Design de cryptographie white-box : et à la fin, c'est Kerckhoffs qui gagne

SSTIC 2016

Charles Hubain & Philippe Teuwen

3 juin 2016

$ quisommesnous

Charles Hubain alias @haxelion

Philippe Teuwen alias @doegox

- @ Quarkslab
- ♥ le libre, la sécu, les Capture-Ze-Flag
- 웹 http://wiki.yobi.be http://haxelion.eu/

Note:

Les résultats présentés ici sont principalement le fruit de recherches menées lorsque nous travaillions pour NXP Semiconductors

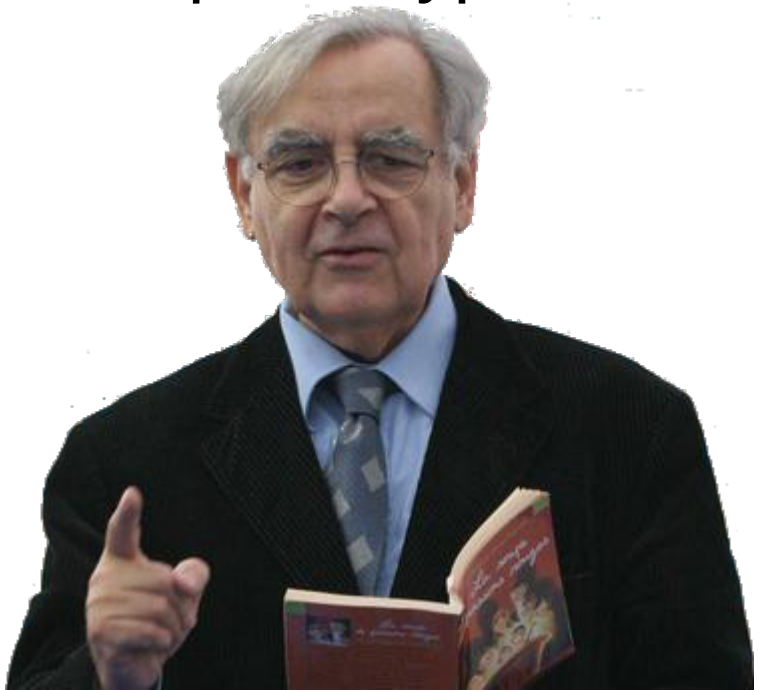
NXP

Crédits: corkami.com

Nos excuses mais...

Ecole de pensée française:

on dit “chiffrer”, pas “crypter”



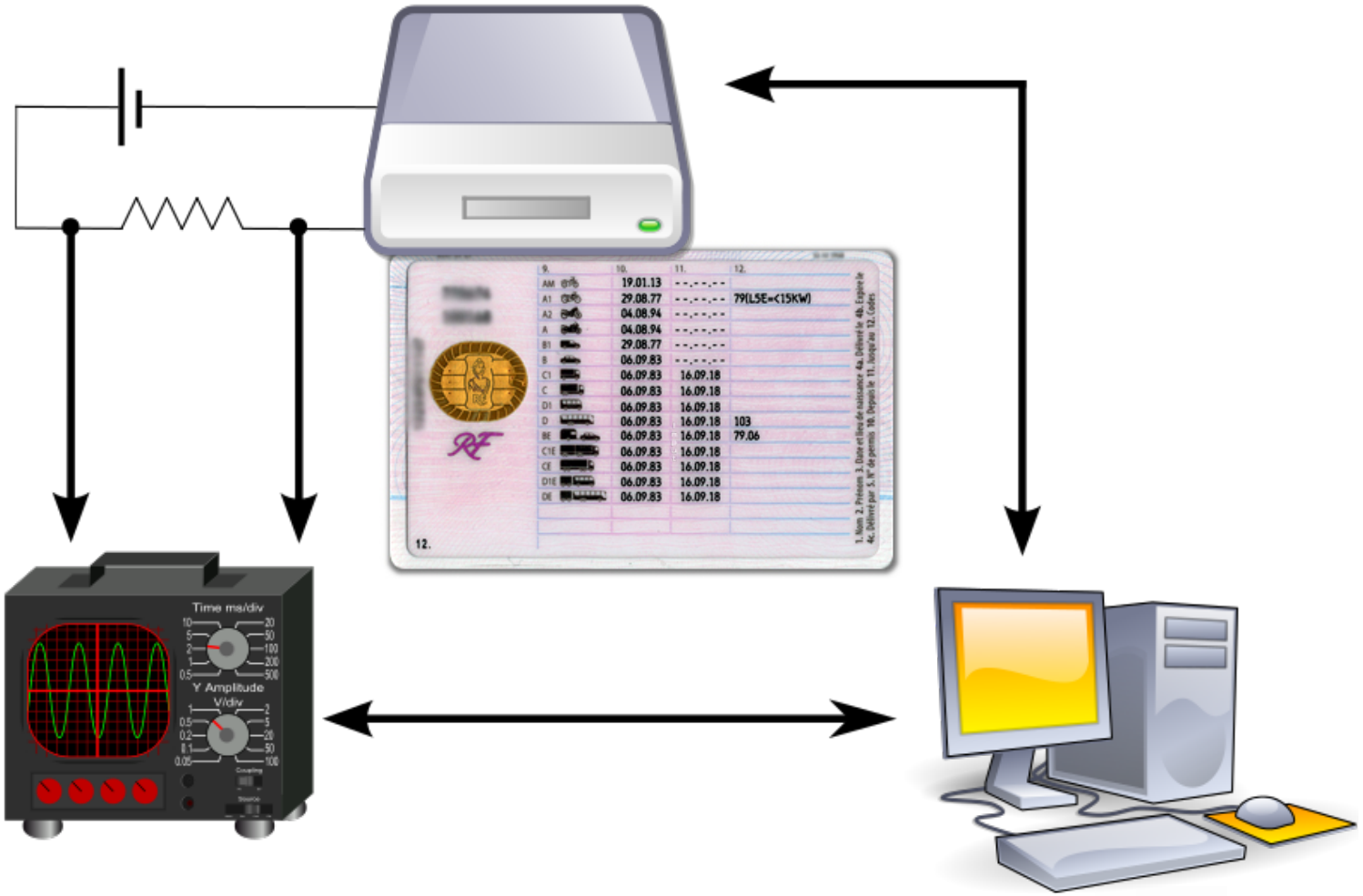
Ecole de pensée belge:

“décrypte le cipher”

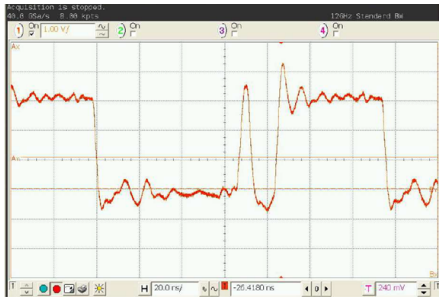
BREVE INTRO A LA CRYPTO WHITE-BOX

Rappel: modèle “Black box” => modèle “Grey box”

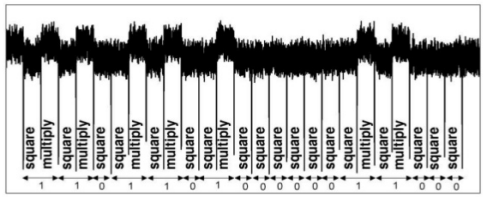
Modèle “Grey box”

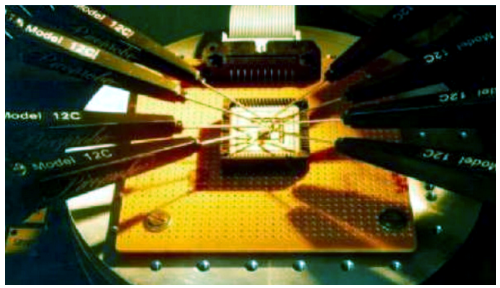


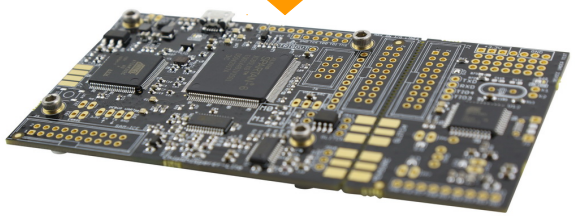
Modèle “Grey box”



Simple **P**ower **A**nalysis: Directly analyze (few) traces, for example RSA:

