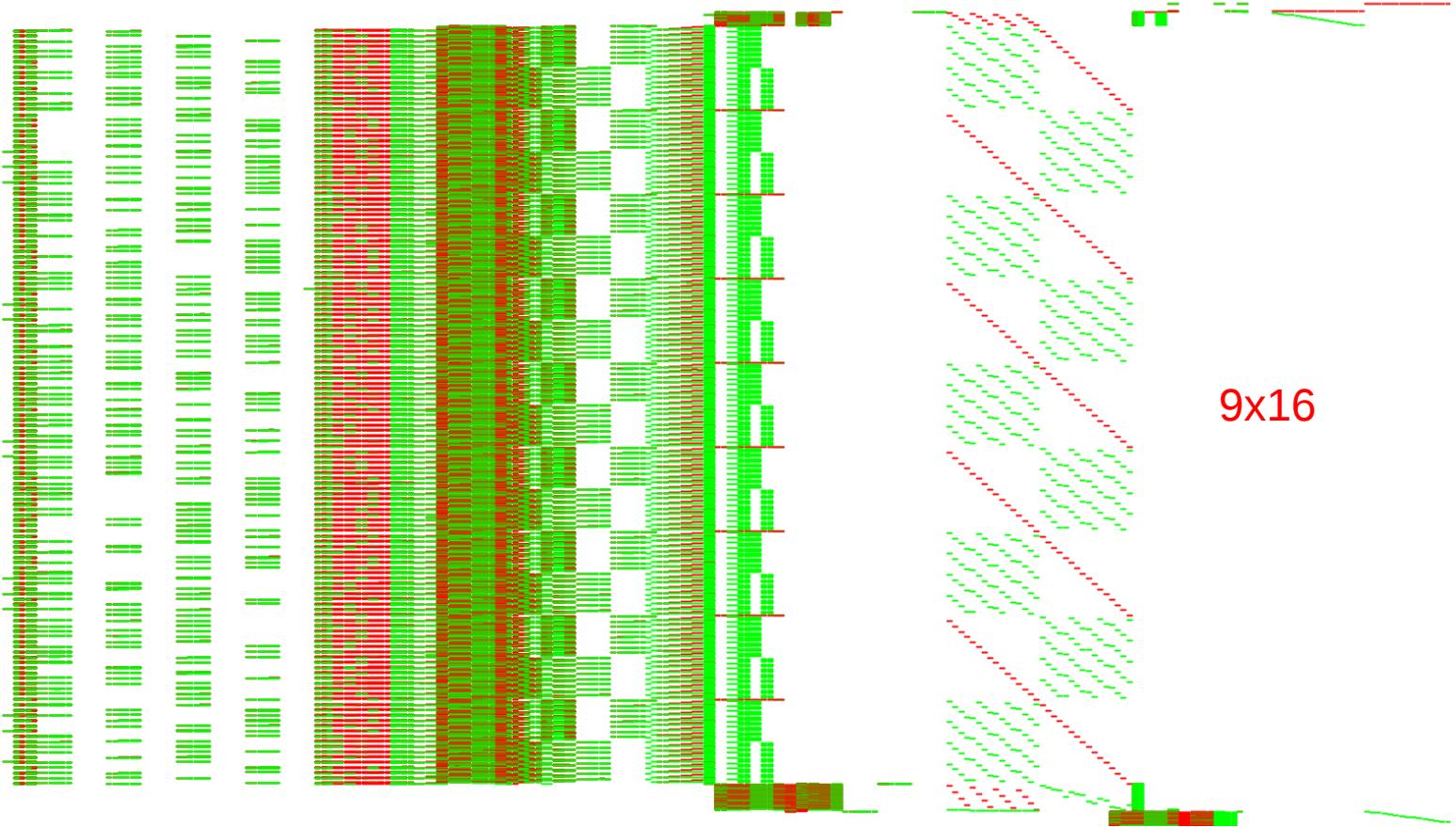
Visual crypto identification: stack!



Visual crypto identification: stack!



Visual crypto identification: stack!



