Hiding your White-Box Designs is Not Enough

TROOPERS16

Philippe Teuwen
16/03/2016

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

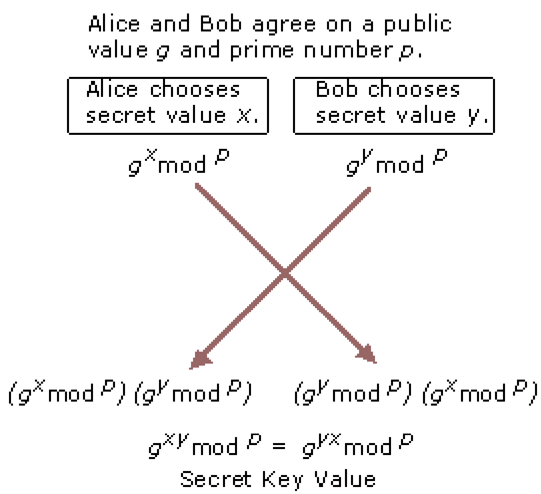
NXP

Outline

- Introduction to white-box cryptography
- Software execution traces
- Differential Computation Analysis

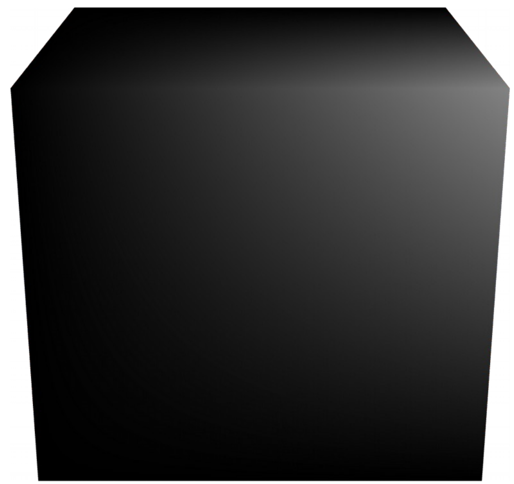
INTRODUCTION TO
WHITE-BOX CRYPTOGRAPHY

Black box model

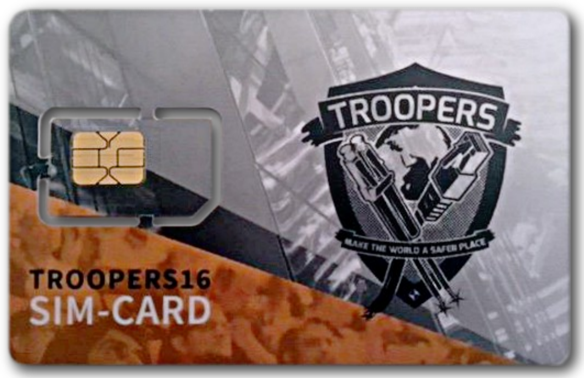


Black box model

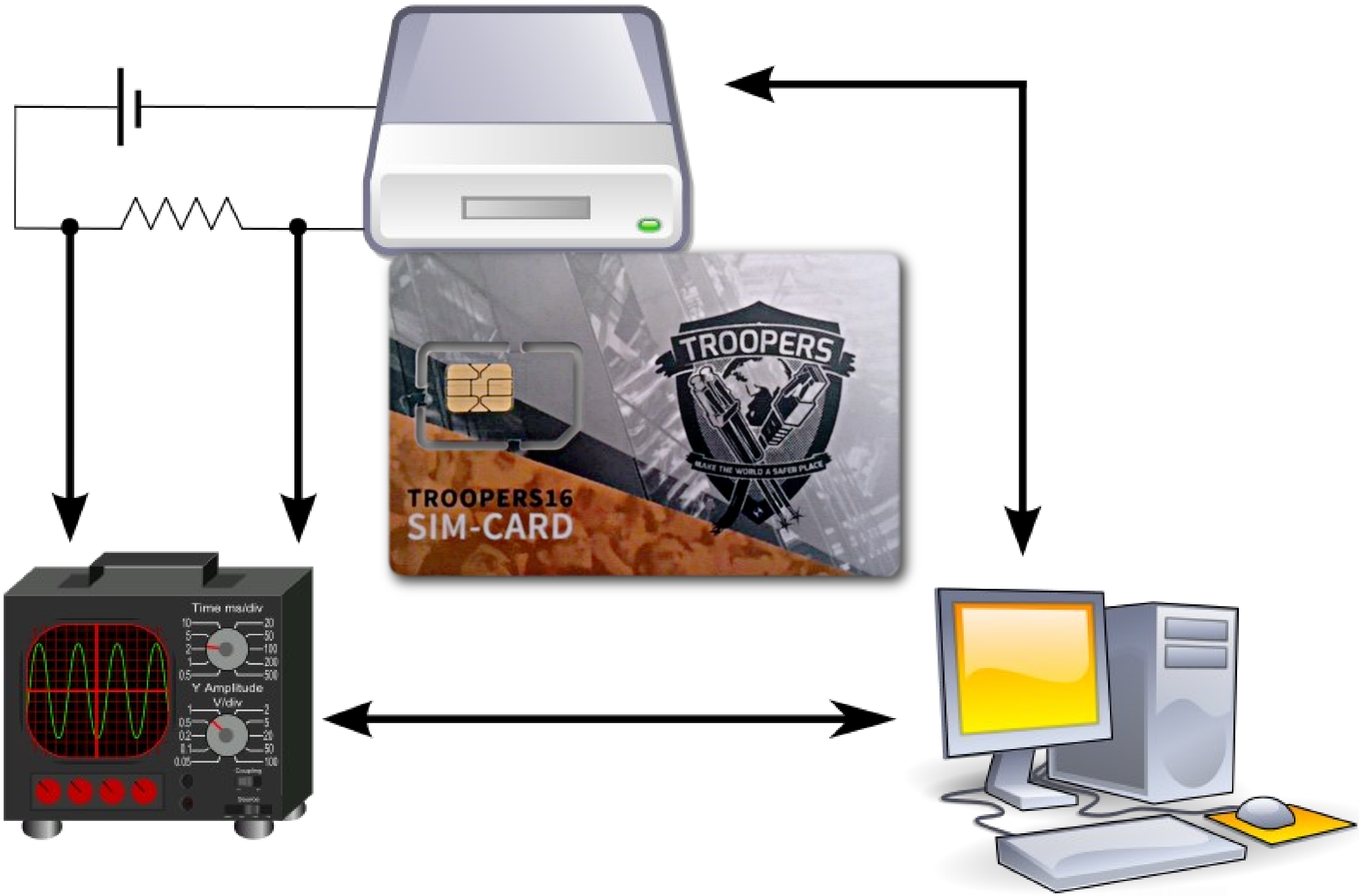
Black box model



Grey box model

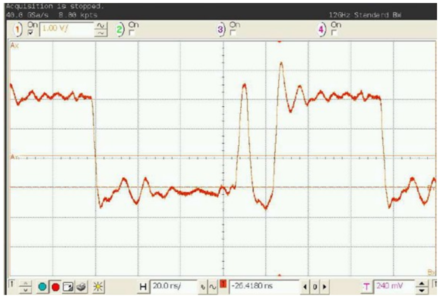


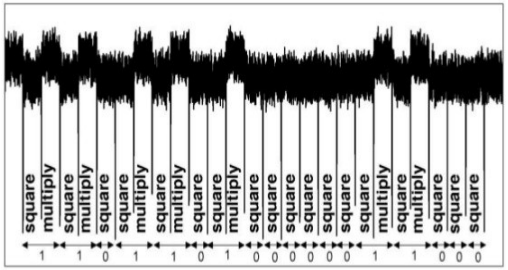
Grey box model

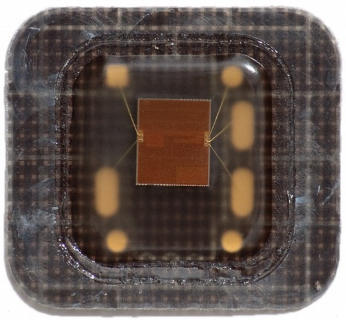


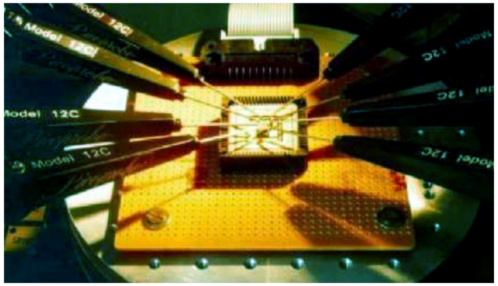
Artwork derived from CC BY-SA Mark Pellegrini work on Wikimedia Commons

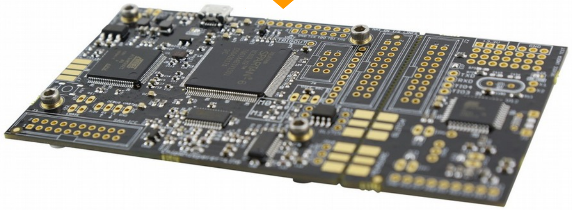
Grey box model





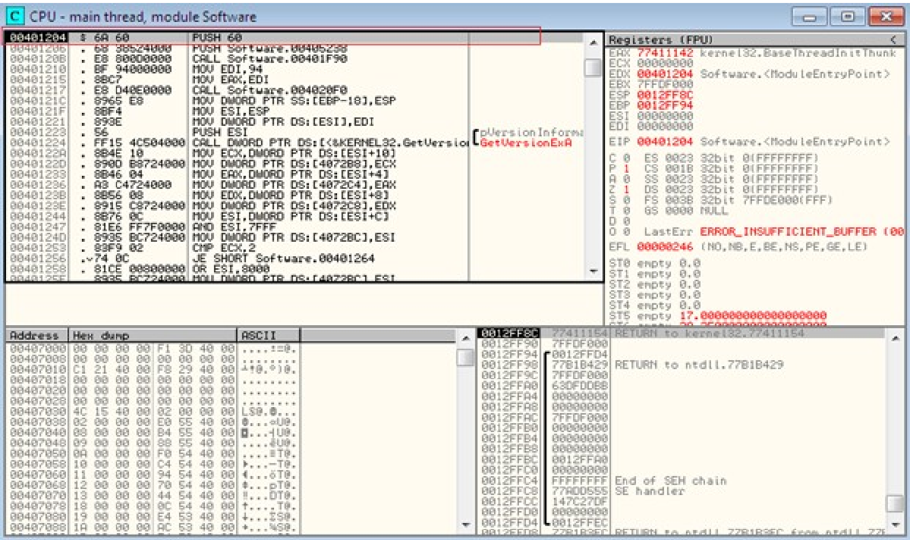


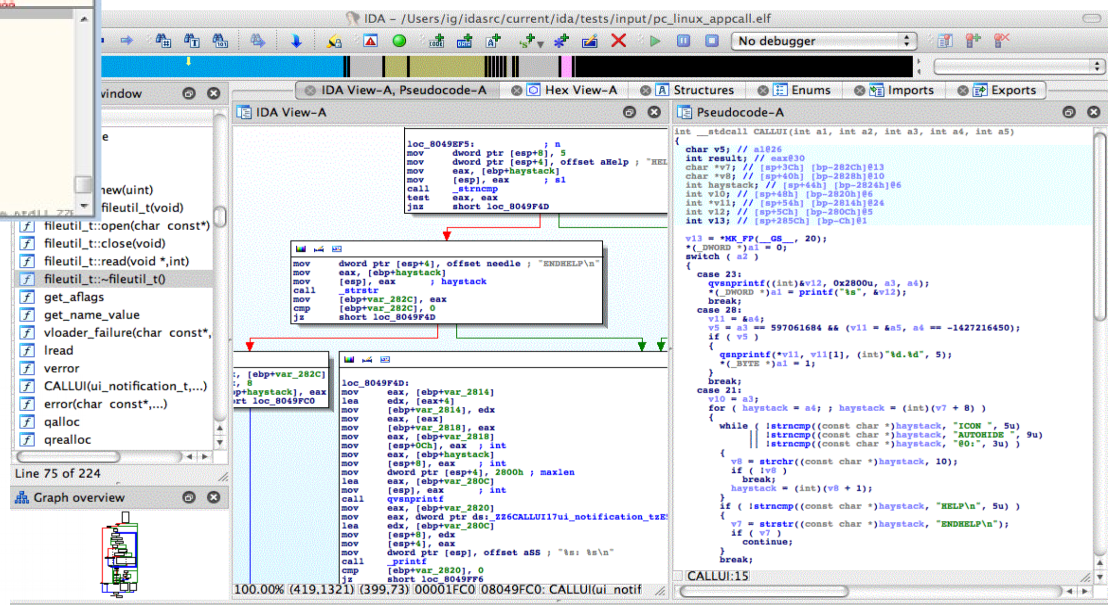




White box model

White box model





White box model

Sole line of defense:

Implementation

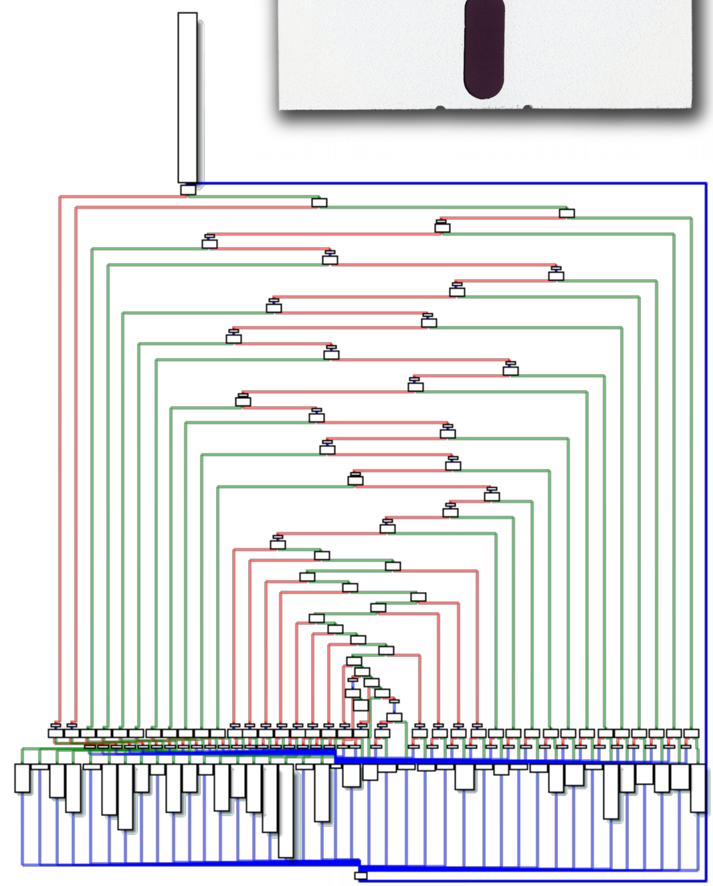
Usual countermeasures

Code obfuscation

Integrity checks

Anti-debug tricks

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



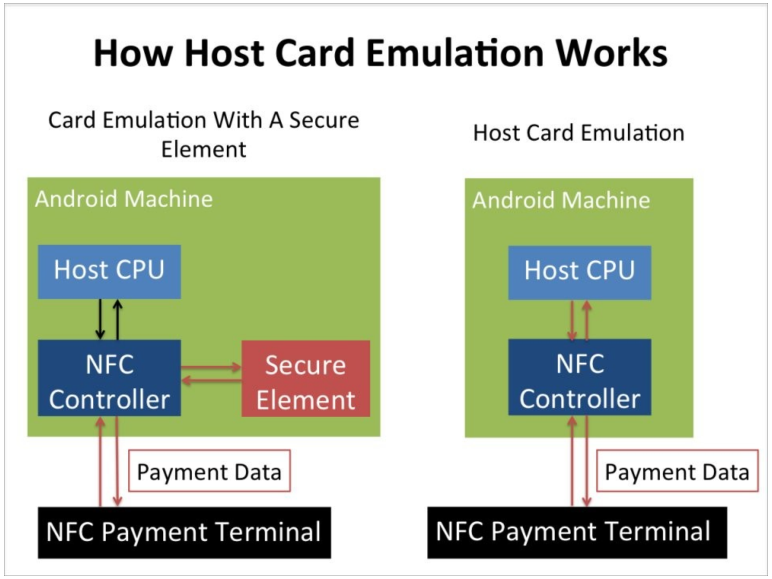
Illustrations: by @xoreaxeaxeax about M/o/Vfuscator and blog.quarkslab.com about OLLVM

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

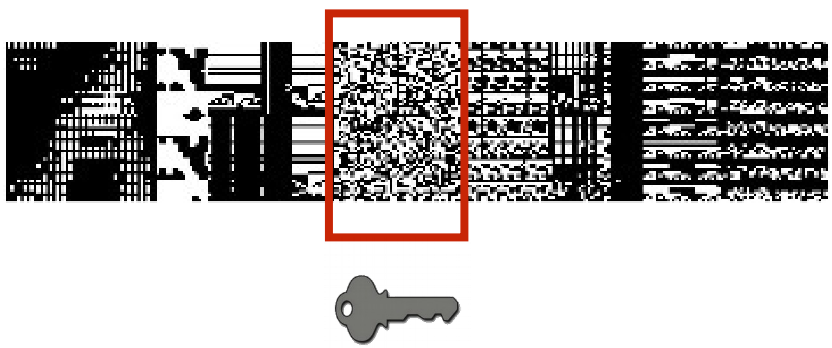
Cryptography under White-box model

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

- DRM schemes ↔ ~~criminals~~ users
- Mobile payment, HCE ↔ 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 x 10^{27} TB
- Practical WB AES:
  Network of smaller tables
  752kB
  Encoding on intermediate values

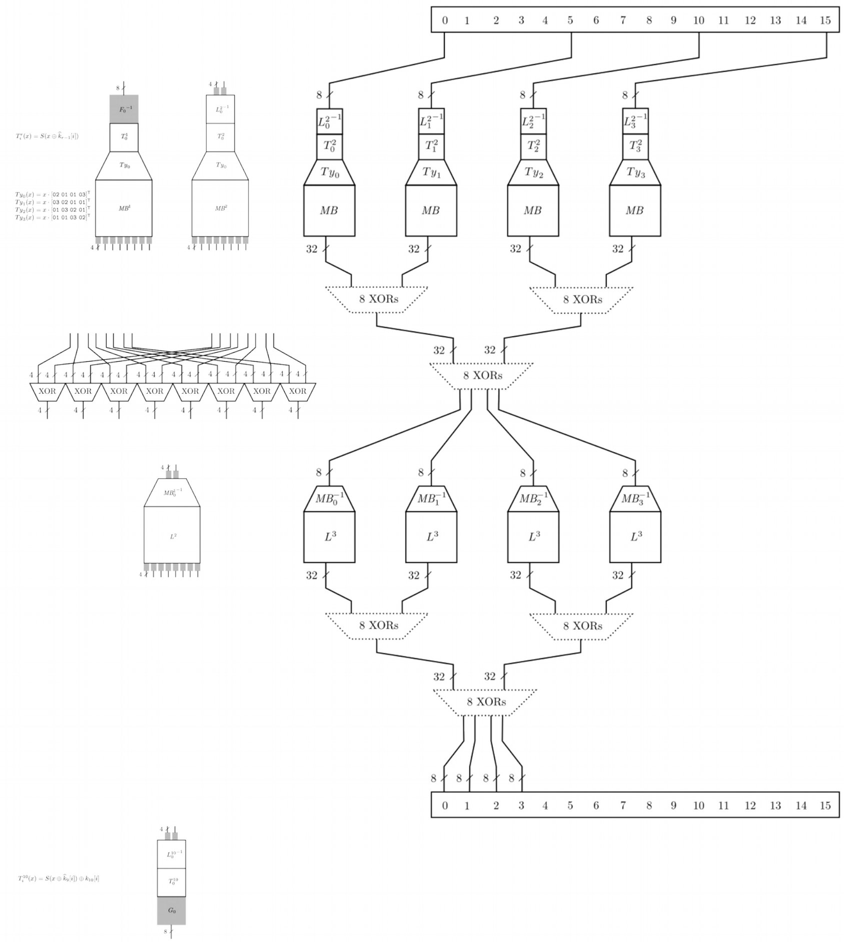


Illustration from “A Tutorial on White-box AES” by James A. Muir

White-box cryptography

History:

- Academic attacks → new designs → ...
- Today, <u>all</u> 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* $\overline{AES}^{(r,j)}_{enc} : (\mathbf{F}_2^8)^4 \to (\mathbf{F}_2^8)^4$ *for* $1 \leq r \leq 9$ *and* $0 \leq j \leq 3$*, called an encoded AES subround with byte permutations, is defined by*

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

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

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

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

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

$$\begin{cases} \mathtt{SR} \circ \bigoplus_{K'_{10}} \circ \{S_{10,i}\}|_{i=0,\dots,15} \circ \bigoplus_{K_9} \circ \Lambda_9'^{-1} & \text{for } R_{10} \ , \\ \mathtt{MC} \circ \mathtt{SR} \circ \Lambda_r'' \circ \{S_{r,i}\}|_{i=0,\dots,15} \circ \bigoplus_{K_{r-1}} \circ \Lambda_{r-1}'^{-1} & \text{for } R_r|_{2\leq r\leq 9} \ , \\ \mathtt{MC} \circ \mathtt{SR} \circ \Lambda_1'' \circ \{S_{1,i}\}|_{i=0,\dots,15} \circ \bigoplus_{K_0} & \text{for } R_1 \ , \end{cases} \tag{5}$$

The second step then implements the function $\tau^k_{r,i}$ 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:

$$\tau^k_{r,i}(x)(L^i_r, R'^i_r) = \left(\underbrace{\alpha^k_{r,i}(x|^4_{8\gamma_r(i)}, x|_{8\gamma_r(i)+4}, x|_{8\gamma_r(i)+5})}_{\textit{depends on } R_{r-1} \textit{ only}}\right) || \left(EP_i\left[PB\left(\underbrace{x|^4_{\gamma_r(0)} \,||\, x|^4_{\gamma_r(1)} \,||\, \dots \,||\, x|^4_{\gamma_r(11)}}_{\textit{depends on } R_{r-1} \textit{ only}}\right) \oplus \left(\underbrace{x|_{\mu_r(0)} \,||\, \dots \,||\, x|_{\mu_r(32)}}_{\textit{depends on } L_{r-1} \textit{ only}}\right)\right]\right)$$

$$\tau^k_r(x) = \tau^k_{r,0}(x) \,||\, \tau^k_{r,1}(x) \,||\, \dots \,||\, \tau^k_{r,11}(x)$$

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

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

$$\begin{aligned} \mu_r(x_0x_1\dots x_{47}, y_0\dots y_{47}) &= y_0\dots y_5 x_{\mu_r^{-1}(0)} x_{\mu_r^{-1}(1)} y_6 \dots y_{11} x_{\mu_r^{-1}(2)} x_{\mu_r^{-1}(3)} \dots y_{42} \dots y_{47} x_{\mu_r^{-1}(22)} x_{\mu_r^{-1}(23)} \dots x_{\mu_r^{-1}(47)} \\ \gamma_r(z_0z_1\dots z_{95}) &= z_{\gamma_r^{-1}(0)} \dots z_{(\gamma_r^{-1}(0)+5)} z_6 z_7 \dots z_{\gamma_r^{-1}(11)} \dots z_{(\gamma_r^{-1}(11)+5)} z_{94} z_{95} \end{aligned}$$

The obfuscated t-box is

$$T'^k_r(x) = (\phi_r \, T^k_r \, \psi^{-1}_{r-1})(x).$$

Hence the transformed function is:

$$E^k(x) = \left[(\lambda^{-1}\delta_n^{-1}\psi_n^{-1}) \cdot \left(\psi_n\delta_n\tau^k_n\phi_n^{-1}\right) \cdot \left(\phi_n T^k_n \psi^{-1}_{n-1}\right) \cdot \dots \cdot \left(\psi_1\delta_1\tau^k_1\phi_1^{-1}\right) \cdot \left(\phi_1 T^k_1 \psi_0^{-1}\right) \cdot (\psi_0\delta_0\beta\lambda)\right](x)$$

with

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

By setting

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

the resulting encryption operation is

$$E^k(x) = \left[\tau'^k_{n+1} \cdot \left(\tau'^k_n \cdot T'^k_n\right) \cdot \dots \cdot \left(\tau'^k_1 \cdot T'^k_1\right) \cdot \tau'^k_0\right](x)$$

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* $\overline{AES}_{enc}^{(r,j)} : (\mathbf{F}_2^8)^4 \to (\mathbf{F}_2^8)^4$ *for* $1 \le r \le 9$ *and* $0 \le j \le 3$*, 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*

$$\Pi_2^{(r,j)} \circ \overline{AES}^{(r,\pi^{(r)}(j))} \circ \Pi_1^{(r,j)} = [illegible] \circ (S, S, S, S) \circ \oplus_{[\bar{k}_i^{(r,j)}]_{0\le [illegible]}}$$

$$[\bar{k}_i^{(r,j)}]_{0\le i\le 3} = [illegible]_{r,\pi^{(r)}(j))}]_{0\le i\le 3})$$

$$\mathtt{MC}^{(r,j)} = [illegible] \circ \Pi_1^{(r,j)} \ .$$

As a re[illegible] algorithm mentioned ab[illegible]ite-box adversa[illegible] has black-box access [illegible] the following structures of each round $R_r|_{r=1,\dots,10}$:

$$\begin{cases} \mathtt{SR} \circ \bigoplus_{K'_{10}} \circ \{S_{10,i}\}|_{i=0,\dots,15} \circ \bigoplus_{K_9} \circ \Lambda_9'^{-1} & \text{for } R_{10} \ , \\ \mathtt{MC} \circ \mathtt{SR} \circ \Lambda_r'' \circ \{S_{r,i}\}|_{i=0,\dots,15} \circ \bigoplus_{K_{r-1}} \circ \Lambda_{r-1}'^{-1} & \text{for } R_r|_{2\le r\le 9} \ , \\ \mathtt{MC} \circ \mathtt{SR} \circ \Lambda_1'' \circ \{S_{1,i}\}|_{i=0,\dots,15} \circ \bigoplus_{K_0} & \text{for } R_1 \ , \end{cases} \quad (5)$$