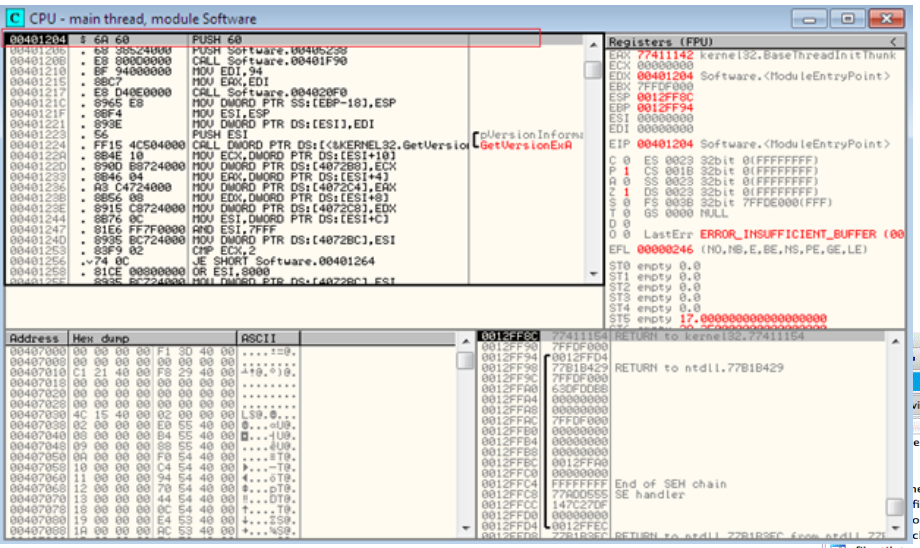


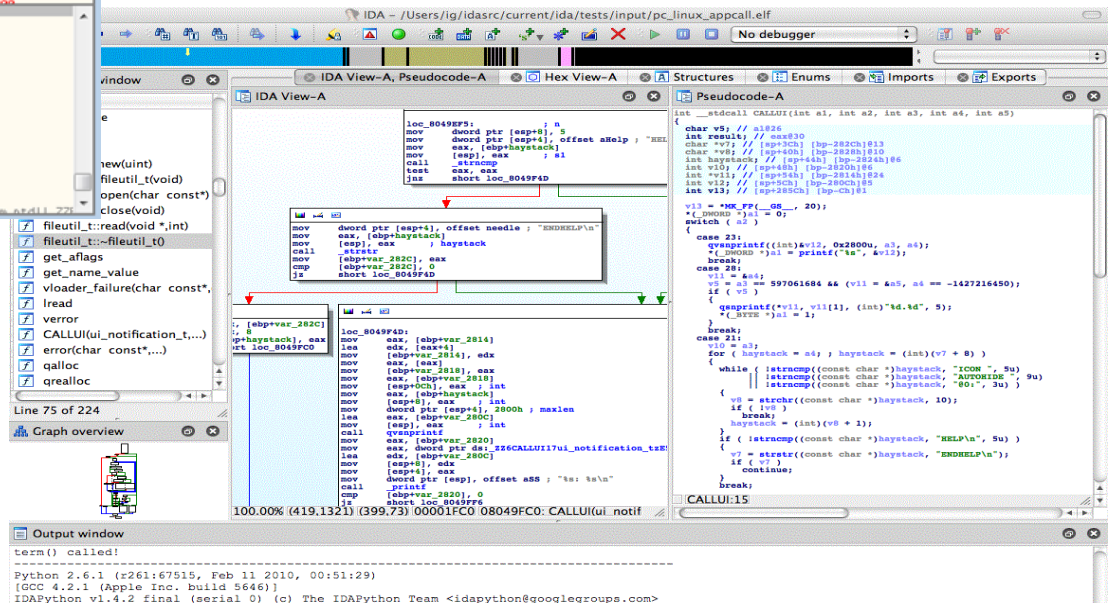




Modèle “White box”

Modèle “White box”



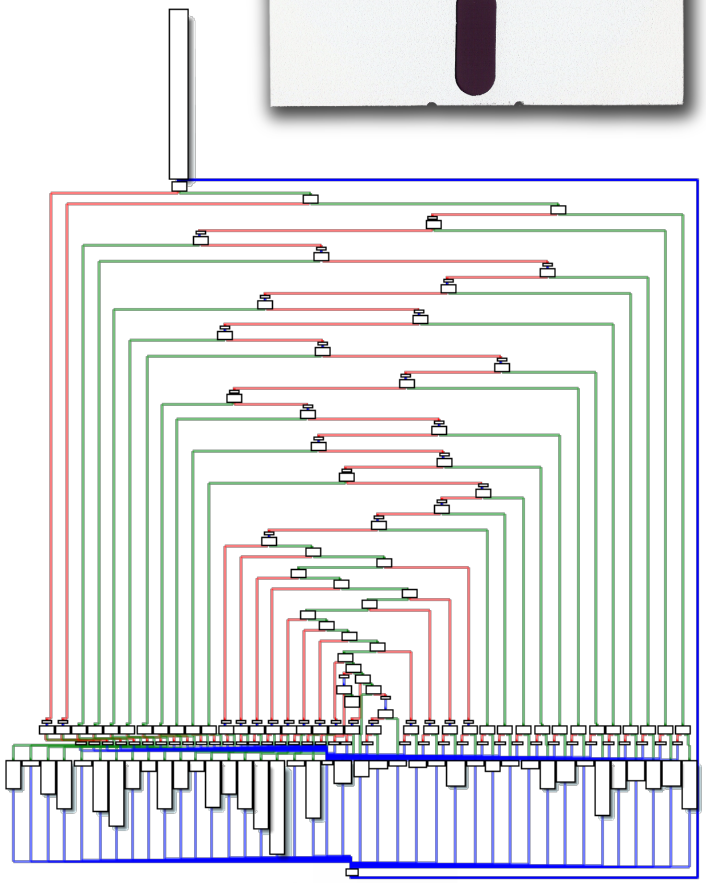


Ligne de défense = implémentation

Obscurcissement de code

Vérifications d’intégrité

Astuces “anti-debug”



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

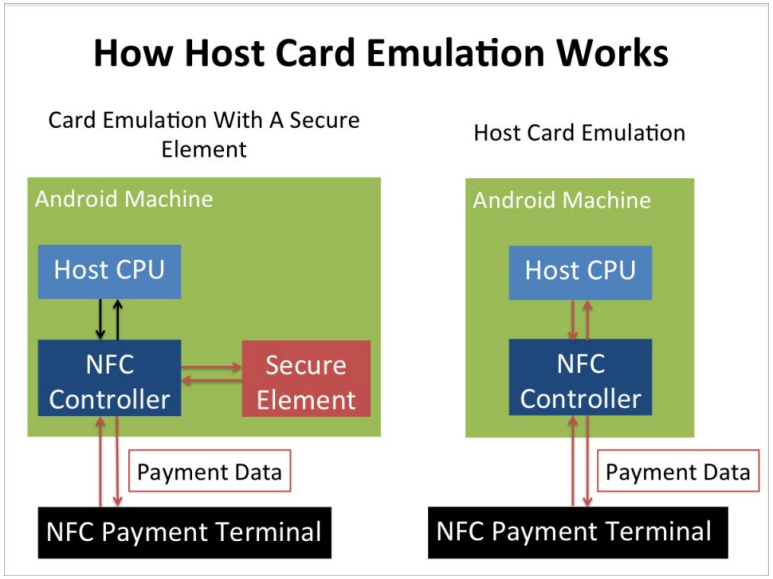
Illustrations: par @xoreaxeaxeax à propos de M/o/Vfuscator et par blog.quarkslab.com à propos d’OLLVM

Cryptographie dans le modèle “White box”

Pourquoi faire?