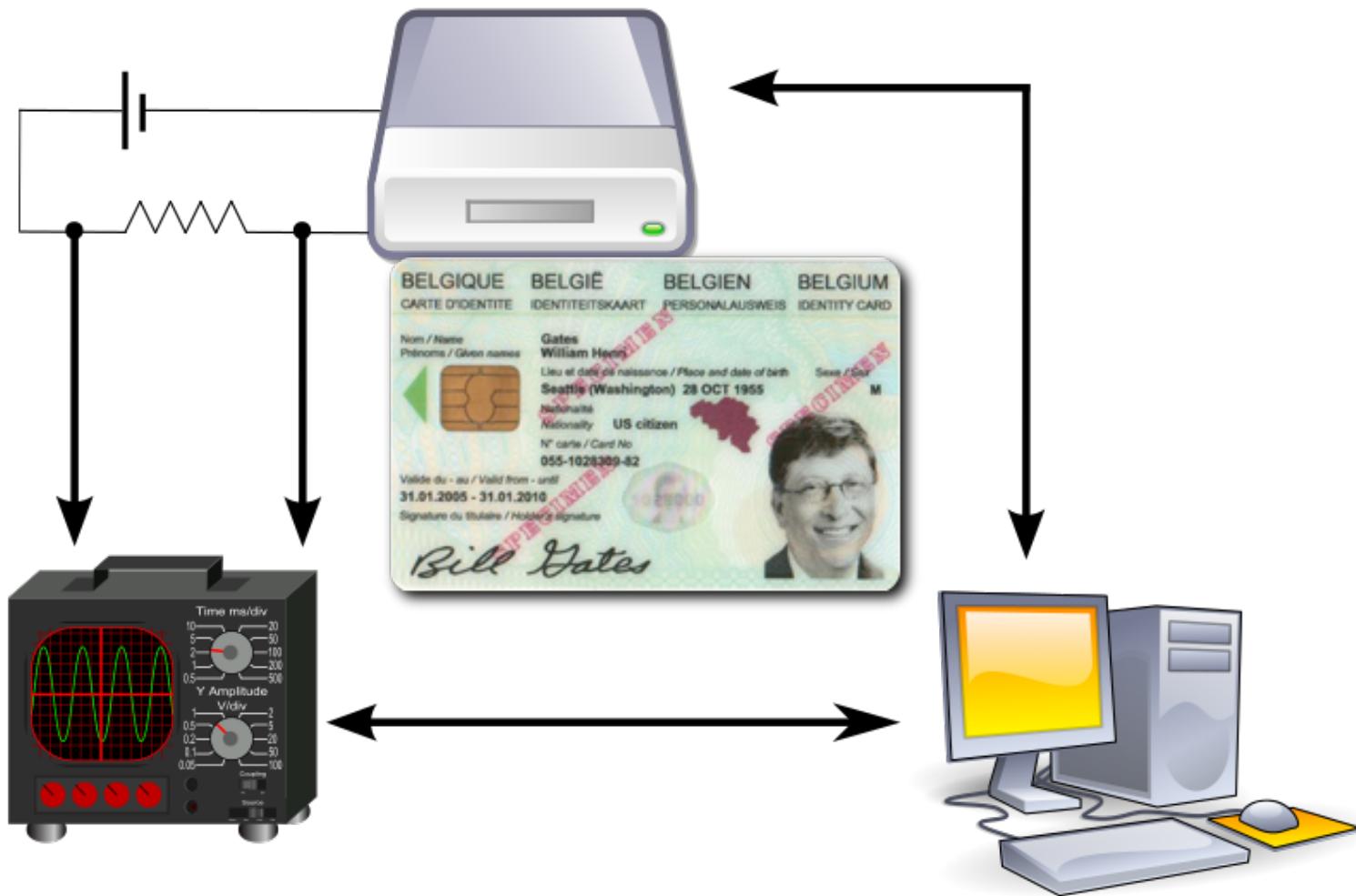


So What?

Where is my key?

DIFFERENTIAL COMPUTATION ANALYSIS

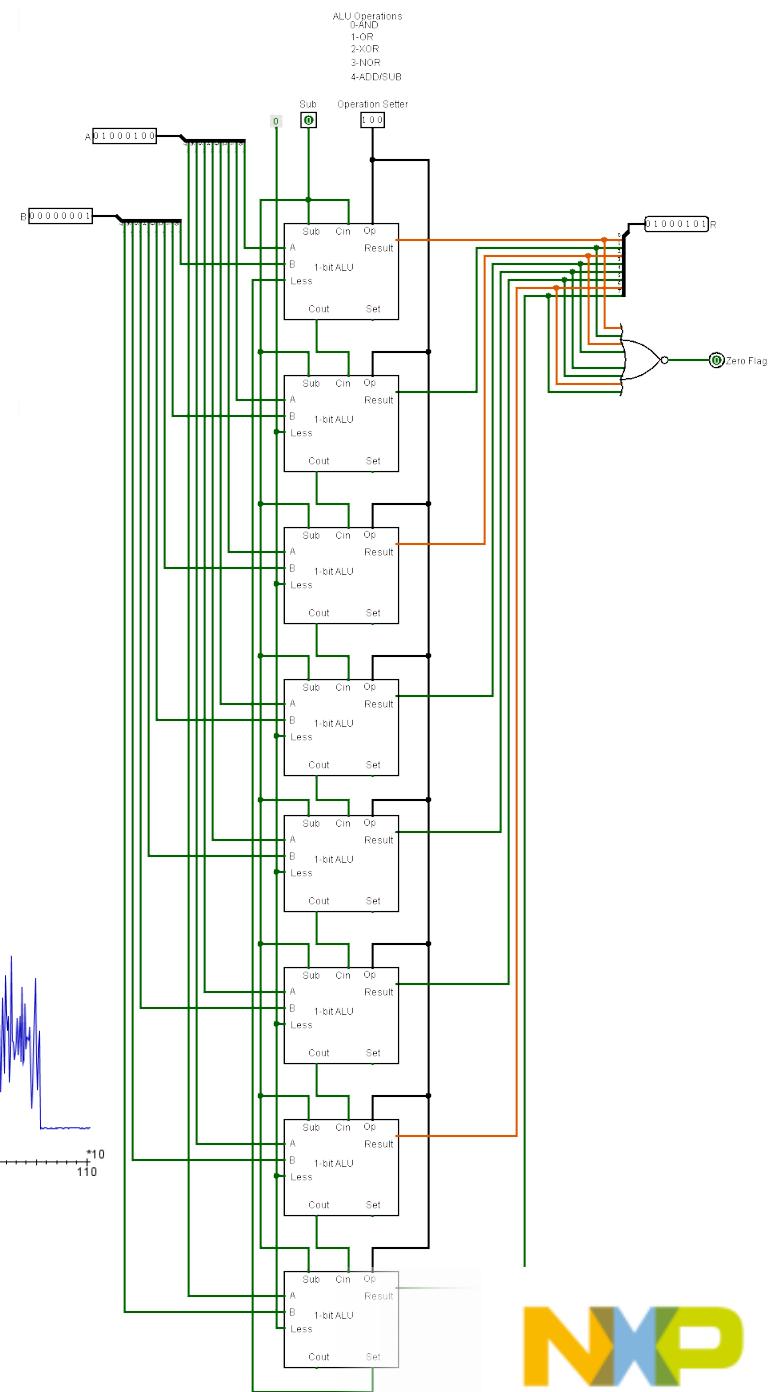
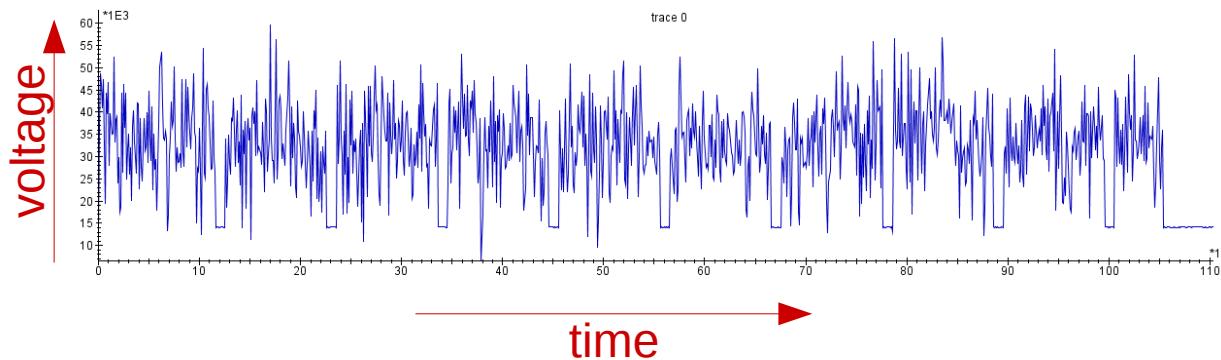
Remember?