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:

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

$$\tau_r^k(x) = \tau_{r,0}^k(x) \| \tau_{r,1}^k(x) \| \dots \| \tau_{r,11}^k(x)$$

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

$$[illegible] R) = \gamma_r(\mu_r((L|0^{24}), [illegible]$$

$$\mu_r([illegible]) = y_0 \dots y_5 x_{\mu_r^{-1}(0)} x_{\mu_r^{-1}} [illegible] x_{\mu_r^{-1}(2)} x_{\mu_r^{-1}(3)} \dots y_{42} \dots [illegible] x_{\mu_r^{-1}(47)}$$

$$[illegible] \dots z_{95}) = z_{\gamma_r^{-1}(0)} \dots z_{(\gamma_r^{-1}} [illegible]^{-1}(11)} \dots z_{(\gamma_r^{-1}(11)+5)} z_{94} z_{95}$$

The o[illegible] is

$$T_r'^k(x) [illegible] \phi_r [illegible]_{-1})(x).$$

[illegible] the tran[illegible]med [illegible]ction is:

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

with

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

By setting

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

the resulting encryption operation is

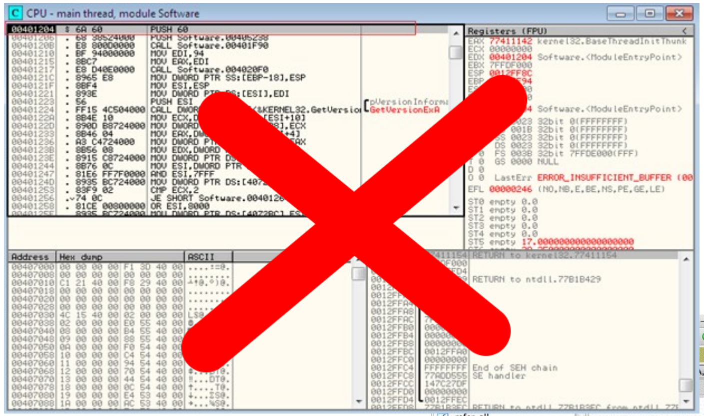
$$E^k(x) = \left[\tau_{n+1}'^k \cdot \left(\tau_n'^k \cdot T_n'^k\right) \cdot \dots \cdot \left(\tau_1'^k \cdot T_1'^k\right) \cdot \tau_0'^k \right](x)$$

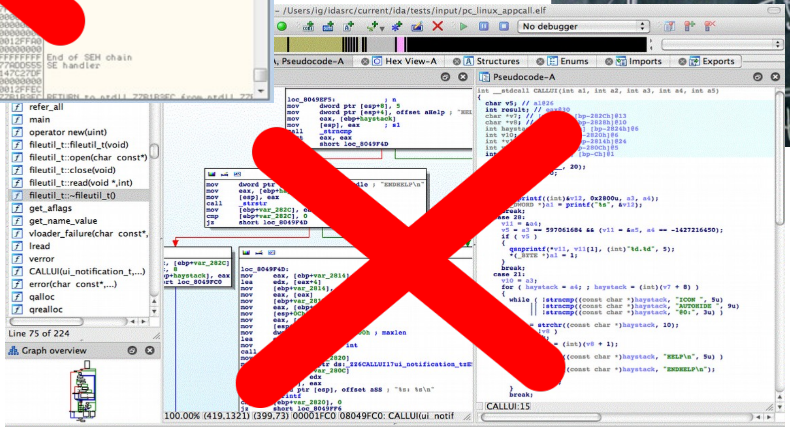
看起來像天書

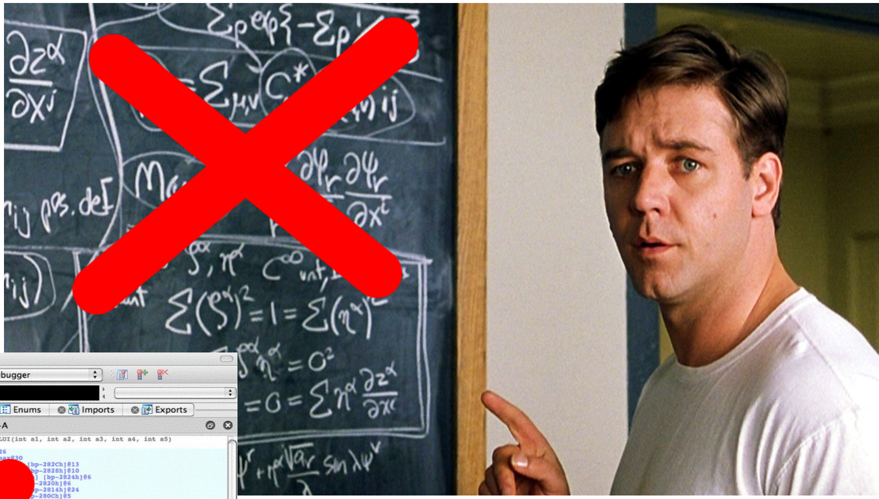
Our goal

Recover white-box keys

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







SOFTWARE EXECUTION TRACES

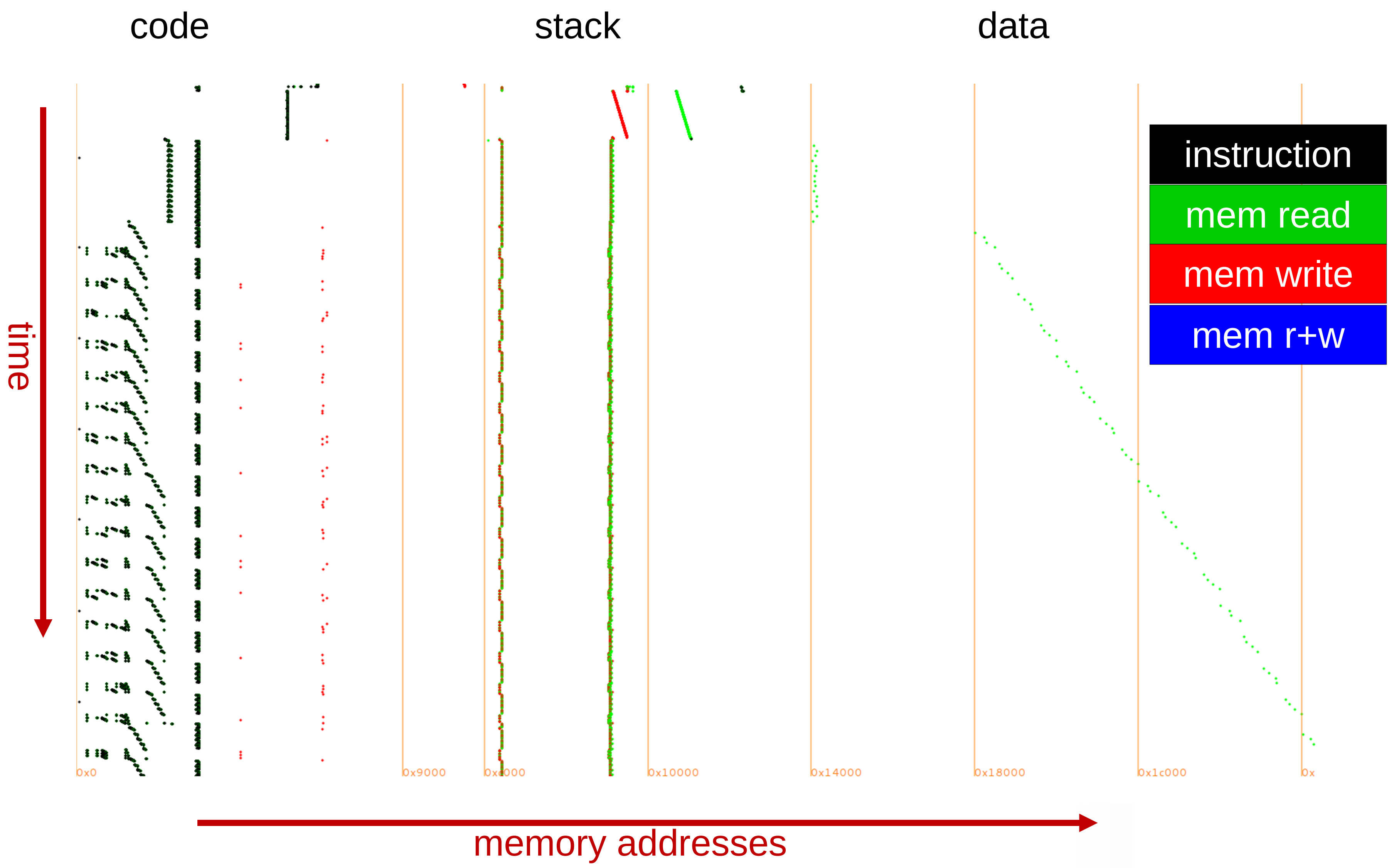
Tracing binaries

Record all instructions and memory accesses

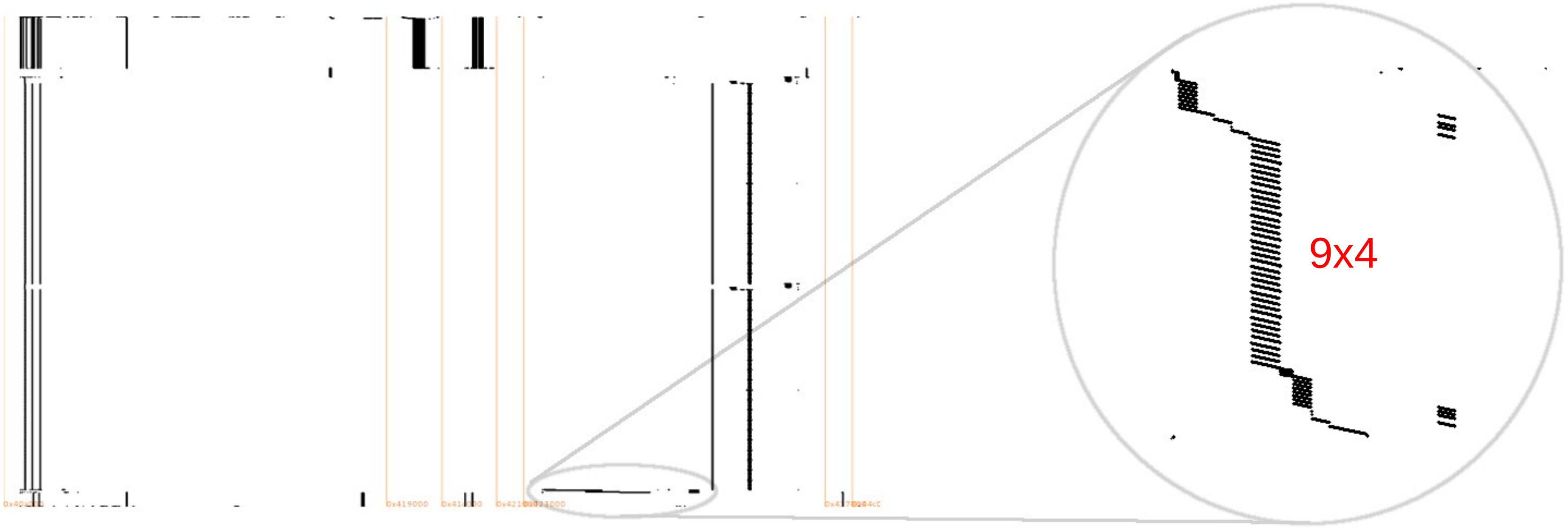
Examples:

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

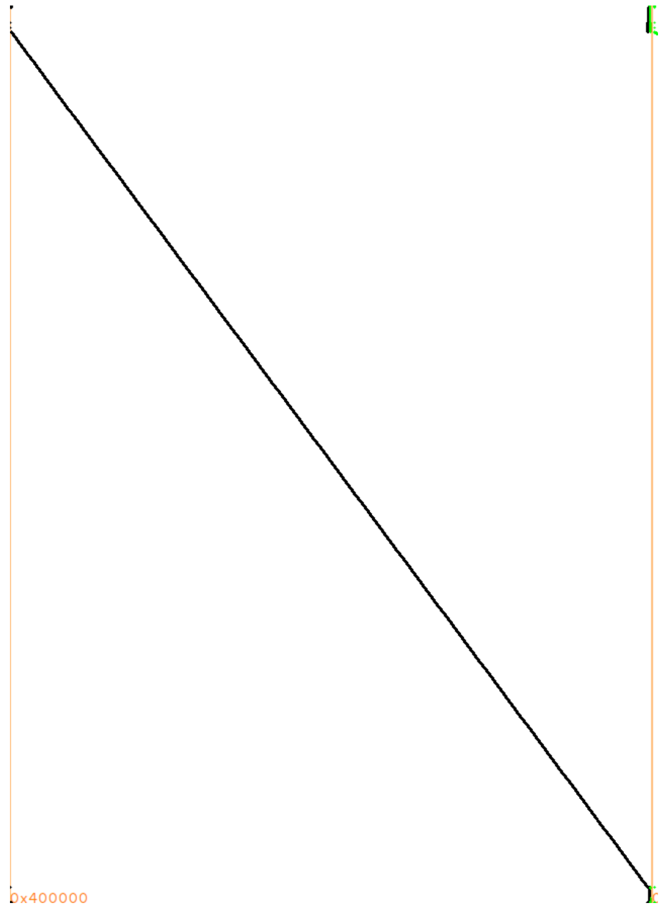
Trace convention: Quarkslab's pTra waterfall