- Gestion de droits numériques (DRM) ↔ ~~criminels~~ utilisateurs
- Paiement mobile, HCE ↔ code malveillants

Source: “l'industrie du film”



Source: Business Insider

Cryptographie “White box”

Obscurcissement de code: inadéquat

→ Chow et al. (2002)

- Implémentation “idéale”:
  4.94×10^{27} TB
- Implémentation pratique d’AES:
  752kB
  Encodage des valeurs intermédiaires

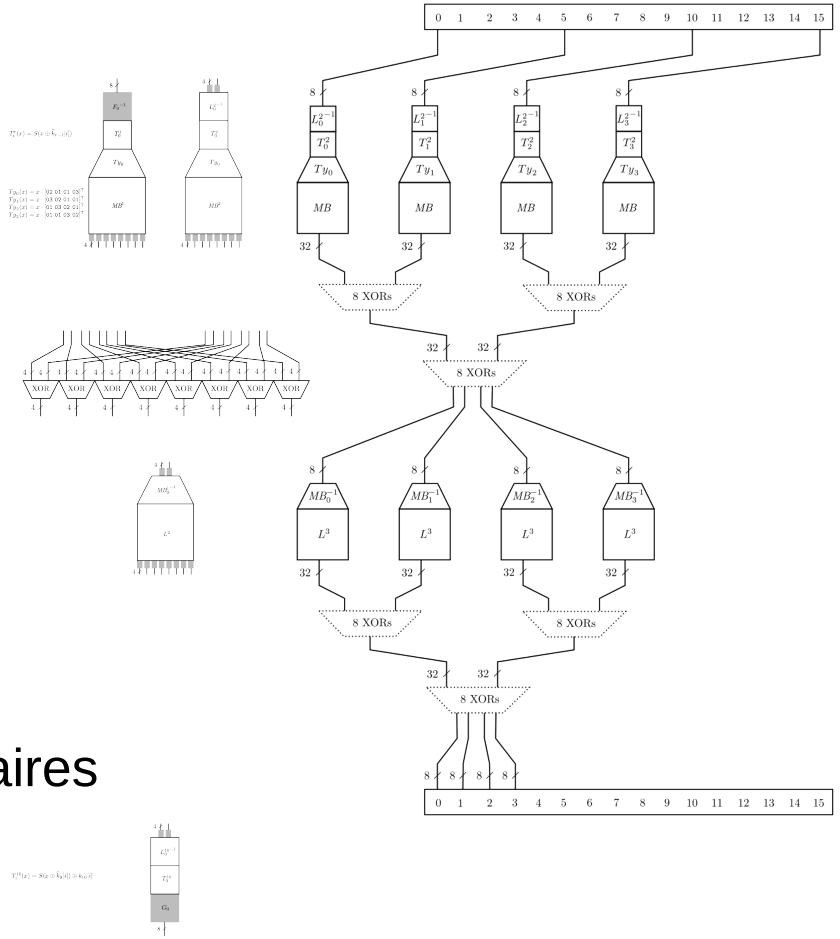


Illustration tirée de “A Tutorial on White-box AES” par James A. Muir

Cryptographie “White box”

Petit résumé:

- Attaques académiques → nouveaux designs → ...
- Tous les modèles académiques ont été cassés

Réponse de l’industrie:

- Gardons nos designs secrets
- Enterrons les implémentations sous des couches d’obscurcissement, de vérif. d’intégrité, trucs “anti-debug”
- Hé ! Regardez on a un Secure Element logiciel !

Attaques “académiques”?

1. défaire les protections
2. être *aware* du design
3. appliquer l’attaque:

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*

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

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

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

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

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:

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

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

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

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

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

The obfuscated t-box is

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

Hence the transformed function is:

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

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

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

Extraits de:

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

Attaques “académiques”?

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*

$$\circ\, AES^{(r,\pi^{(r)}(j))} \circ \ldots^{(r,j)} = \mathtt{MC}^{(r,j)} \circ (S,S,S,S) \circ \oplus_{r} \ldots_{\leq 3} \ ,$$

$$\ldots\text{ith} \qquad [\bar{k}_i^{(r,j)} \ldots^{(r,j)})^{-1}([k_i^{(r,\pi^{(r)}(j))} \ldots_{\leq i}$$

$$\ldots\text{and} \qquad \mathtt{MC}^{(r,j)} \ldots^{(r,j)} \circ \mathtt{MC} \circ \Pi_1^{(r,j)} \ .$$

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

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

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:

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

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

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

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

$$\mu_r(x_0x_1 \ldots y_0 \ldots y_0 \ldots y_5 x_{\mu_r^{-1}(0)} x_{\mu_r^{-1}(1)} y \ldots (2) x_{\mu_r^{-1}(3)} \ldots y_{42} \ldots y_4 \ldots x_{\mu_r^{-1}(47)}$$
$$\gamma_r(\ldots z_1 \ldots = z_{\gamma_r^{-1}(0)} \ldots z_{(\gamma_r^{-1}(0)+5} \ldots (1)) \ldots z_{(\gamma_r^{-1}(11)+5)} z_{94} z_{95}$$

The obfusca… $T_r'^k(x) = (\ldots(x)$.

Hence the t…nsform…function is:

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

with

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

By setting

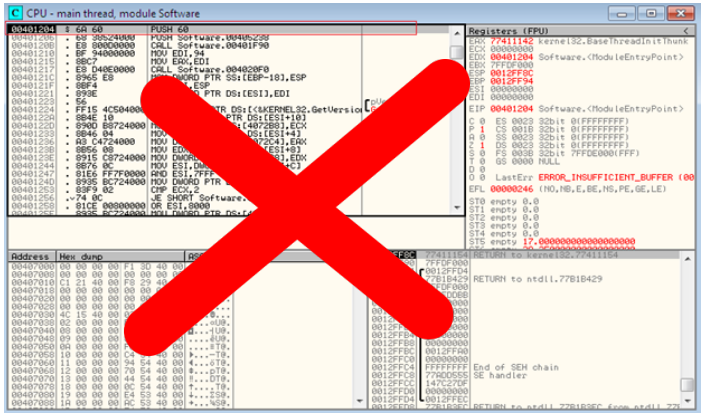
$$\tau_r'^k = \begin{cases} \psi_0 \, \delta_0 \, \beta \, \lambda & r = 0 \\ \psi_r \, \delta_r \, \tau_r^k \, \phi_r^{-1} & r = 1, .., n \\ \lambda^{-1} \, \delta_n^{-1} \, \psi_n^{-1} & r = n+1 \end{cases}$$

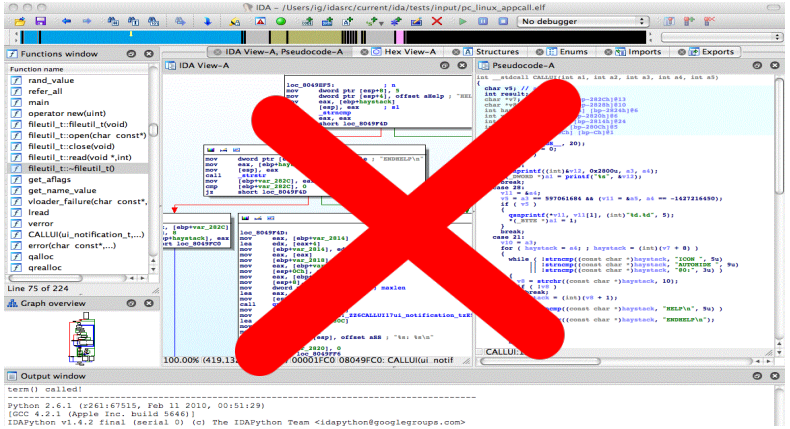
the resulting encryption operation is

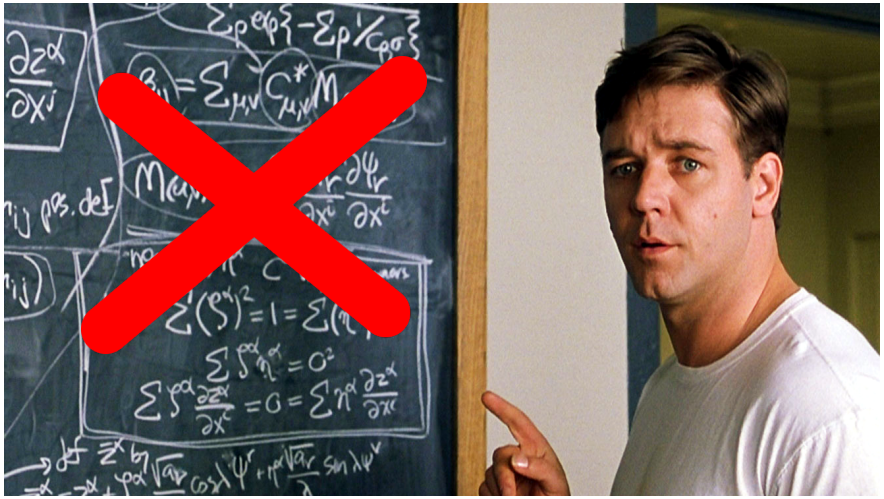
$$E^k(x) = \left[\tau_{n+1}'^k \cdot \left(\tau_n'^k \cdot T_n'^k \right) \cdot \ldots \cdot \left(\tau_1'^k \cdot T_1'^k \right) \cdot \tau_0'^k \right] (x)$$

看起来像天書

Notre but:







TRACES D’EXECUTION

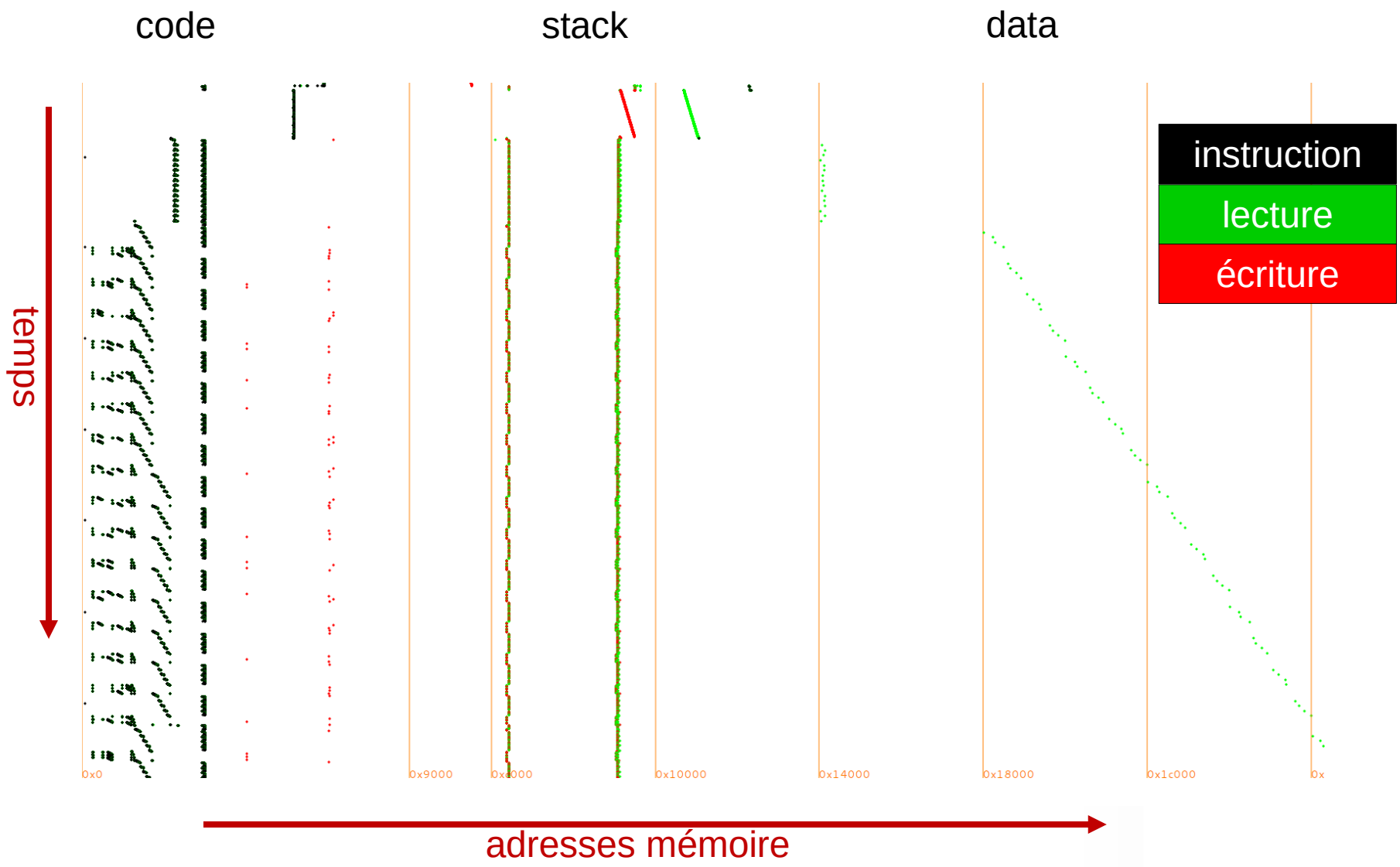
Tracer un exécutable

Enregistrer toutes les instructions et les accès mémoire

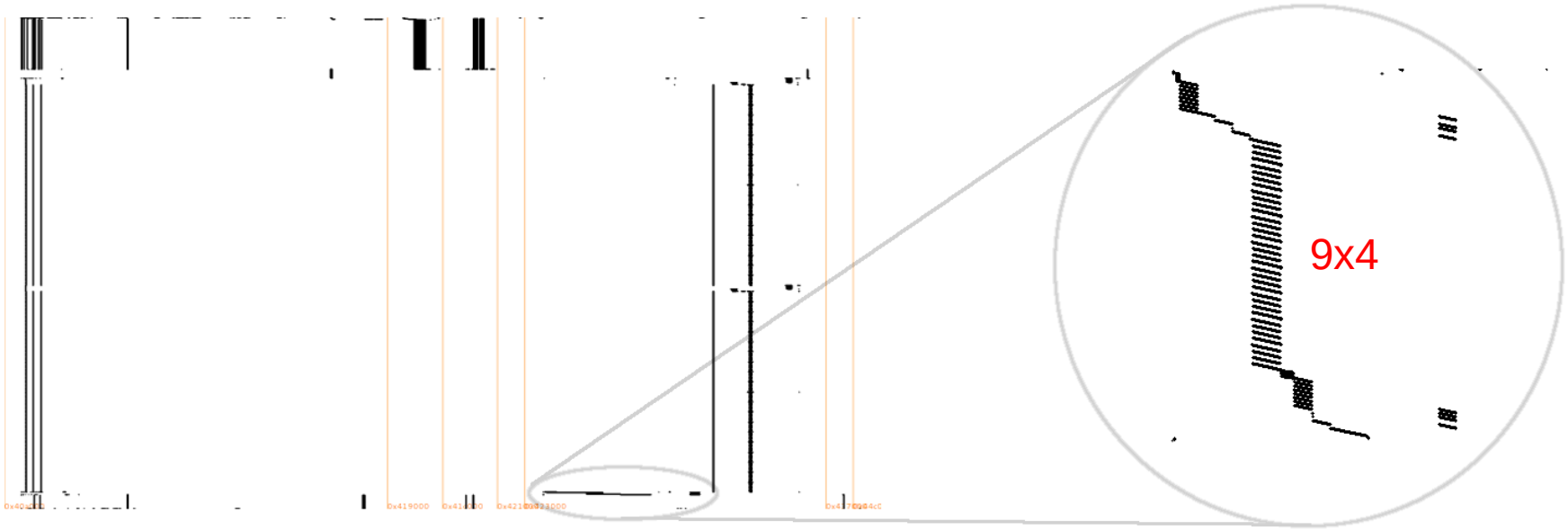
Exemples:

- Intel PIN
- Valgrind
- Instrumenter une VM (Java, Python,…)
- Instrumenter un émulateur