All started with Differential Power Analysis

by P. Kocher et al. (1998)

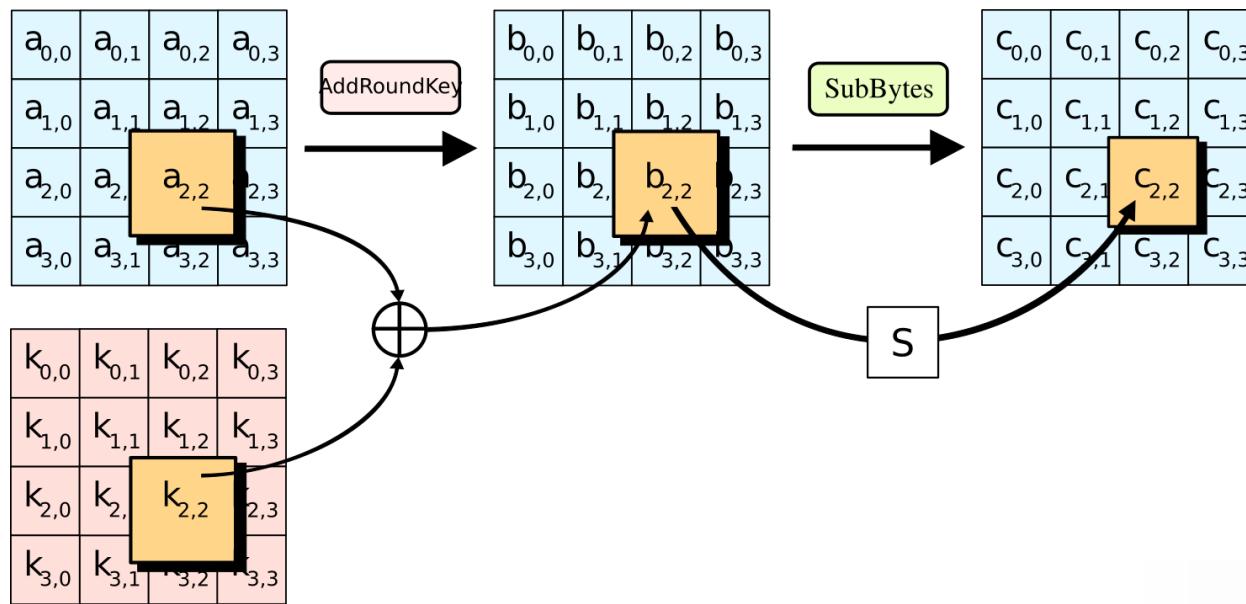
- Probable correlations:
power consumption vs.
Hamming weight of internal values
 - Record many traces
while providing different inputs



Differential Power Analysis

Some intermediate values in first (or last) round depend only on known data and a fraction of the round key

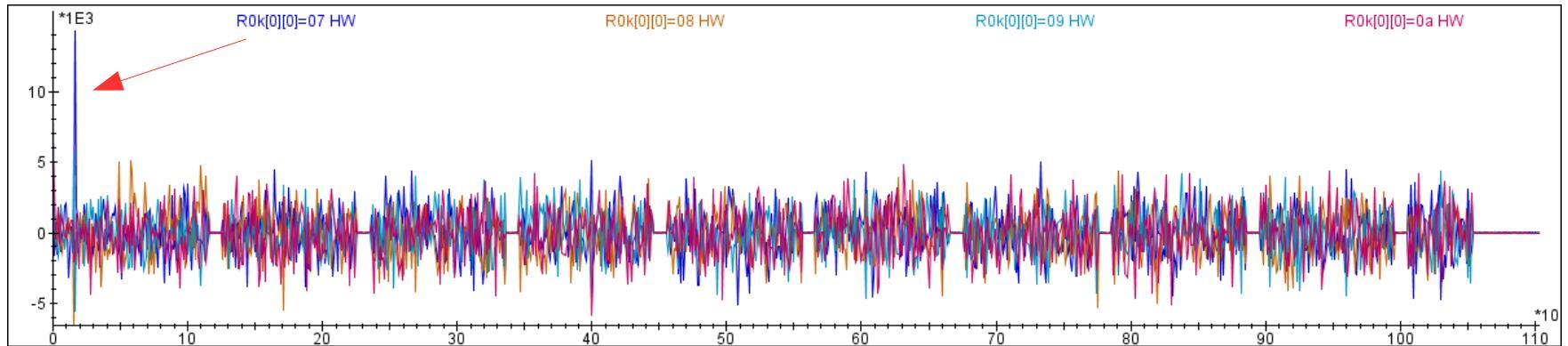
E.g. for AES:



Differential Power Analysis

- 1) Make a guess on that fraction of key
- 2) Evaluate targeted intermediate value for each plaintext: 0 or 1?
- 3) Sort traces accordingly in two buckets and average them
- 4) Compute differences between those averages

If the key guess is correct, it'll show up:



Differential Power Analysis

Very powerful grey box attack!

Requirements:

- Either known input or known output
- Ability to trace power consumption (or EM radiations)
- Some leakage

Differential Computation Analysis

Port the white-box to a smartcard and measure power consumption

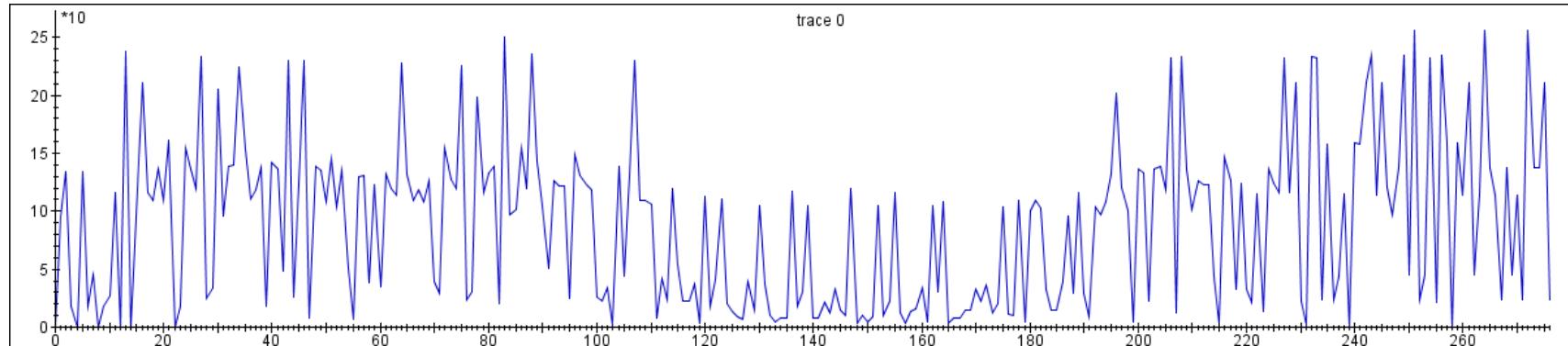
Differential Computation Analysis

Port the white-box to a smartcard and measure power consumption

Software execution traces → “power traces”