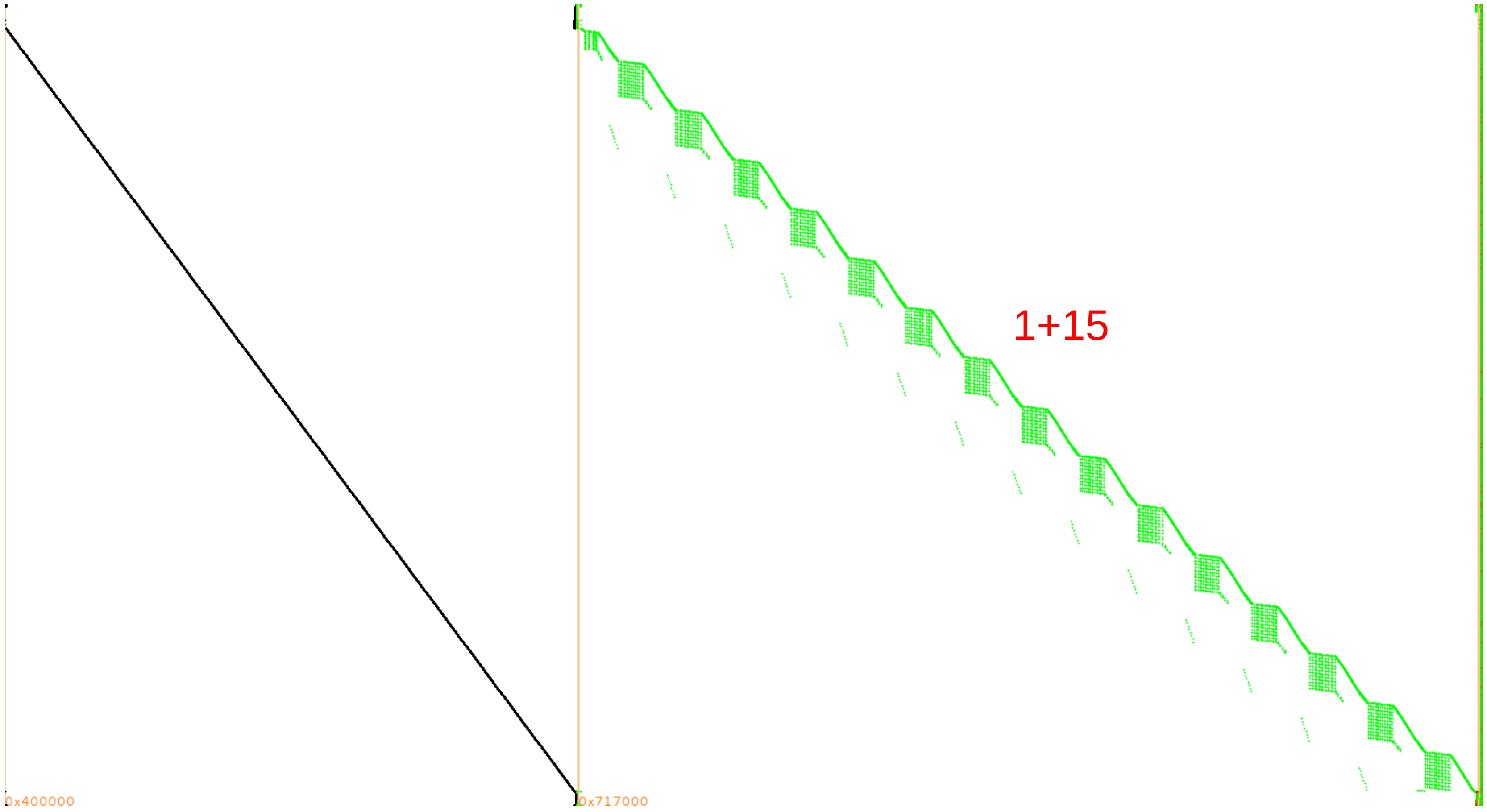
Visual crypto identification: code



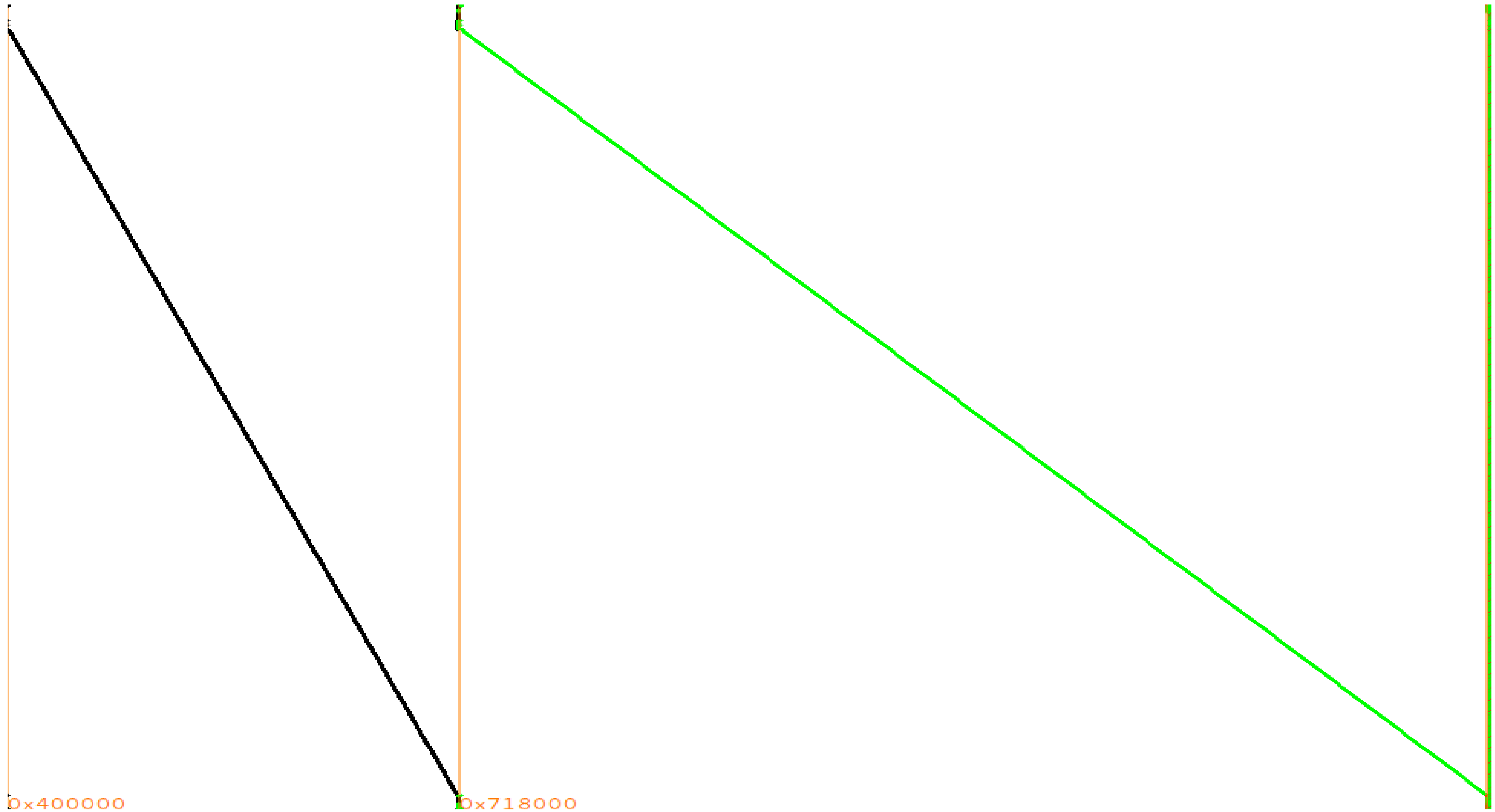
Visual crypto identification: code?



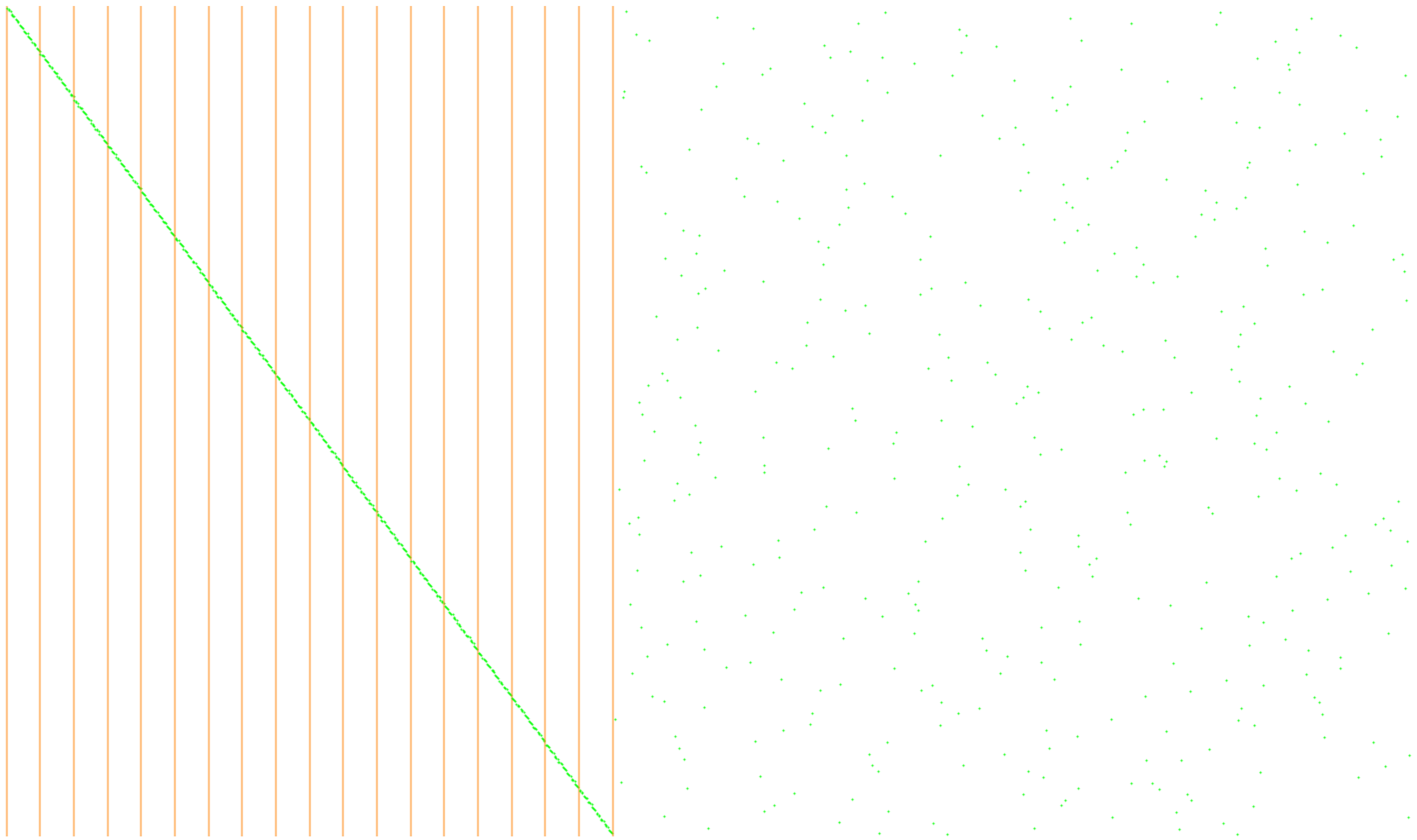
Visual crypto identification: code? data!