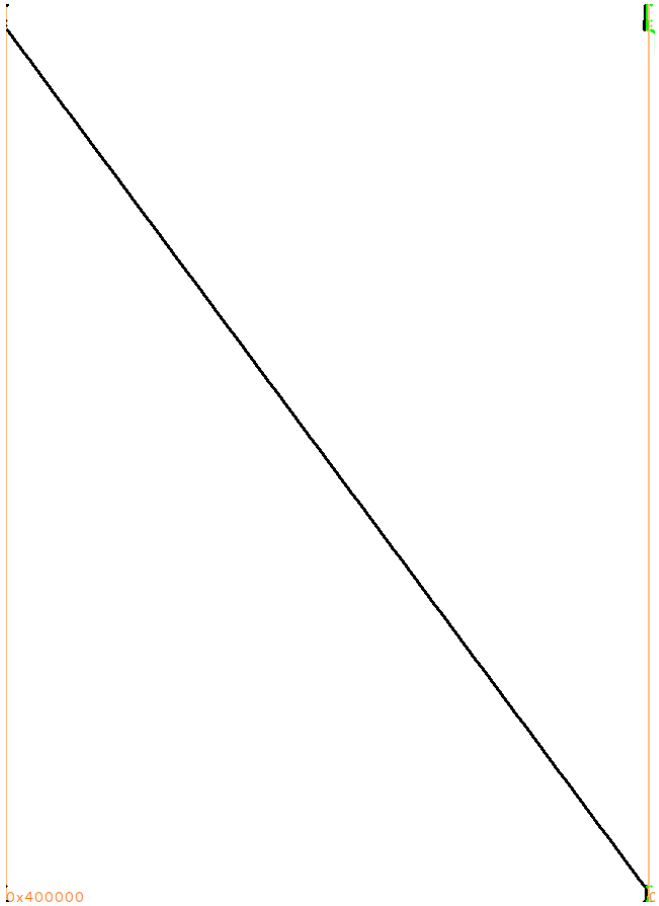
Même convention que pTra de Quarkslab



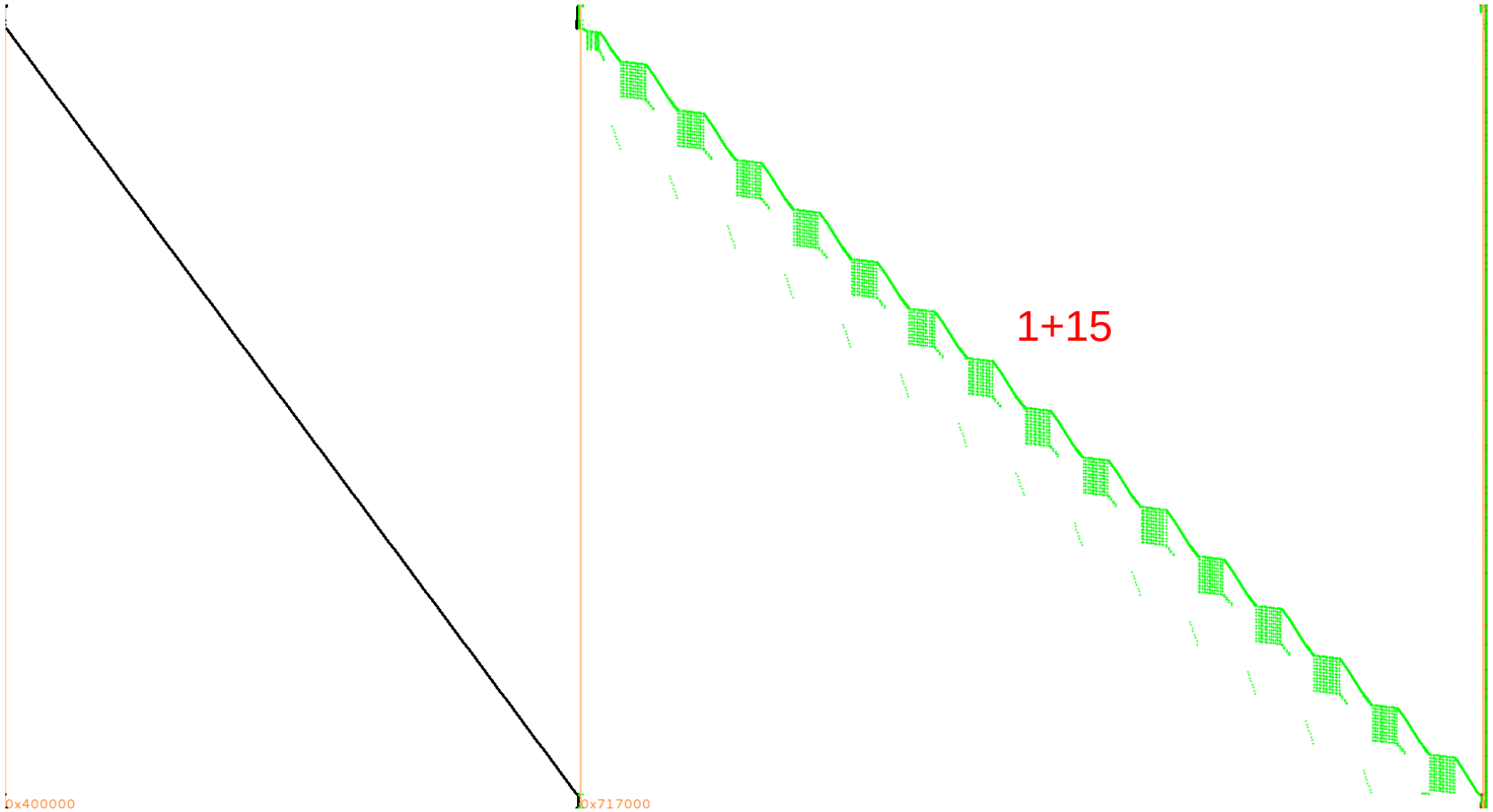
Identification visuelle de crypto: par le code



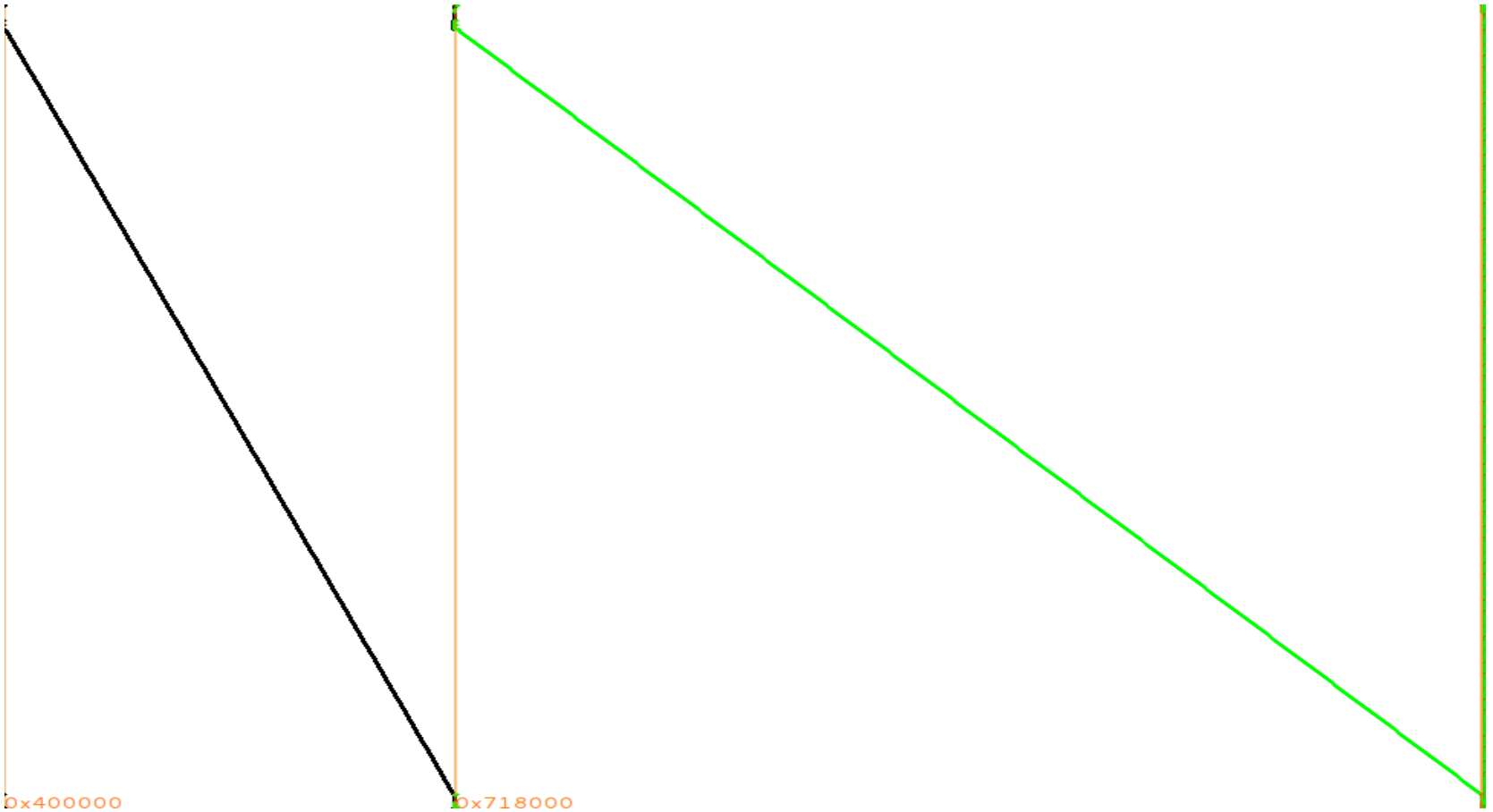
Identification visuelle de crypto: par le code ?