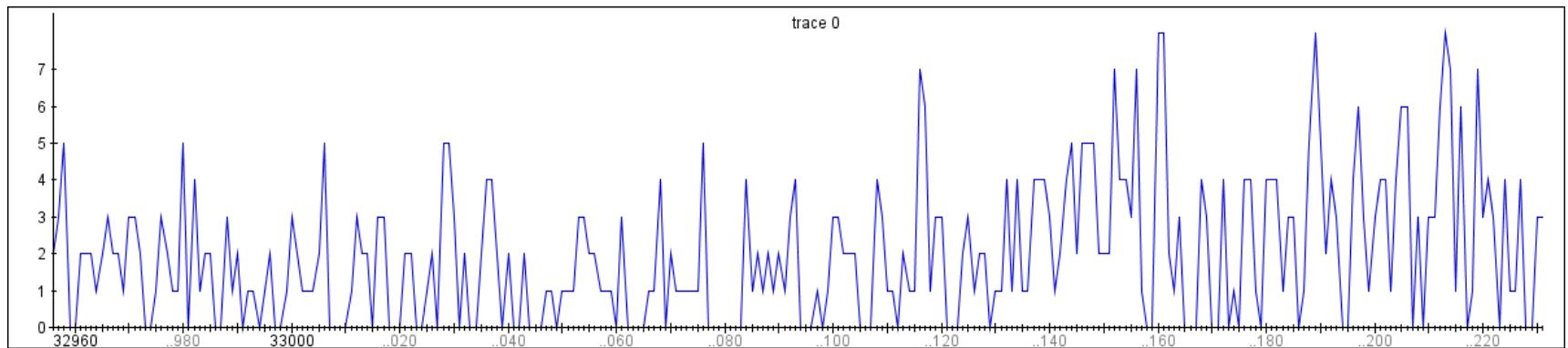
Memory accesses / data / stack writes / ...

E.g. build a trace of all 8-bit data reads:



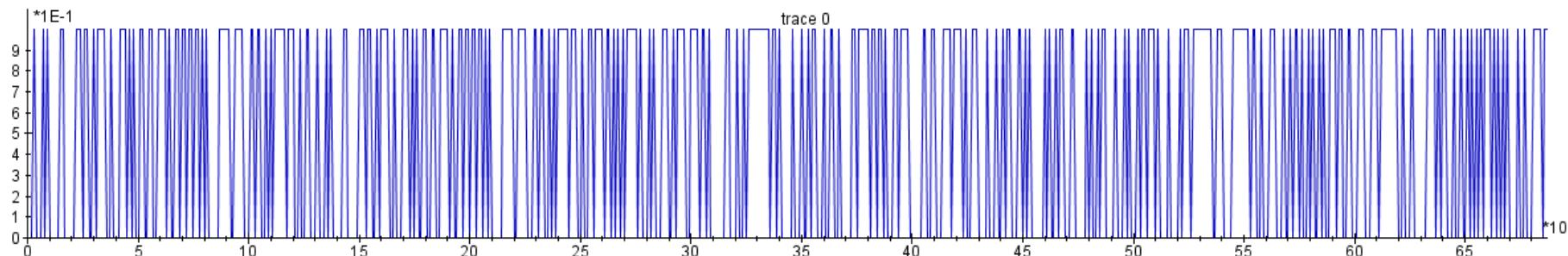
Differential Computation Analysis

→ Build Hamming weight traces?

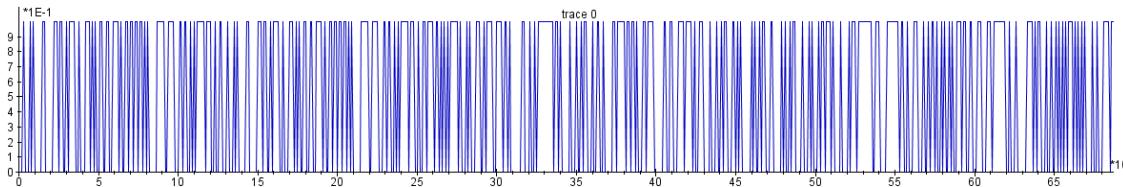


Differential Computation Analysis

→ Serialize bytes in a succession of bits

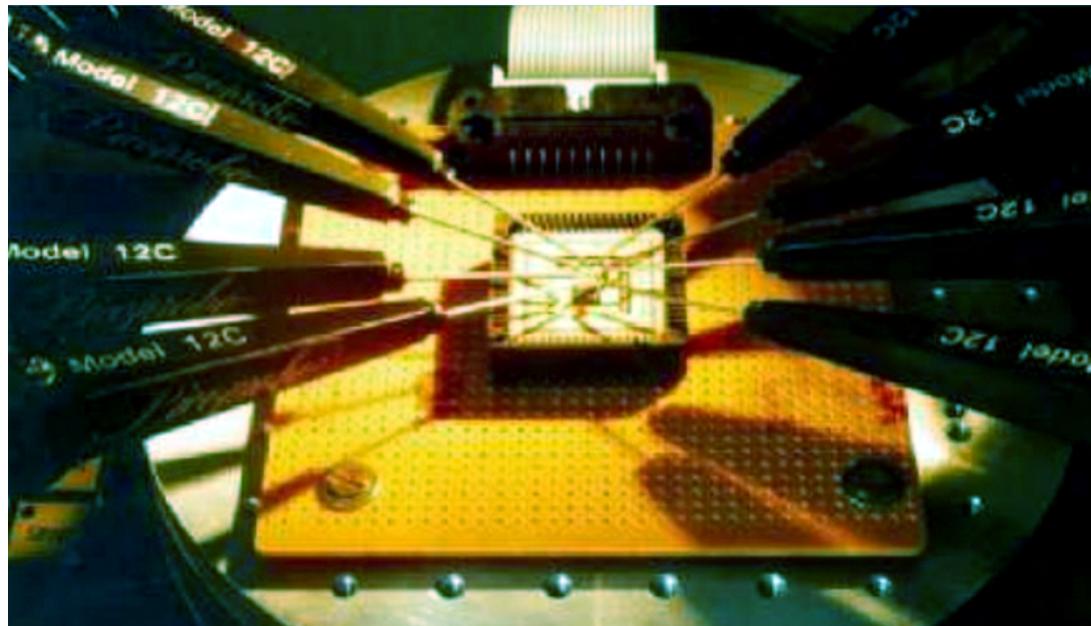


Differential Computation Analysis



Looks weird but works great!