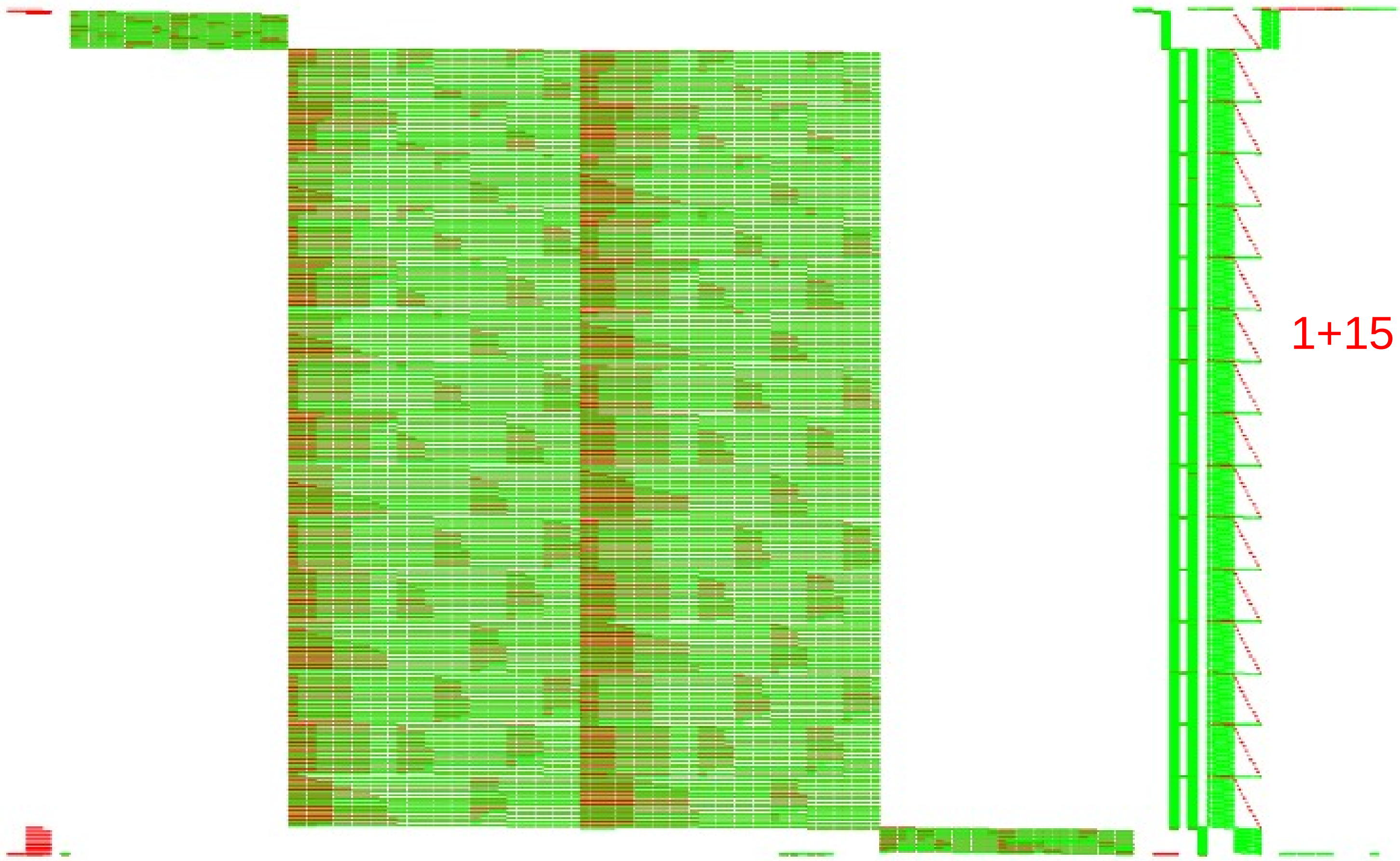
Visual crypto identification: code? data?



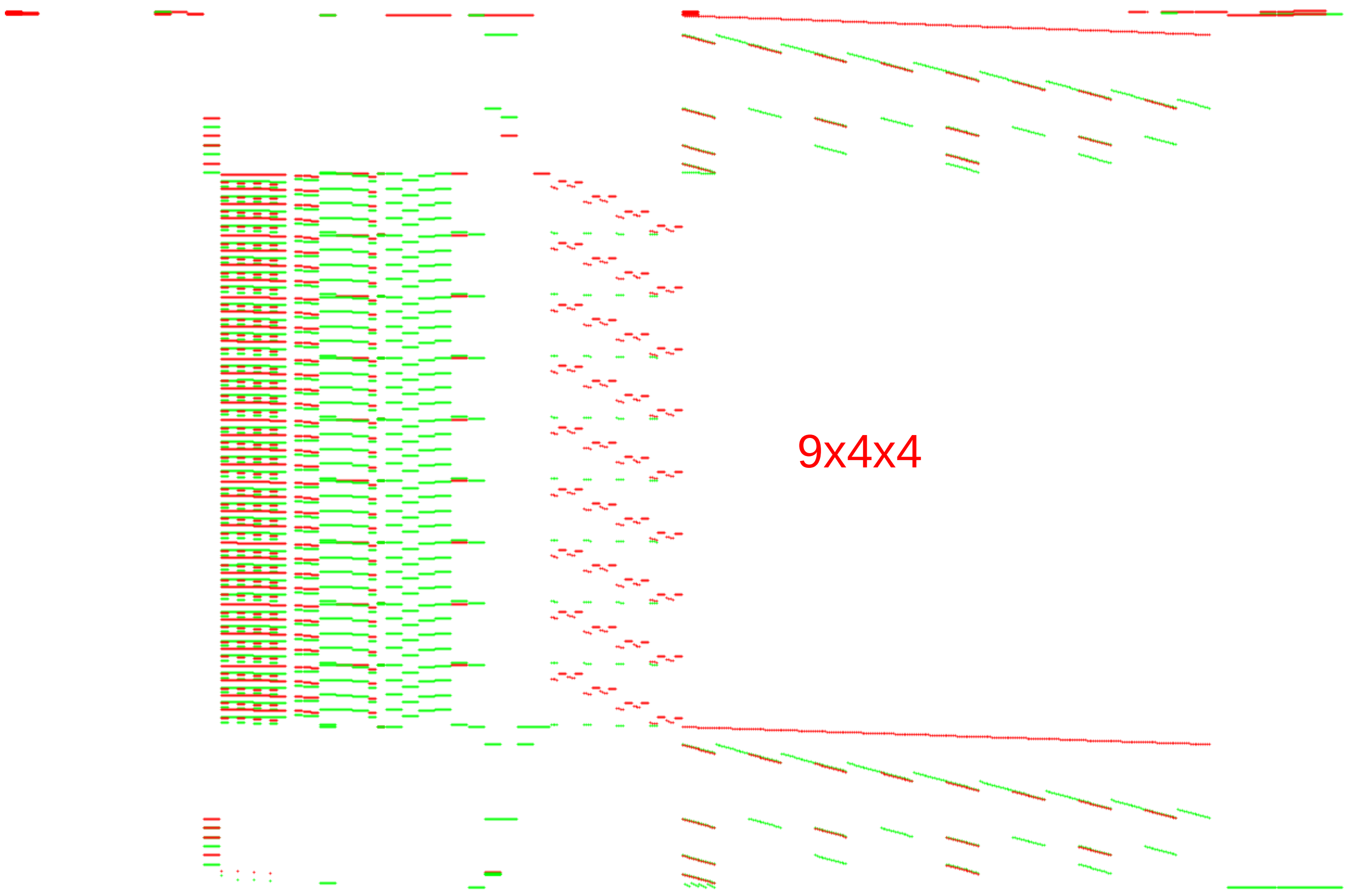
Visual crypto identification: data?



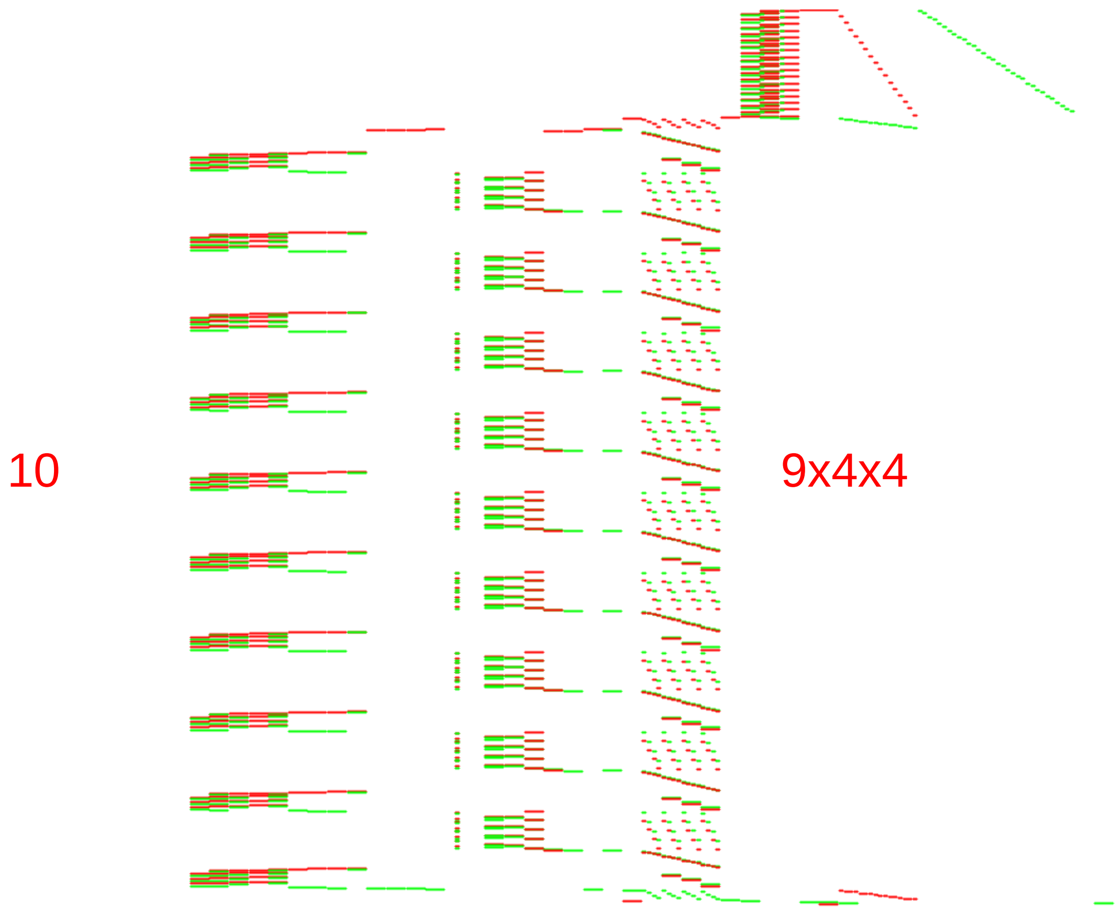
Visual crypto identification: stack!



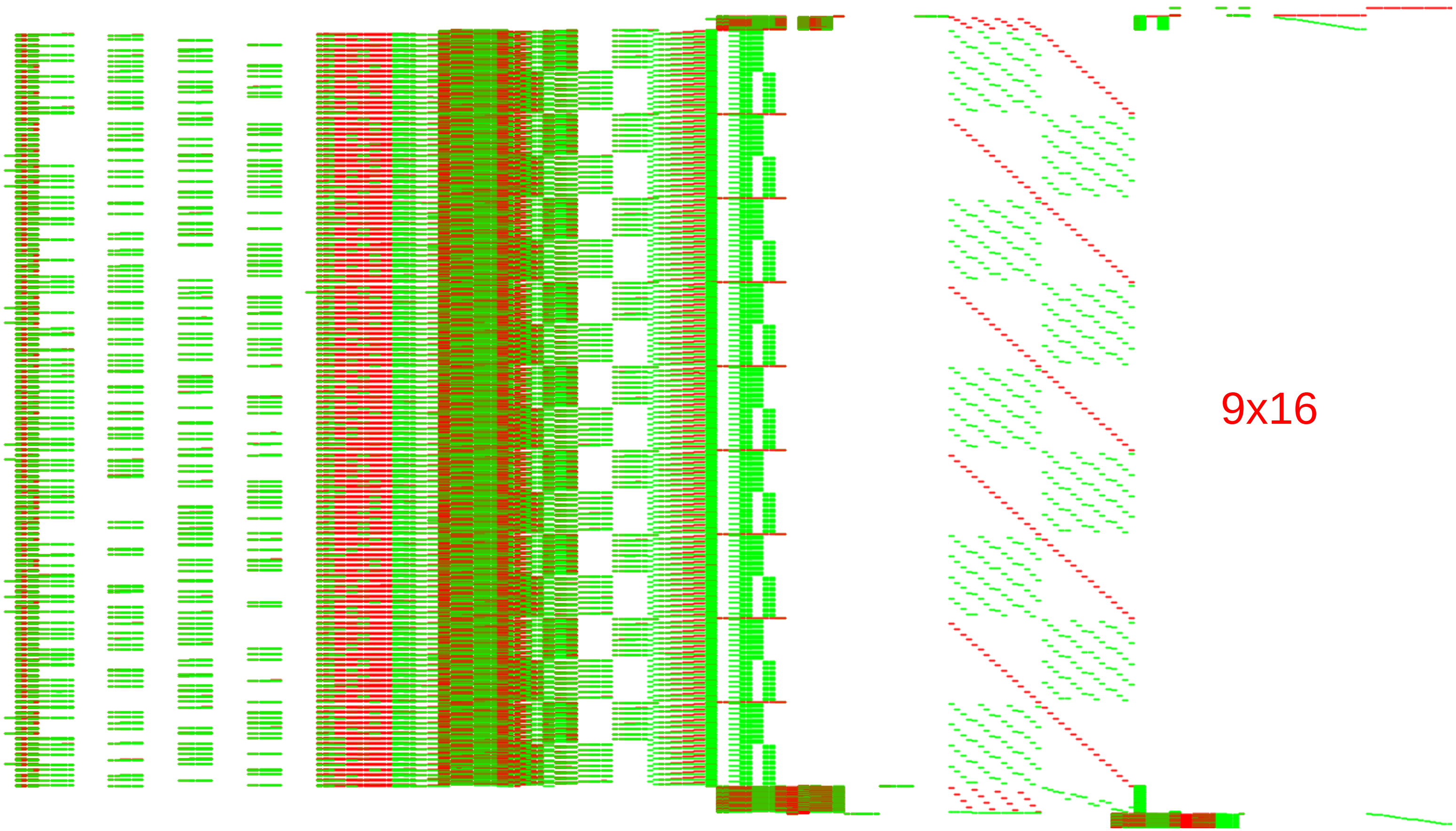
Visual crypto identification: stack!