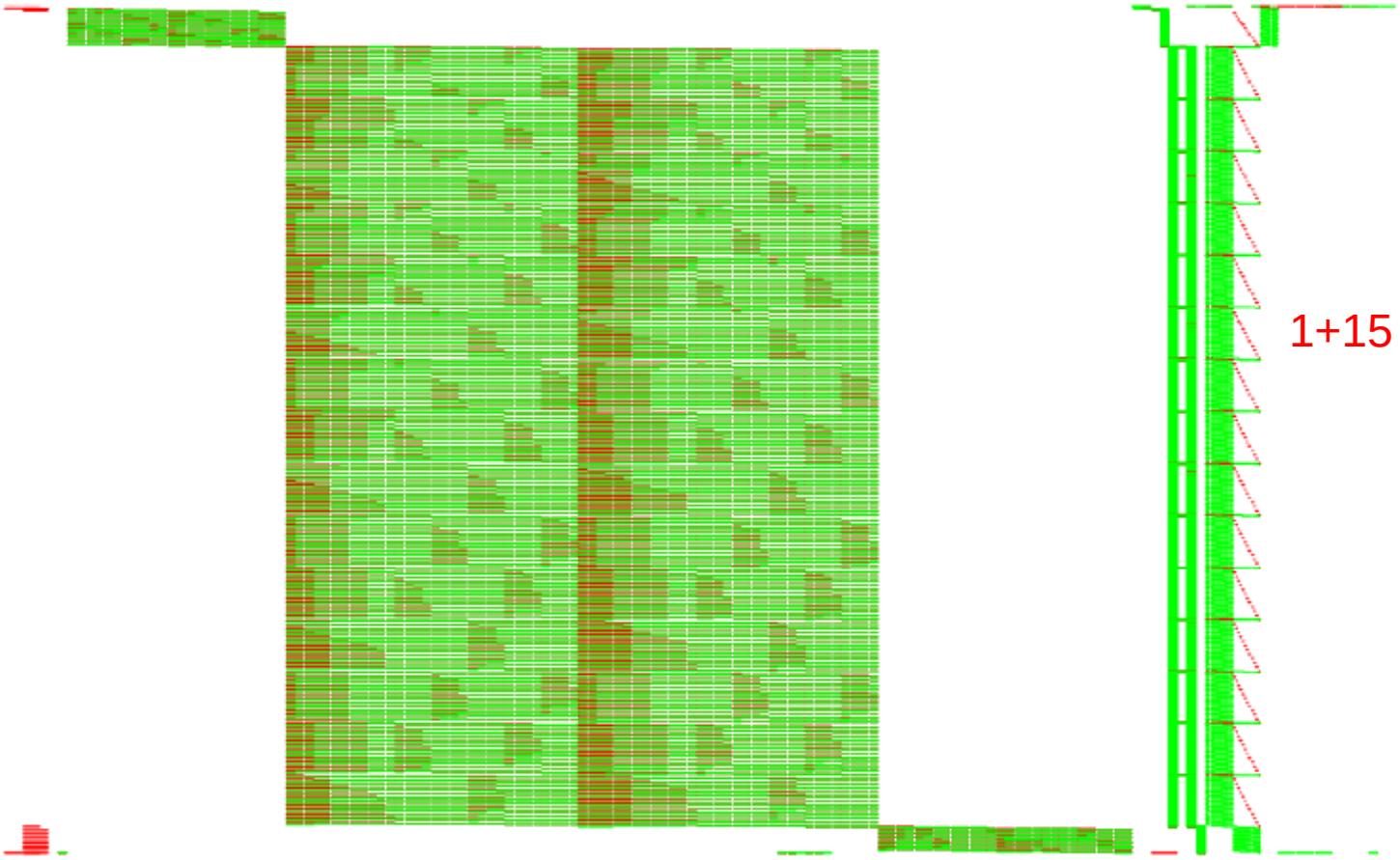
Identification visuelle de crypto: par les données !



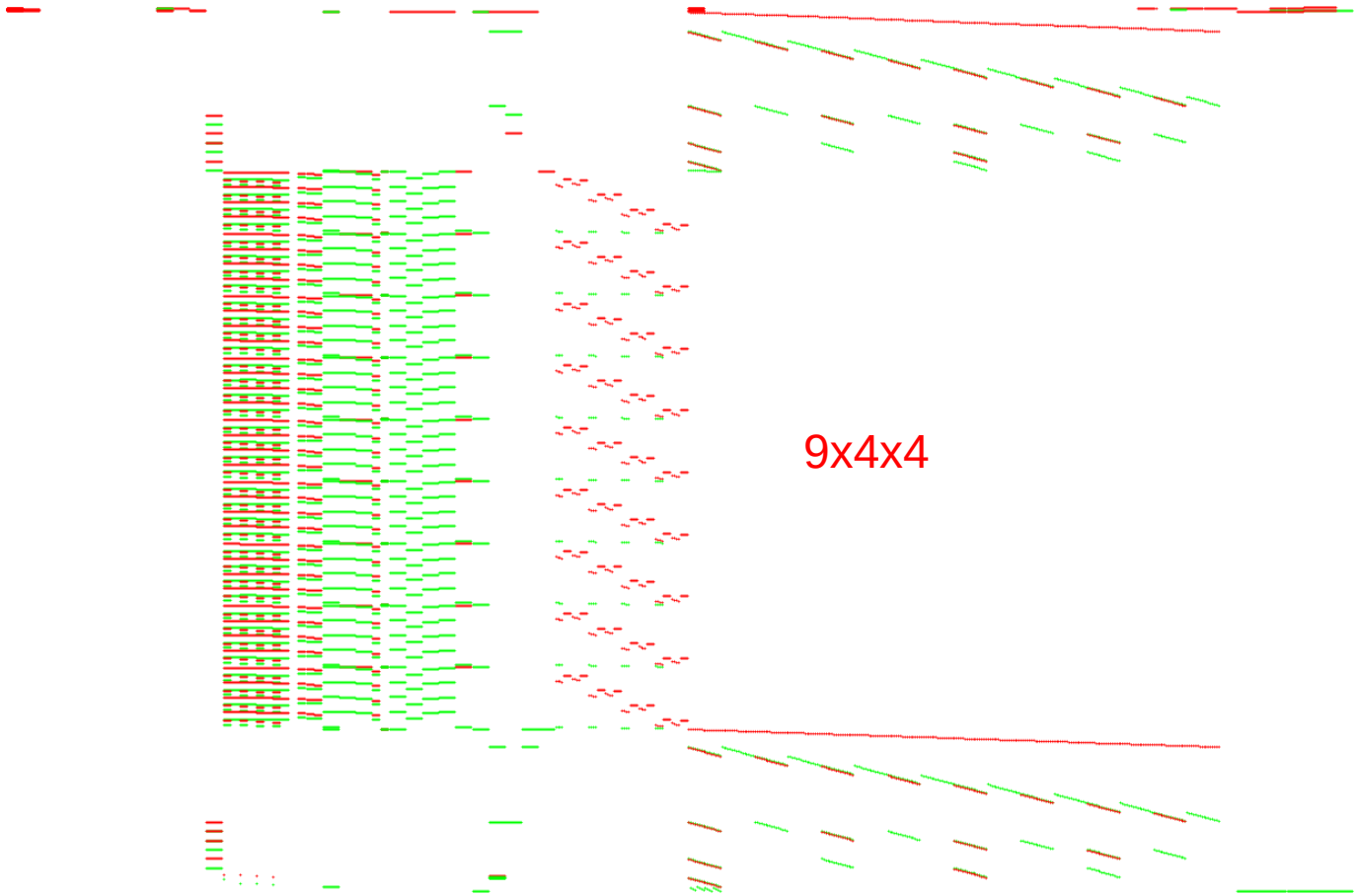
Identification visuelle de crypto: par les données ?