As if:



Next step

Feed traces in your favorite DPA tool

- Riscure Inspector SCA software
- ChipWhisperer opensource software
- Matlab...
- **Daredevil !**

Tips

What to trace?

- Stack writes
- Data reads
- Accessed addresses

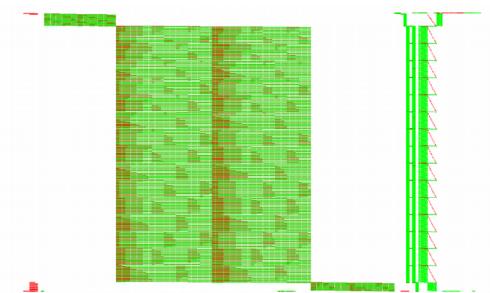
plaintexts and/or ciphertexts

- May require binary instrumentation

Large white-box? Minimize amount of traced information

- Trace only first (or last) round
- Standard deviation analysis to compress the trace

Wyseur challenge



by Brecht Wyseur, 2007

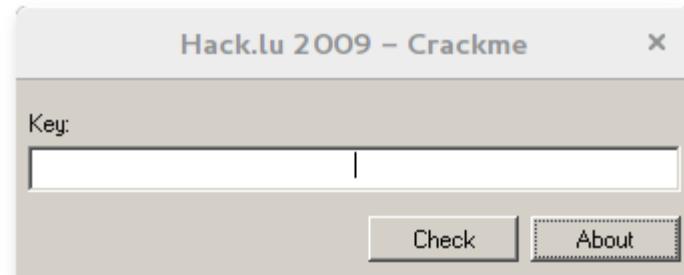
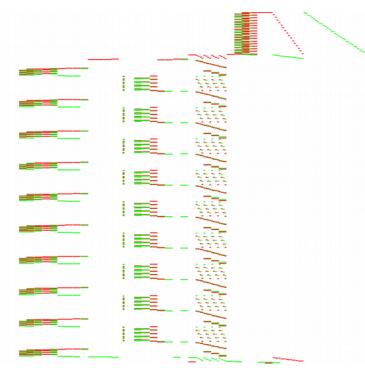
DES implementation based on Chow “plus some personal improvements”

Downloading Linux binary...

1h and 65 traces later (of a full binary execution), key got broken!

Hack.lu 2009 challenge

Windows *crackme* by Jean-Baptiste Bédrune
AES implementation based on Chow



Laziness → Wine/Linux + xdotool (kbd+mouse emulation)
16 traces
(CTF challenge, no internal encodings)

SSTIC 2012 challenge

Python white-box by Axel Tillequin
DES implementation in a marshalled object

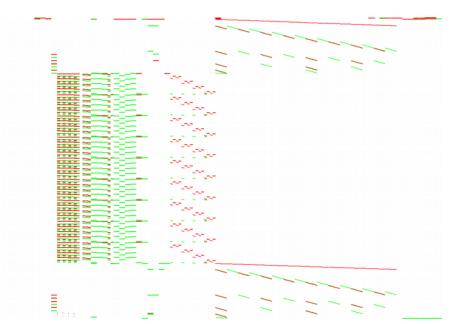
Python + PIN = Boom

→ Instrumenting “Bits” helper class

Again, 16 traces

Again, no internal encodings

Karroumi



Latest academic attempt to “fix” Chow (2011)

Dual Ciphers, i.e. isomorphic AES ciphers:

$$\forall p, k : E_k(p) = f^{-1} \left(E'_{g(k)}(h(p)) \right)$$

Our own binary challenge...