Visual crypto identification: stack!



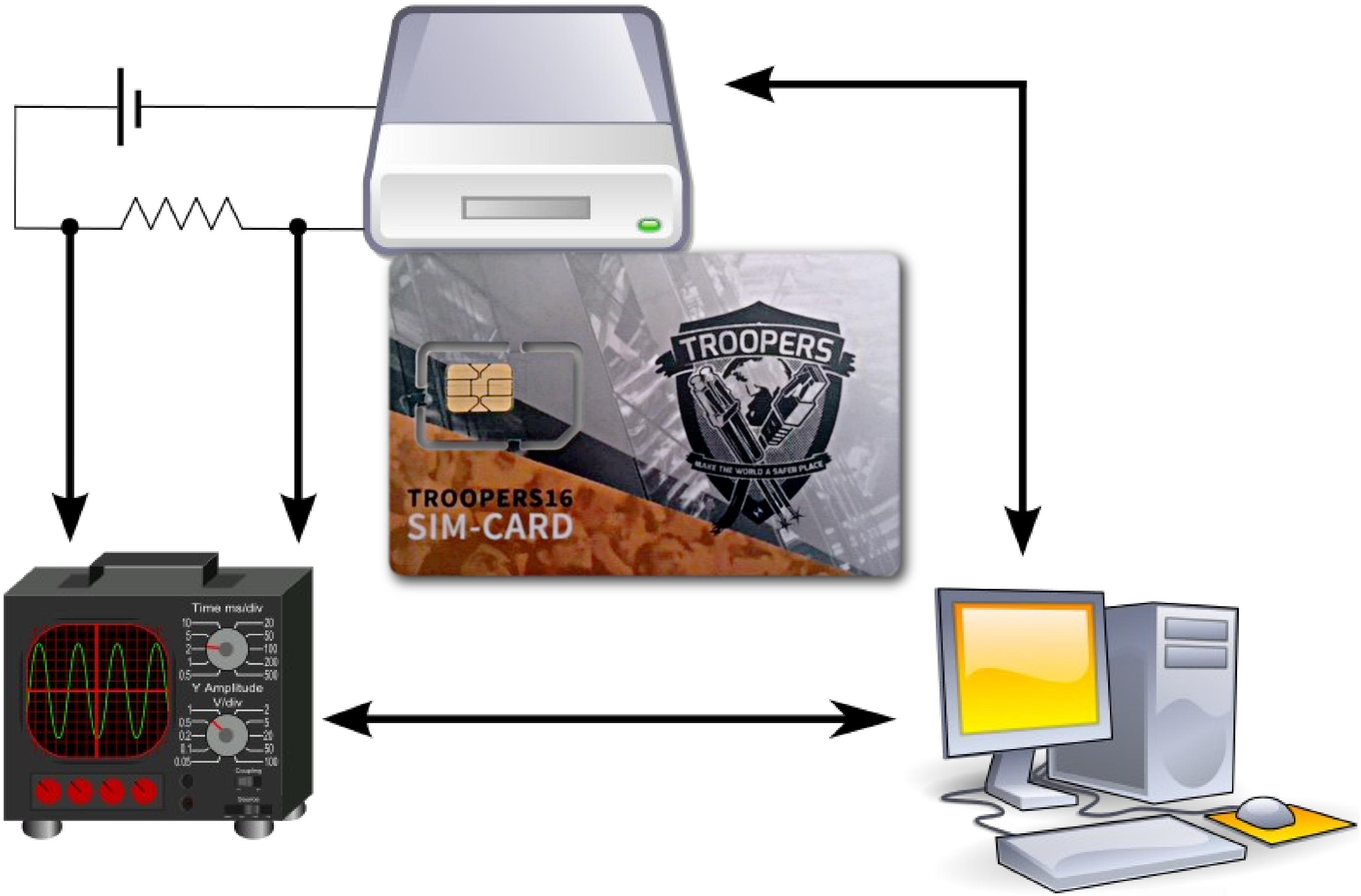
Visual crypto identification: stack!



Where is my key?

DIFFERENTIAL COMPUTATION ANALYSIS

Remember?



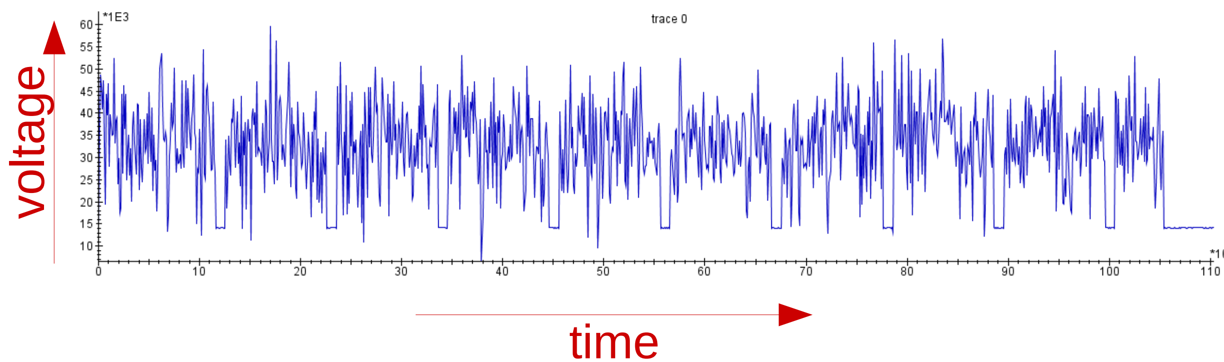
TROOPERS
MAKE THE WORLD A SAFER PLACE
TROOPERS16
SIM-CARD
Time ms/div
Y Amplitude
V/div

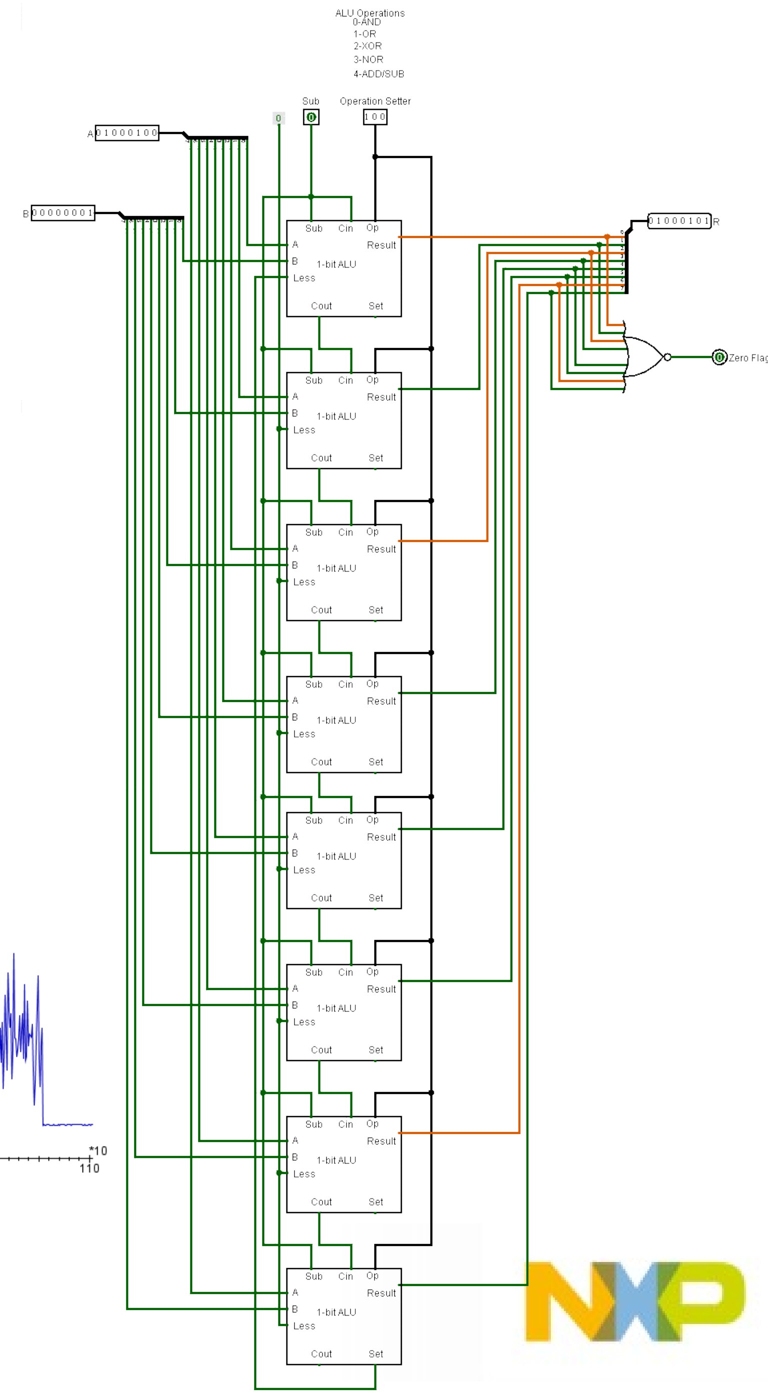
All started with Differential Power Analysis

by P. Kocher et al. (1998)

- Probable correlations:
  power consumption vs.
  Hamming weight of internal values

- Record many traces
  while providing different inputs

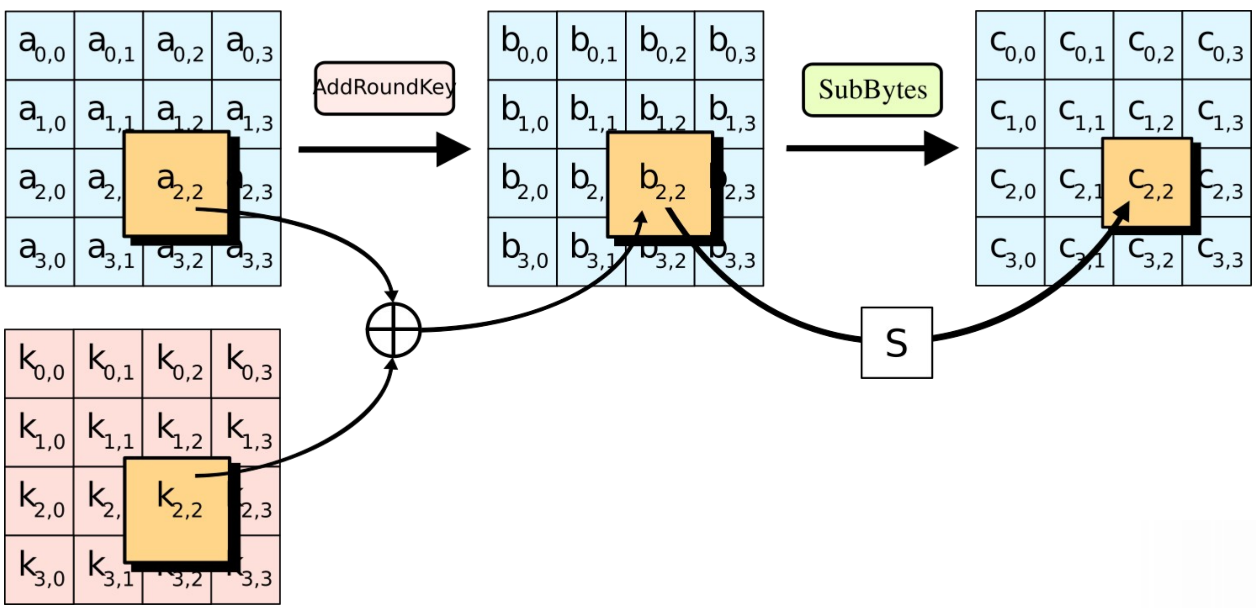




Differential Power Analysis

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

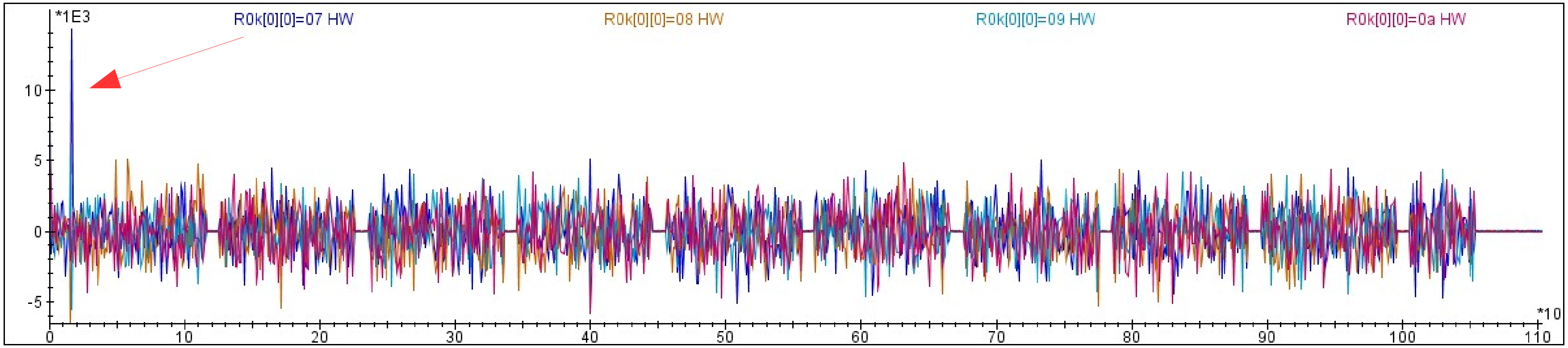
E.g. for AES:



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

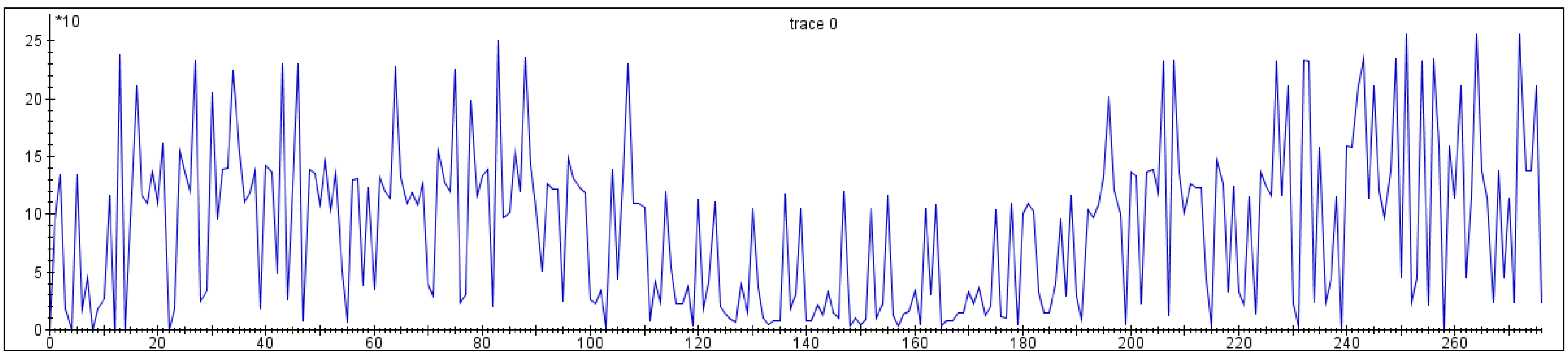
Differential Computation Analysis

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

Software execution traces → “power traces”

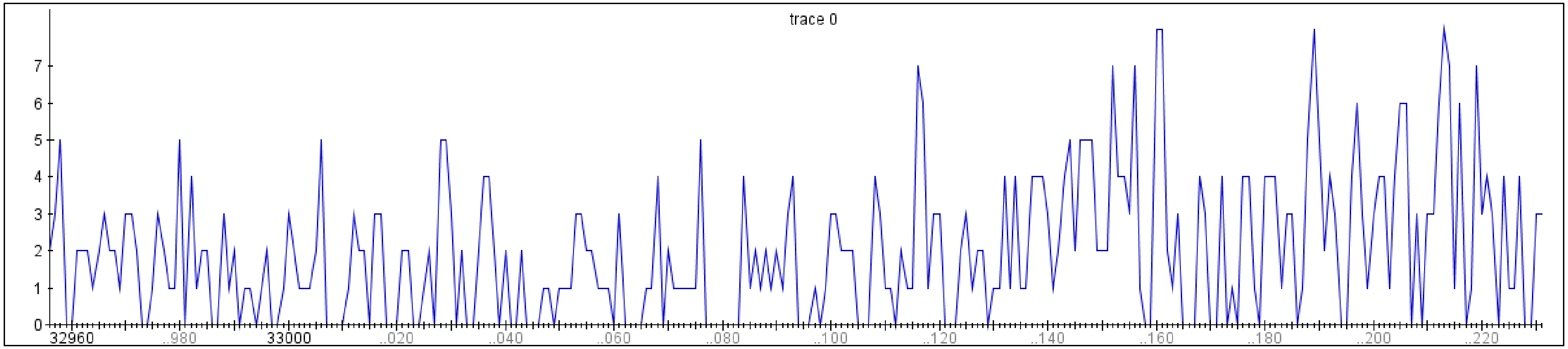
Memory accesses / data / stack writes / …

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



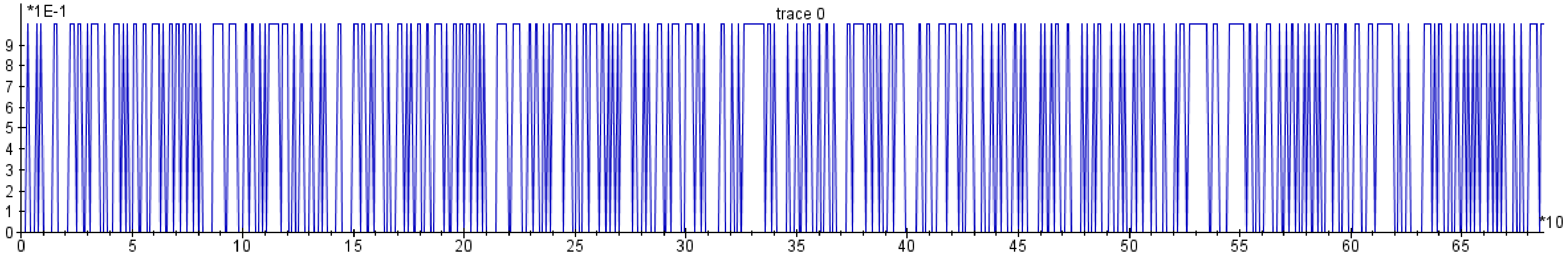
Differential Computation Analysis

→ Build Hamming weight traces?

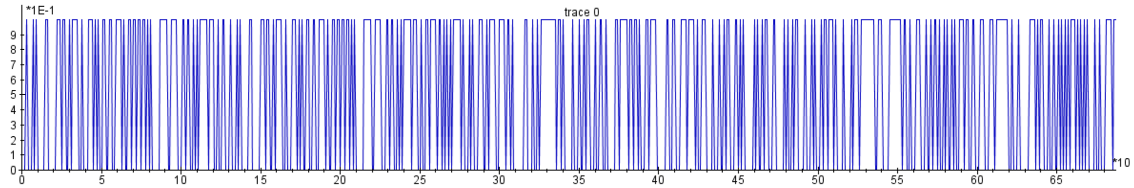


Differential Computation Analysis

→ Serialize bytes in a succession of bits



Differential Computation Analysis



Looks weird but works great!

As if:

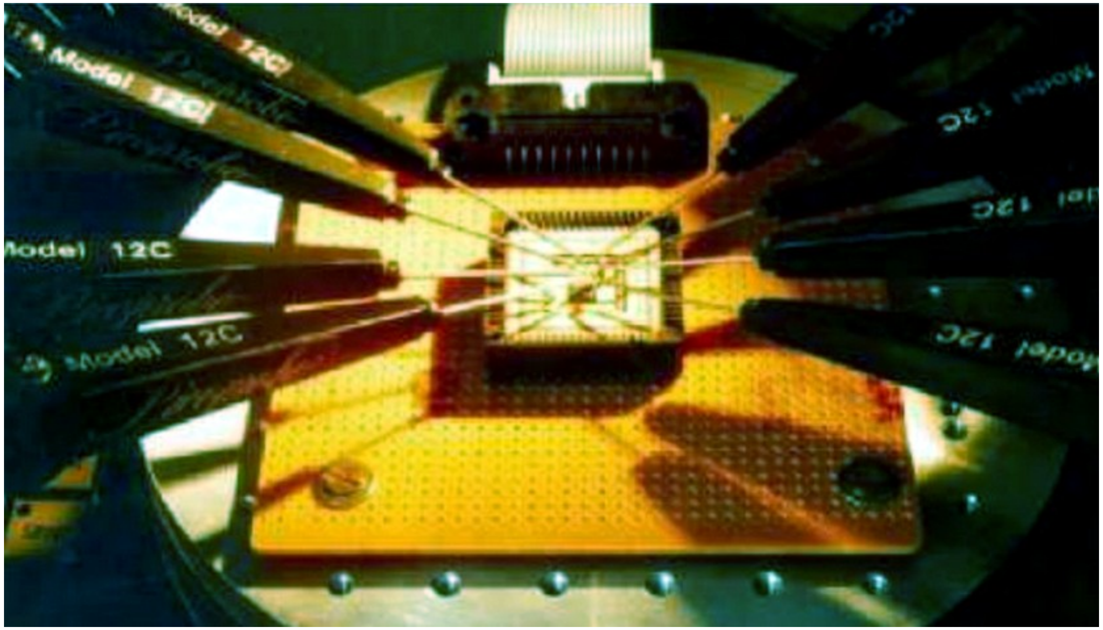


Image source: Brightsight

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

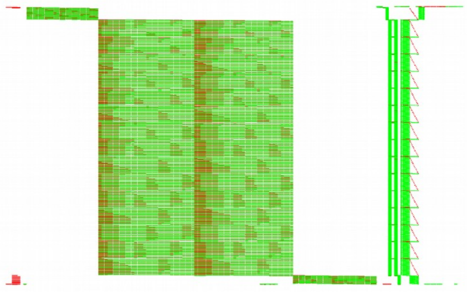
plaintexts and/or ciphertexts

- 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



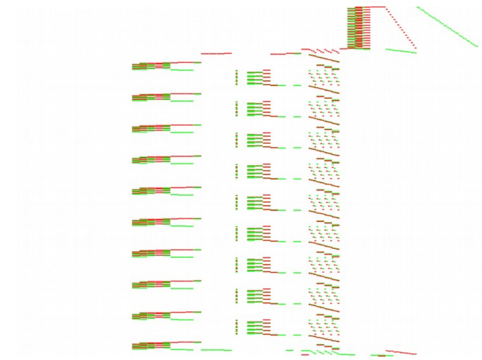
by Brecht Wyseur, 2007

DES implementation based on Chow “plus some personal improvements”