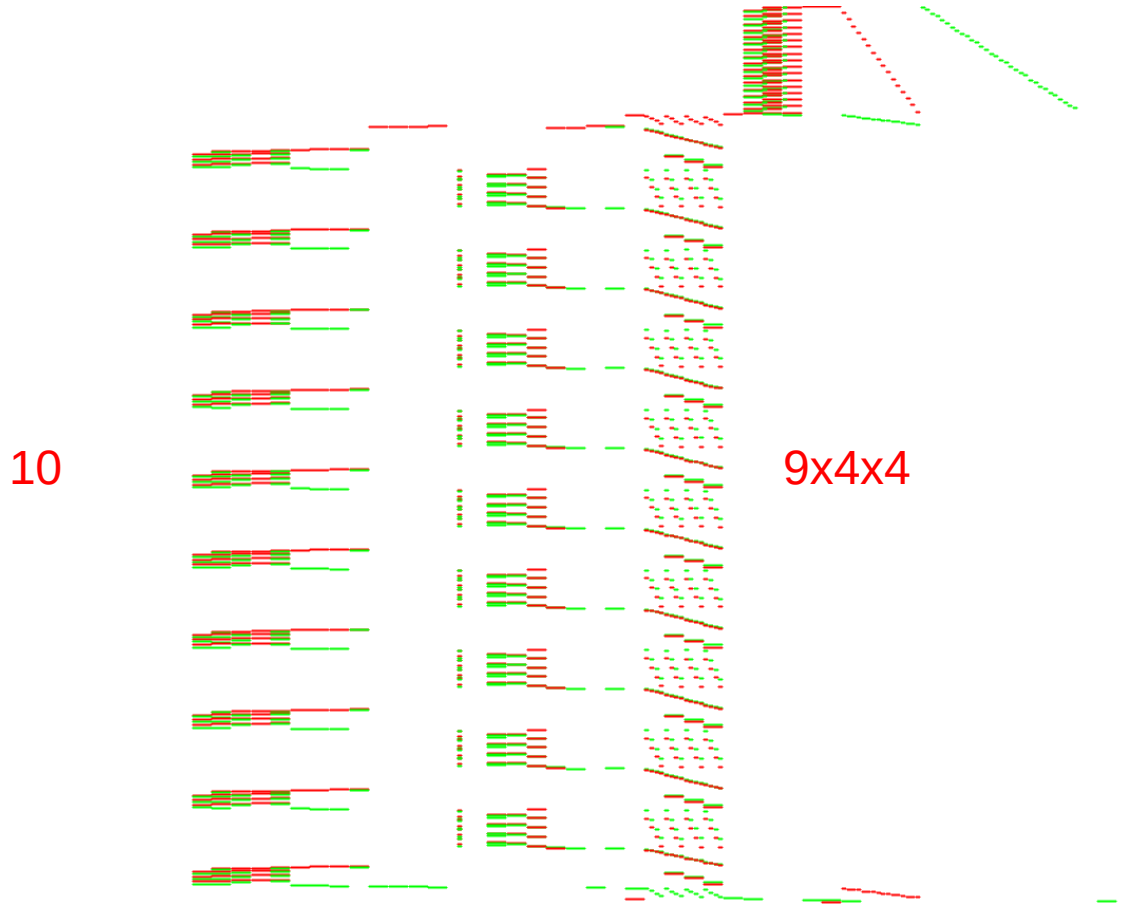
Identification visuelle de crypto: par la pile !



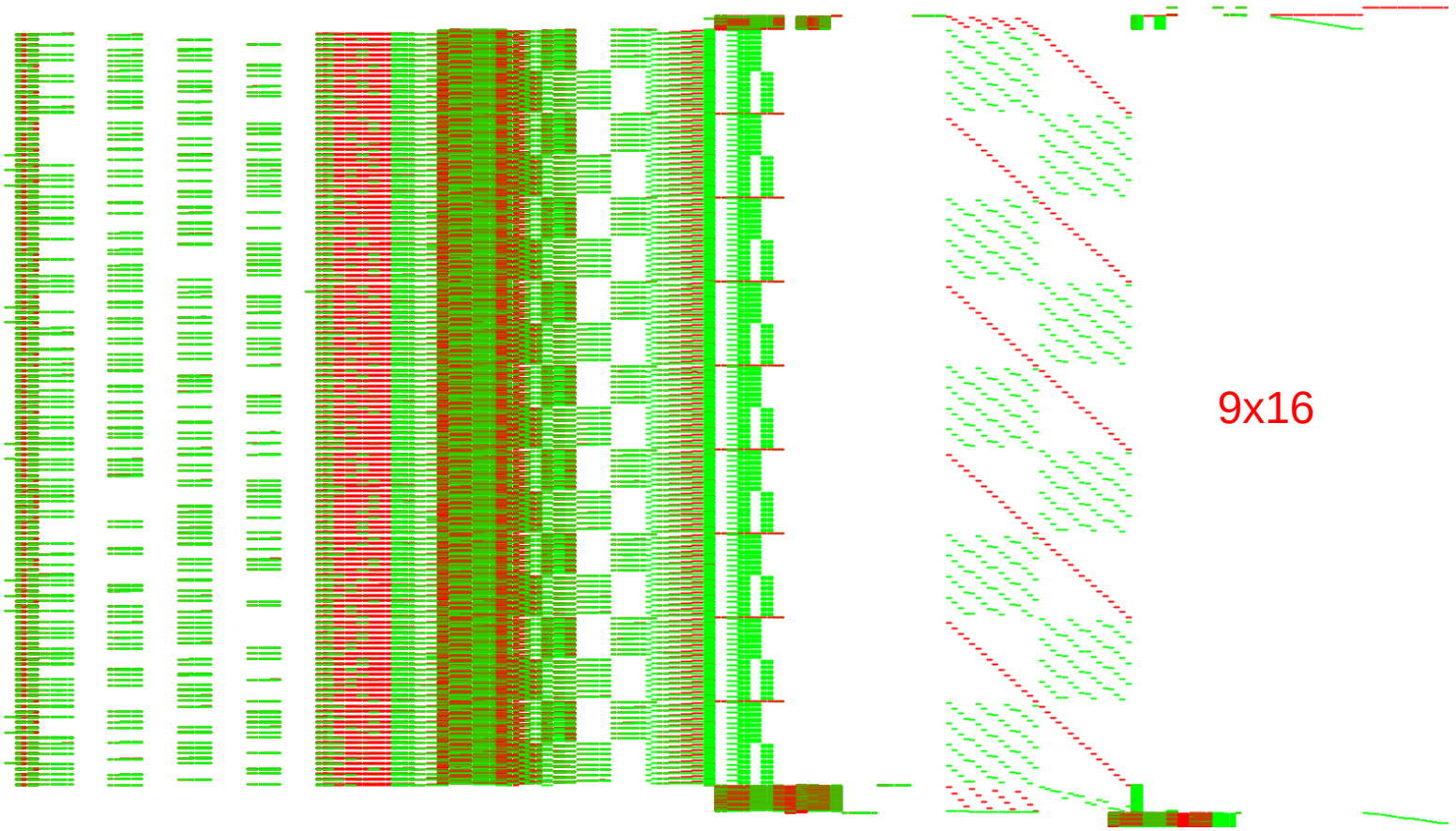
Identification visuelle de crypto: par la pile !



Identification visuelle de crypto: par la pile !



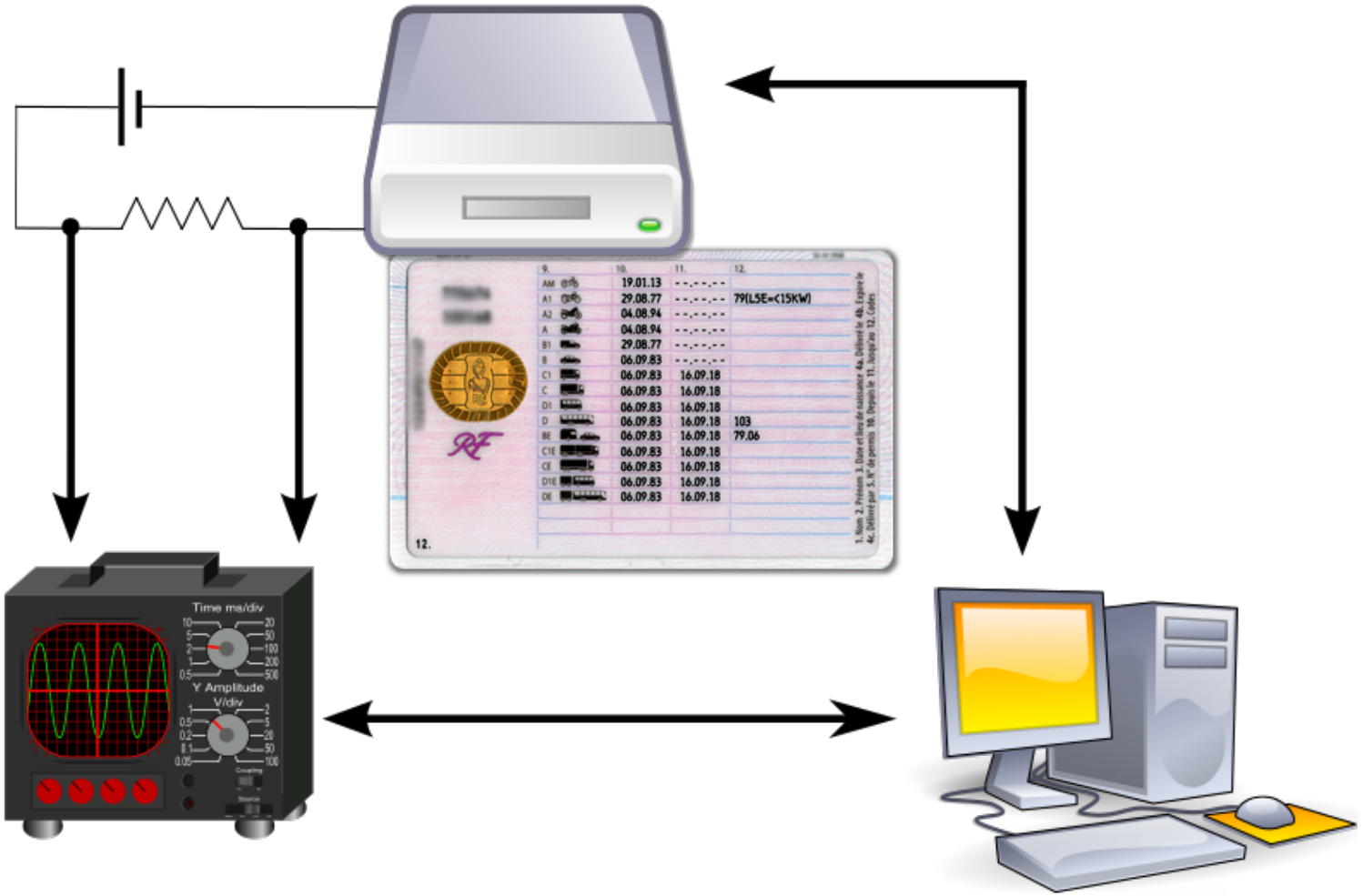
Identification visuelle de crypto: par la pile !