2000 traces, 500 traces after some tuning

Some proprietary white-boxes

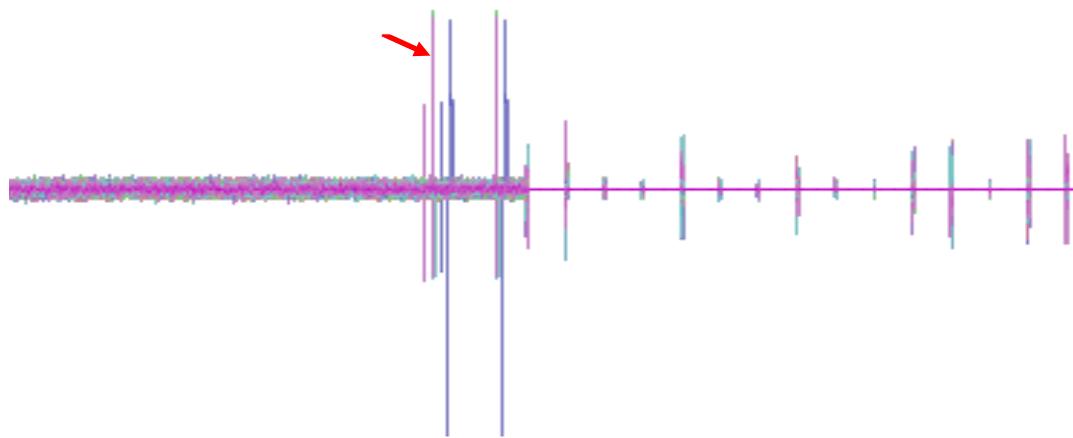
DES & AES

Broken in 200 to 2500 traces



Back to White-Box design

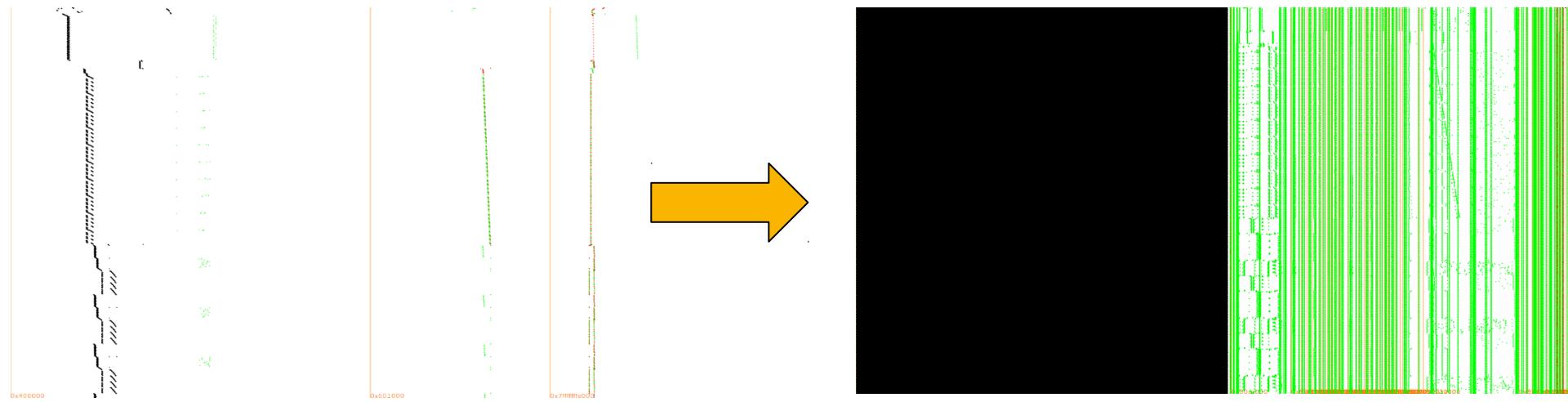
Known key analysis



- 1) Identify first leaking samples (the original source)
- 2) Find the corresponding instruction
- 3) Find the corresponding source code line

Works also on obfuscated VMs: Mlo/Vfuscator2

Applied on a standard AES implementation

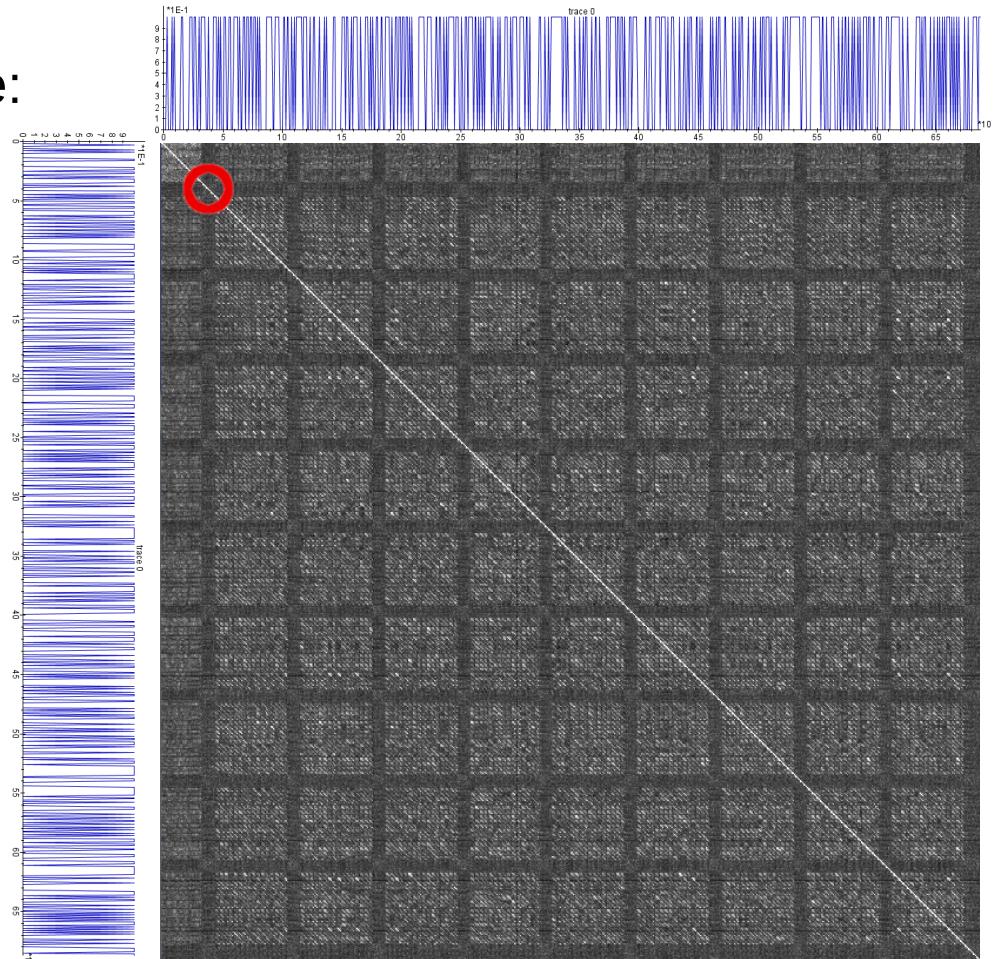
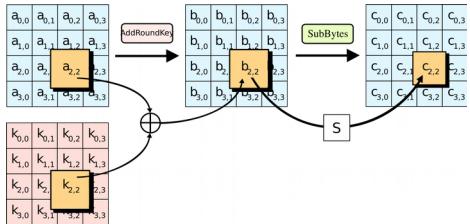


M/o/Vfuscator2 on AES

Auto-correlation reveals structure:

Huge traces, compressed by looking at standard deviation
4Mb -> 6.6kb

First round Sbox output



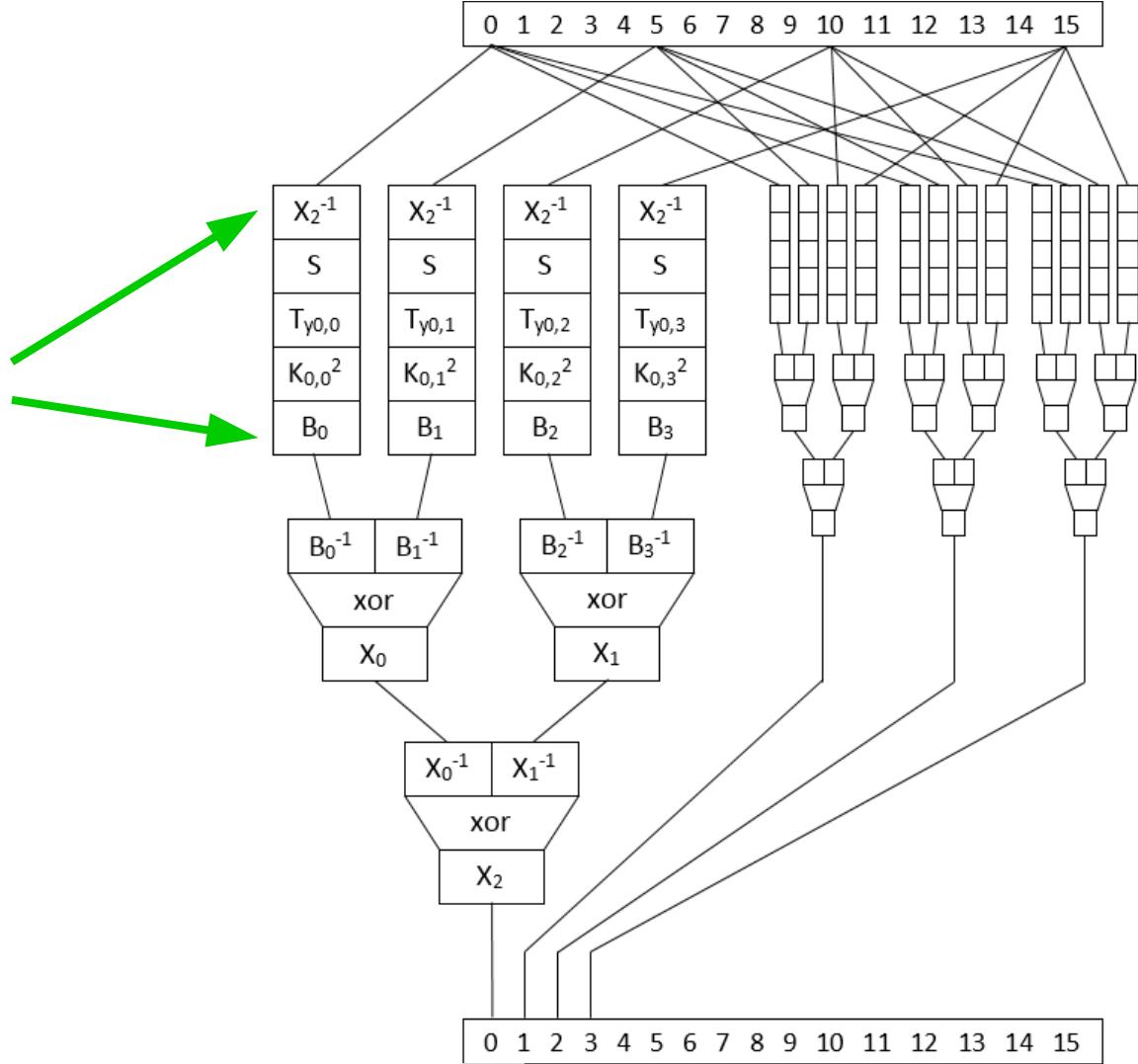
20 – 30 traces

http://wiki.yobi.be/wiki/MoVfuscator_Writeup

Can DCA fail?

Yes!

Wide intermediate
non-linear encodings (8x8)
blind the SBox non-linearity



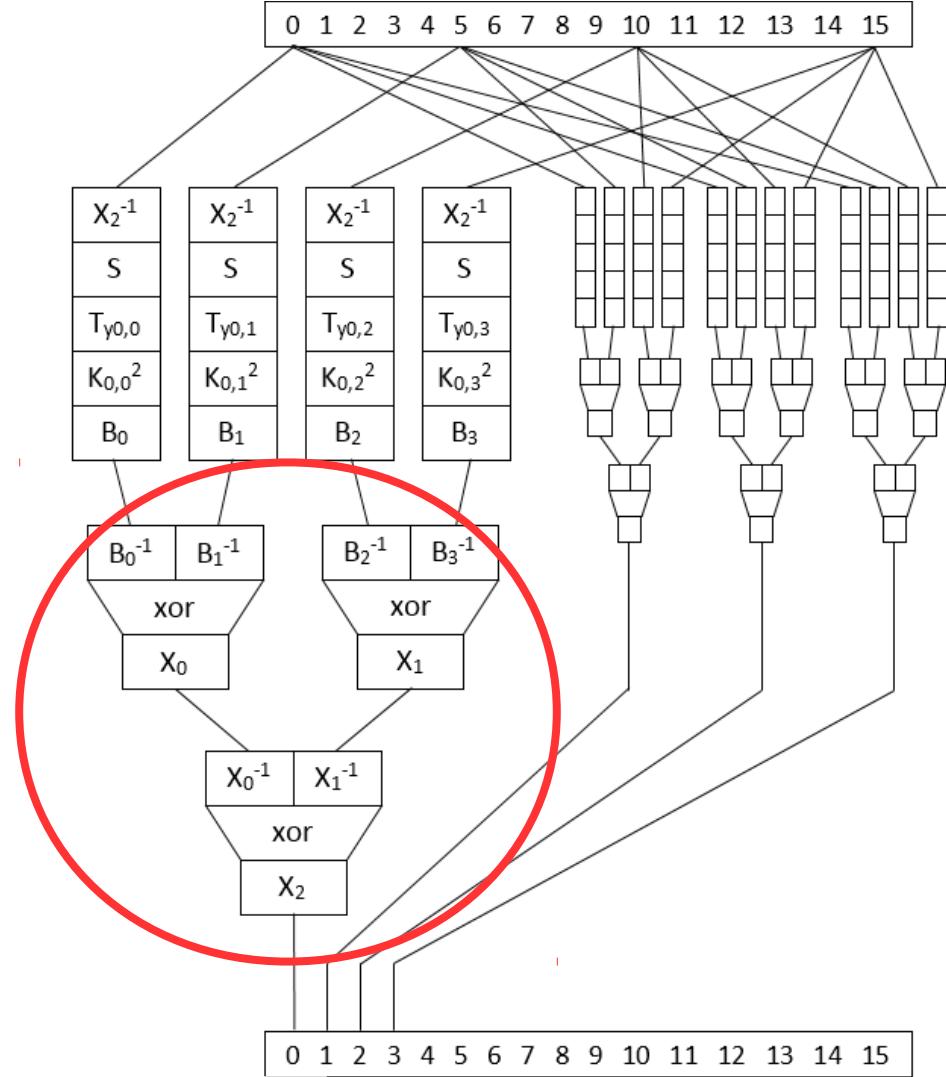
Can DCA fail?

Yes!

Wide intermediate
non-linear encodings (8x8)
blind the SBox non-linearity

But very large tables!

- Trend to reuse those tables
- reuse encodings
- other types of attack



cf my write-ups of
NoSuchCon 2013 and CHES 2015
http://wiki.yobi.be/wiki/CHES2015_Writeup

Other countermeasures?

Runtime randomness?

- Here, no trustworthy TRNG available

Runtime random delays?

- Trace instructions → realign

Building proper white-box technology is a delicate matter...

Forget about “perfect” security, but if cost of an attack is larger than the benefit for the attacker, you achieved your goal.

Oops, it seems our cheap attack raised the bar...