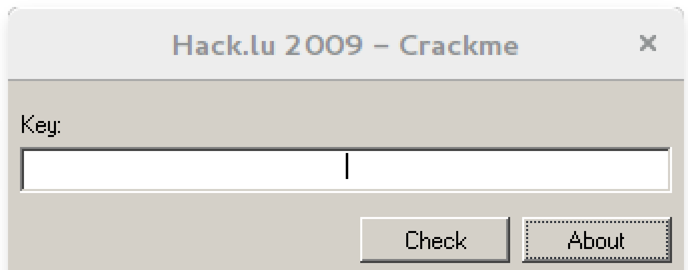
Downloading Linux binary…

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

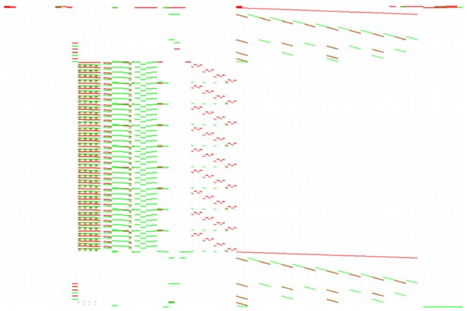
Python + PIN = Boom

→ 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

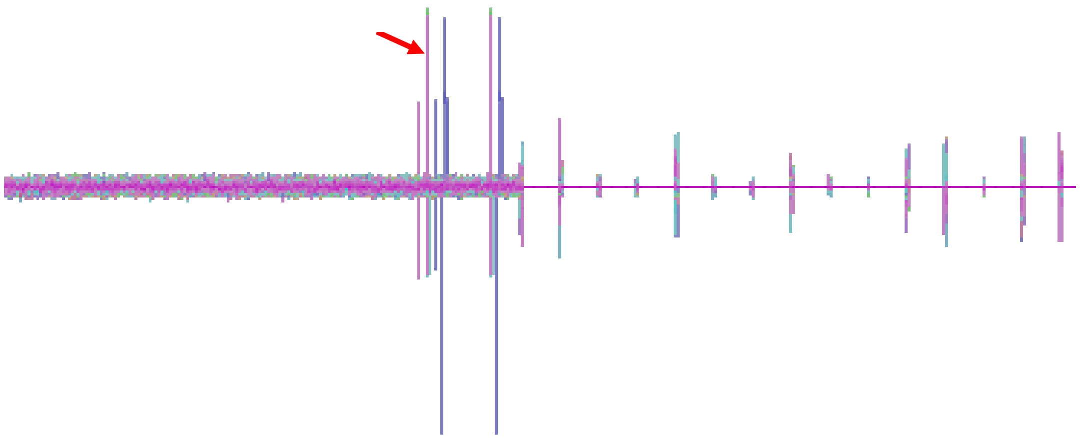
Some proprietary white-boxes

DES & AES

Broken in 200 to 2500 traces

Back to White-Box design

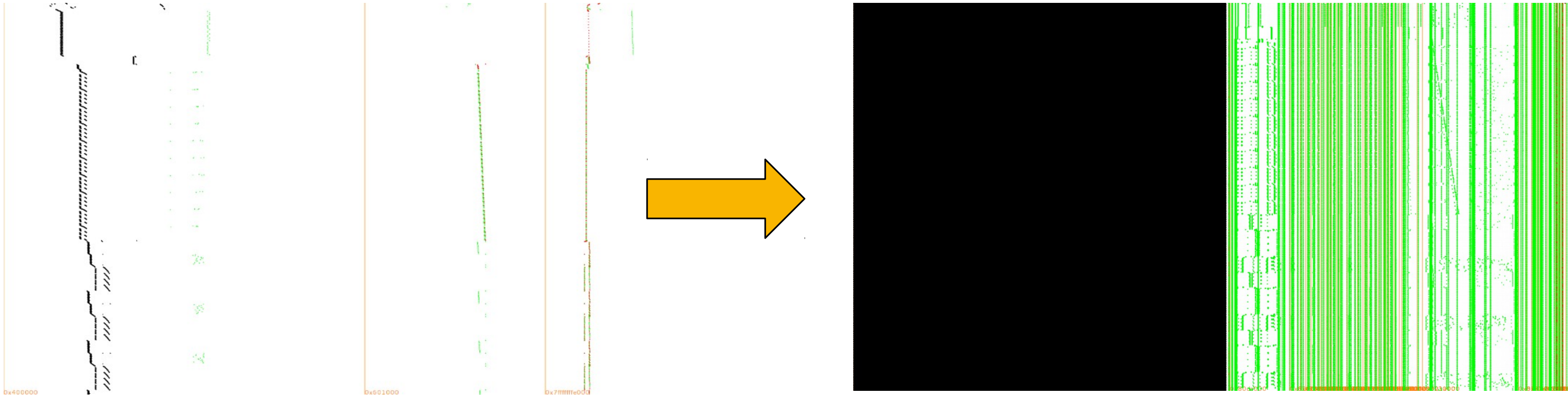
Known key analysis



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: M/o/Vfuscator2

Applied on a standard AES implementation

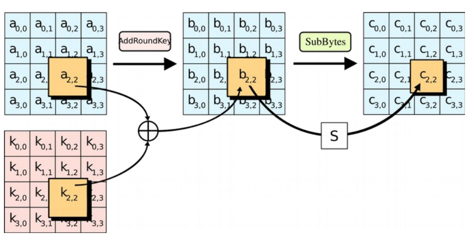


M/o/Vfuscator2 on AES

Auto-correlation reveals structure:

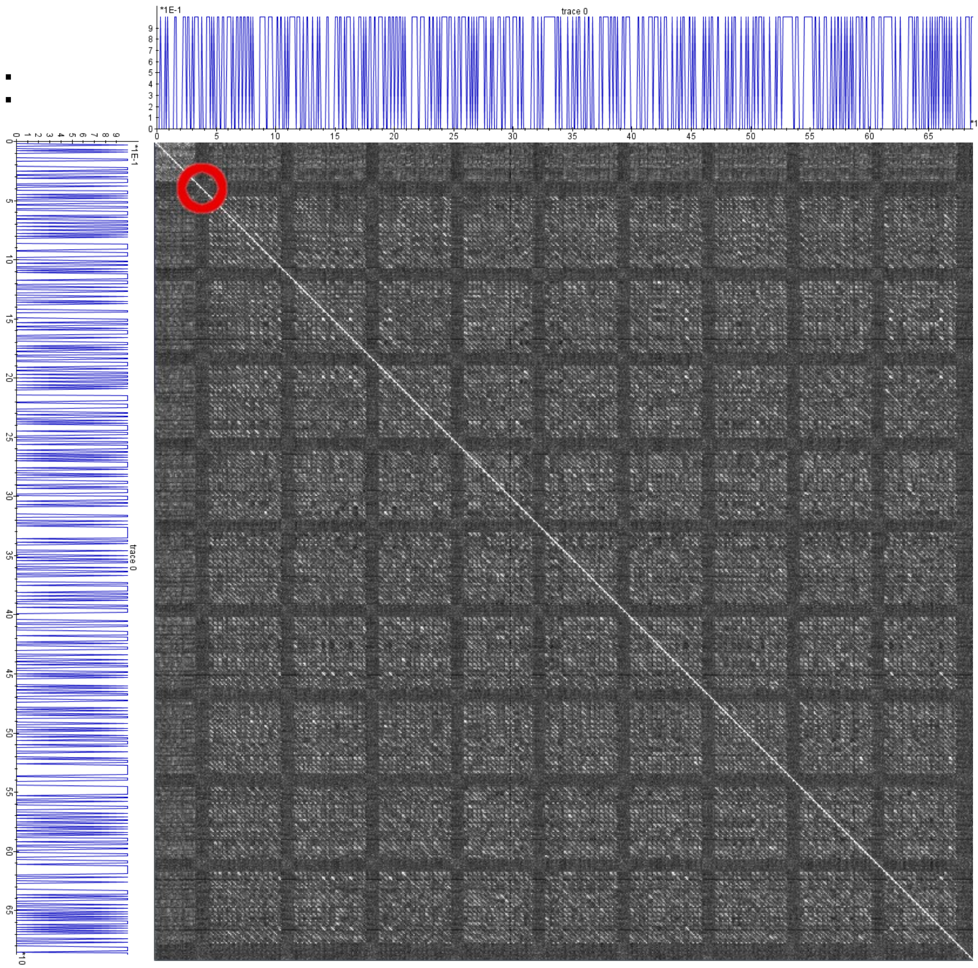
Huge traces, compressed by looking at standard deviation
4Mb -> 6.6kb

First round Sbox output



20 – 30 traces

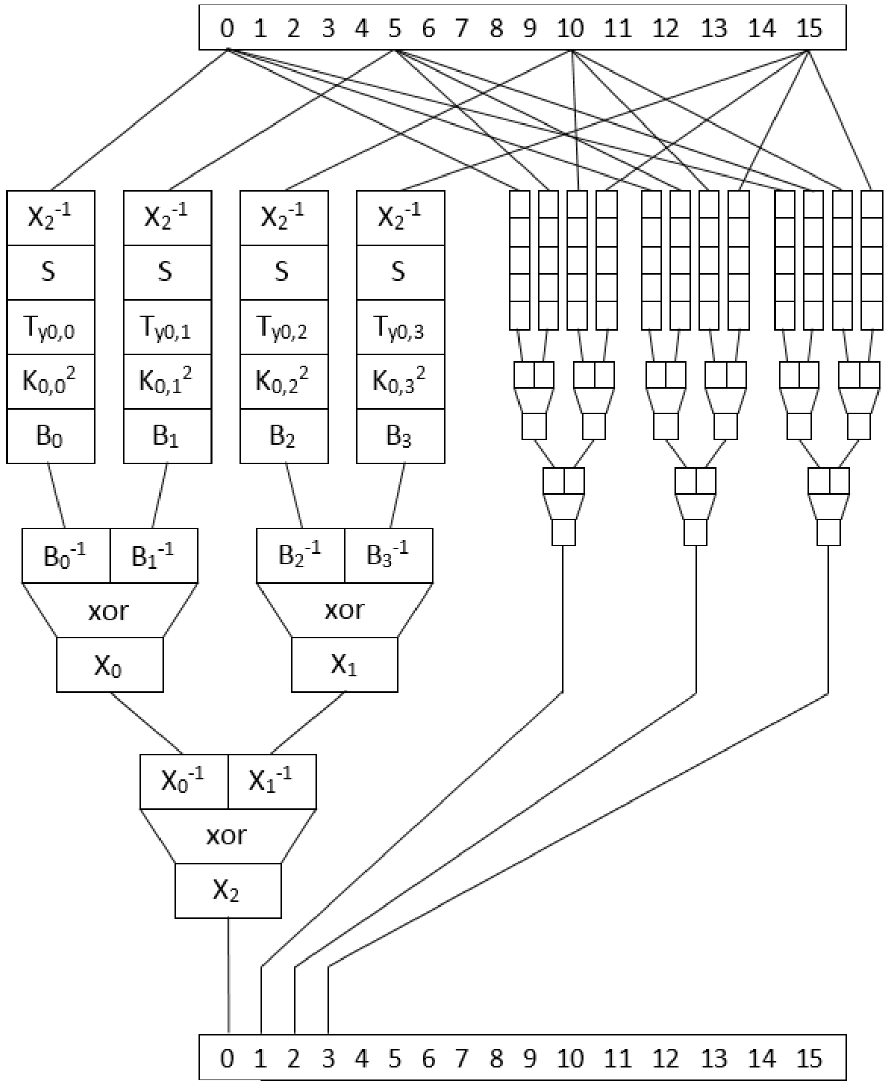
http://wiki.yobi.be/wiki/MoVfuscator_Writeup



Can DCA fail?

Yes!

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



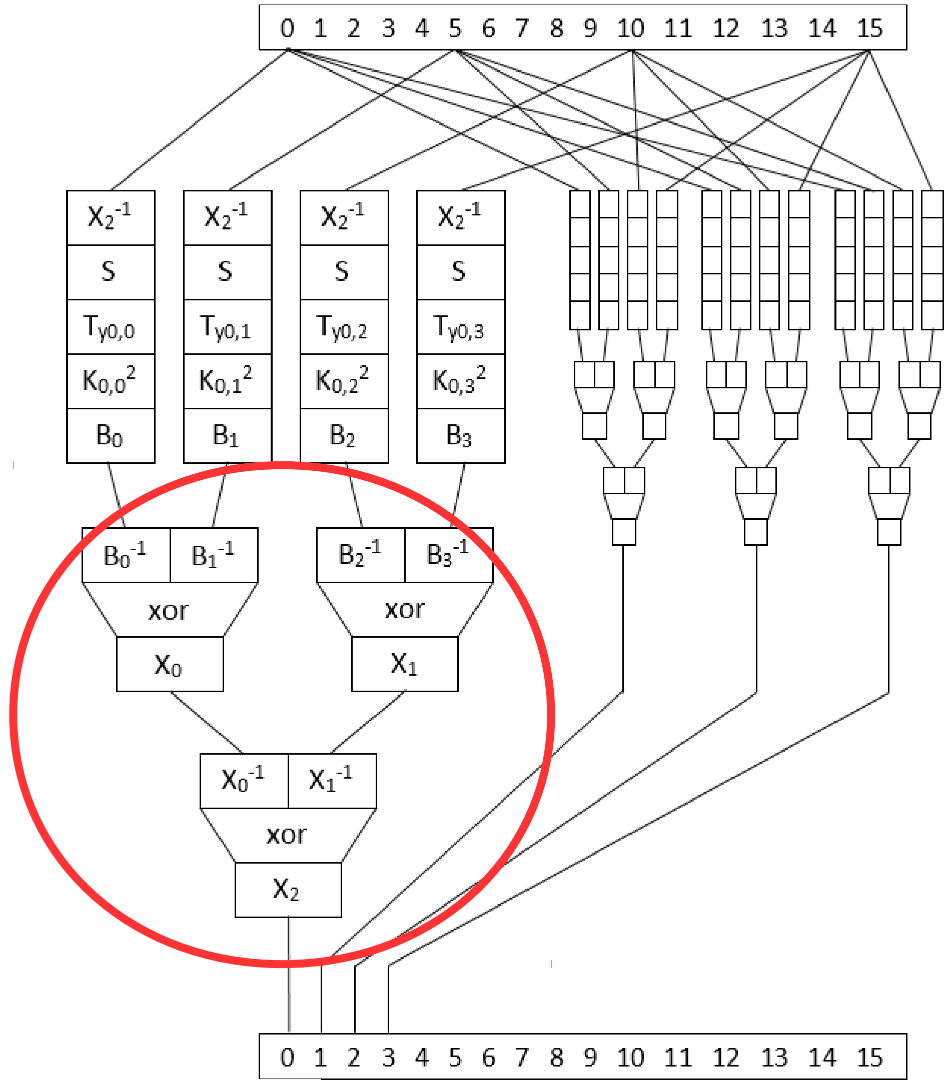
Can DCA fail?

Yes!

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

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,…

Take also care of code lifting, inversing 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

TROOPERS release!

https://github.com/SideChannelMarvels

Tracer
- TracerGrind
- TracerPIN
- TraceGraph

Deadpool
- White-boxes
- Attack automation

Daredevil
- Side-channel analysis (CPA)

Side-Channel Marvels

TROOPERS release!

https://github.com/SideChannelMarvels

Current team:

Charles Hubain (Quarkslab)

Joppe Bos (NXP)

Michael Eder (TUM, Fraunhofer AISEC)

Paul Bottinelli (EPFL)

Philippe Teuwen (Quarkslab)

Van Huynh Le (U.Twente, NXP)

Wil Michiels (NXP, TU/e)

Oh, BTW...

Orka
- Docker images

THANK YOU!
QUESTIONS?

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

@doegox

Image source: “A Beautiful Mind”