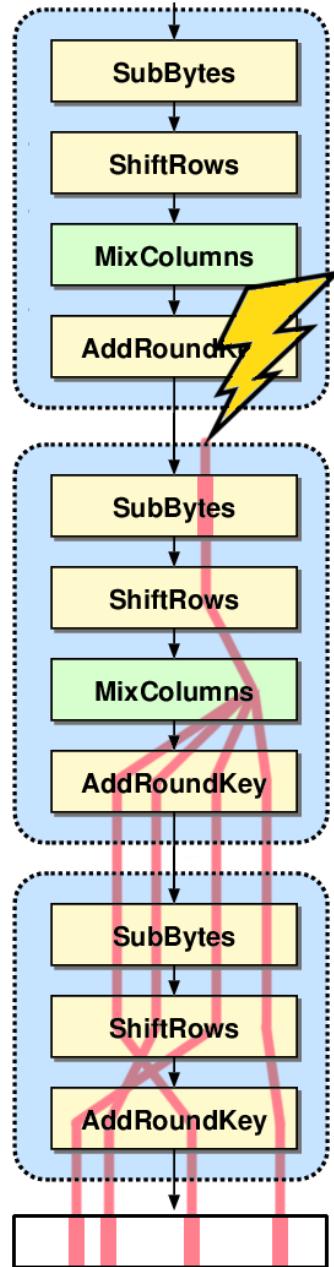
Other grey box attacks within reach: Higher order DPA, CPA, DFA,....



DFA, let's talk about it...

Differential Fault Analysis on AES (Dusart et al. 2003)

- Prerequisite: replay input, see clear output
- Injection: source ; bin statically ; bin dynamically
- Blind injection
- 8 “good” faults are enough (on AES-128 enc or dec)
- Analysis time: a few seconds



Take also care of code lifting, inverting f(),...

“Now this is not the end.
It's not even the beginning of the end.
But it is, perhaps, the end of the beginning.”





Side-Channel Marvels

<https://github.com/SideChannelMarvels>



Tracer

- TracerGrind
- TracerPIN
- TraceGraph



Deadpool

- White-boxes
- Attacks automation



Daredevil

- Side-channel analysis (CPA)



JeanGrey

- Fault analysis (DFA)



Side-Channel Marvels

<https://github.com/SideChannelMarvels>

Charles Hubain (Quarkslab)

Joppe Bos (NXP)

Sans oublier...

Michael Eder (TUM, Fraunhofer AISEC)

Paul Bottinelli (EPFL)

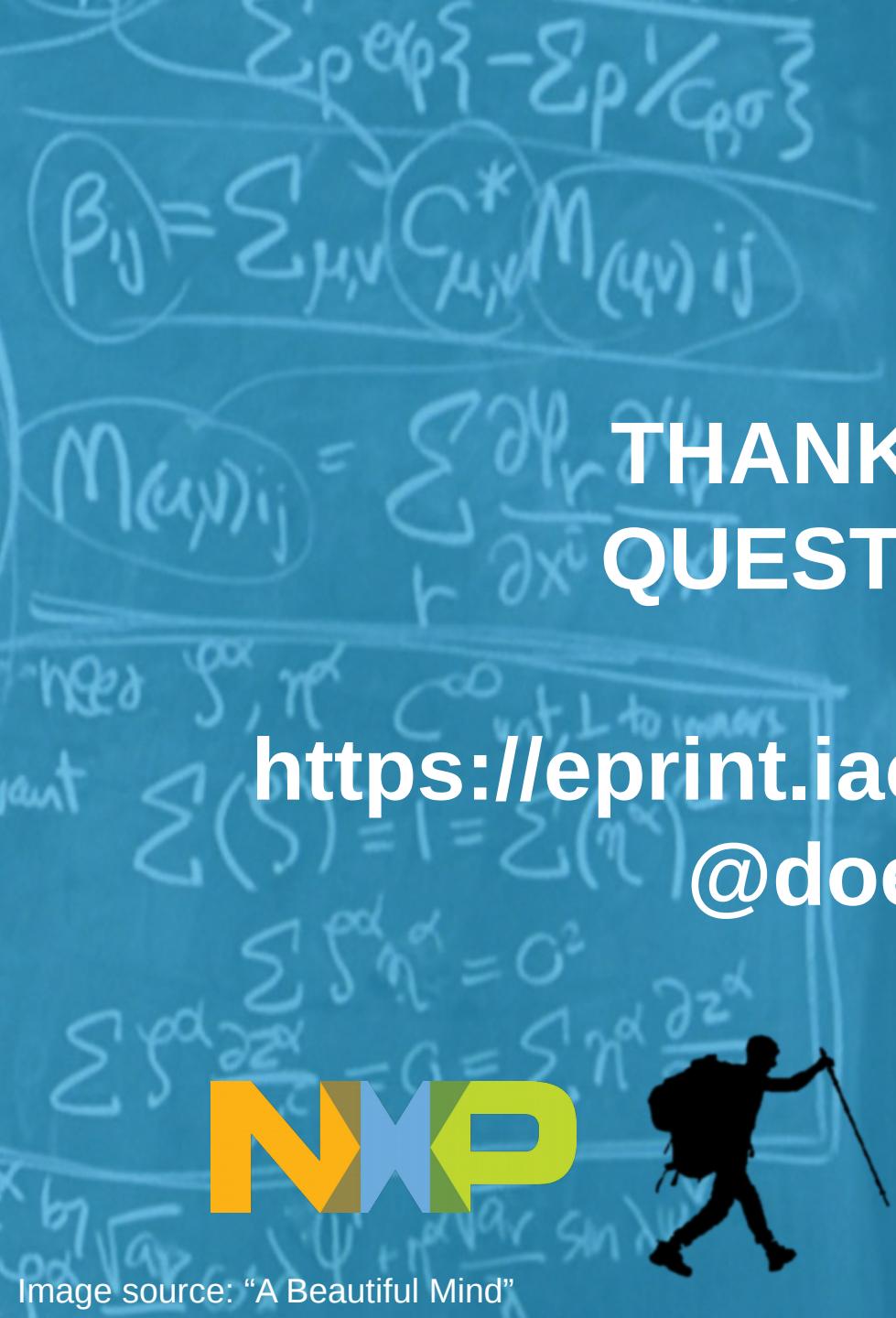
Philippe Teuwen (Quarkslab)

Van Huynh Le (U.Twente, NXP)

Wil Michiels (NXP, TU/e)



Orka
- images Docker



THANK YOU!
QUESTIONS?

<https://eprint.iacr.org/2015/753>

@doegox



Quarkslab