Et ma clé dans tout ça ?

DIFFERENTIAL COMPUTATION ANALYSIS

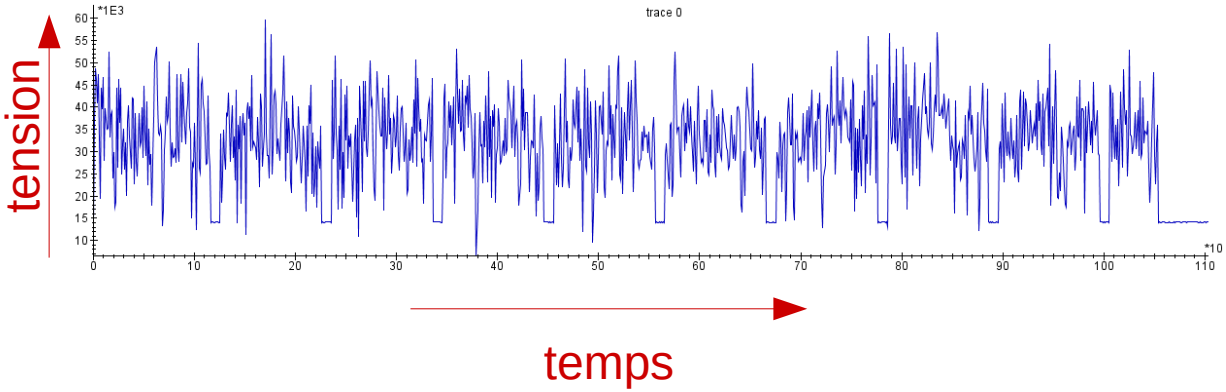
Rappelez-vous...

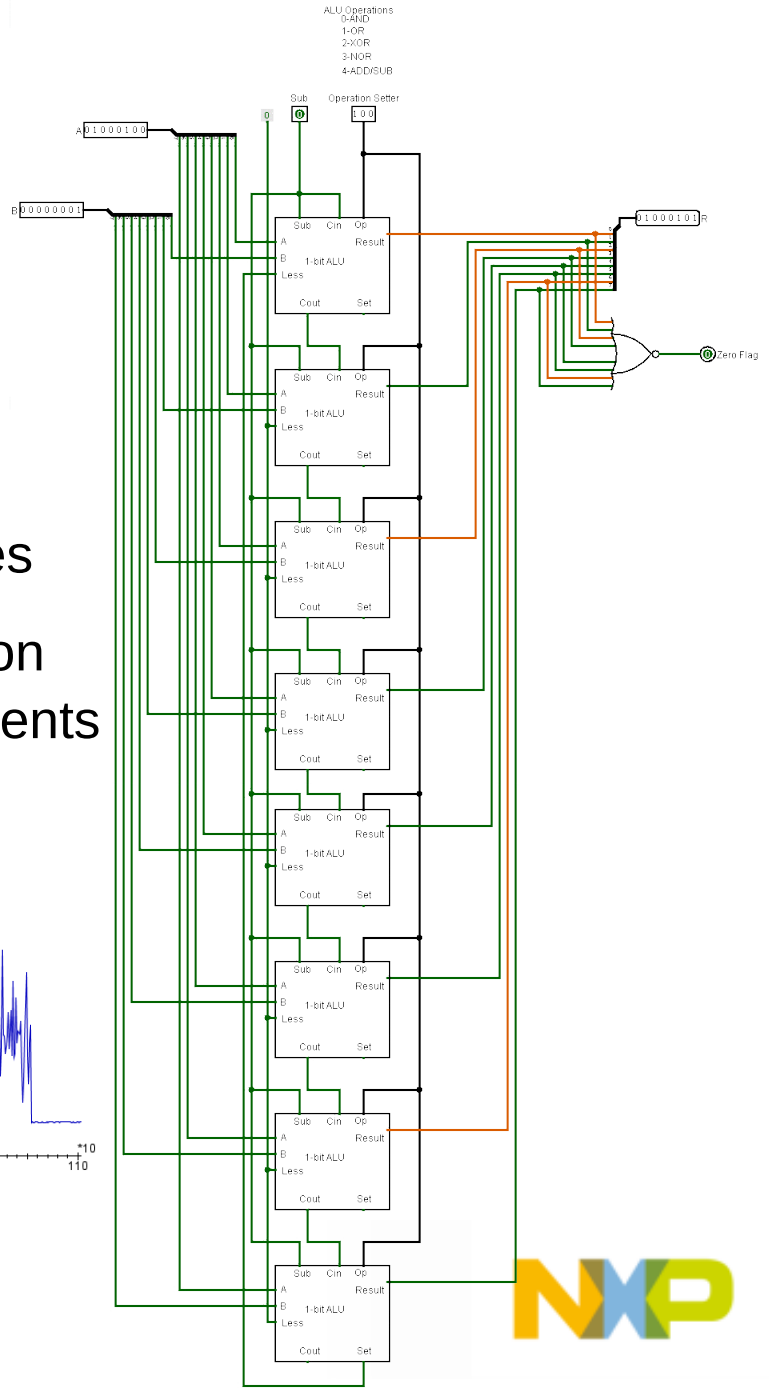


Differential Power Analysis

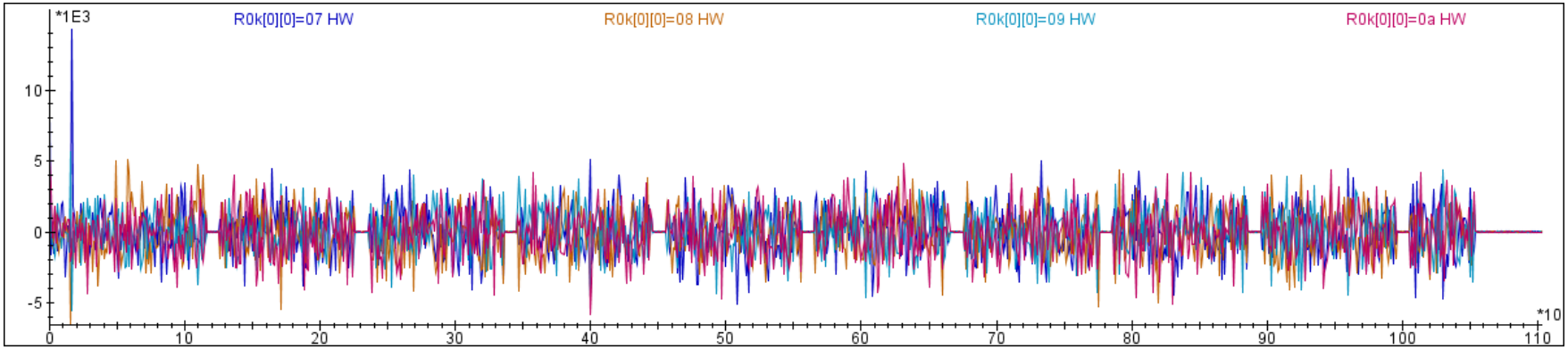
par P. Kocher et al. (1998)

- Corrélations possibles entre:
  consommation électrique et
  poids de Hamming de valeurs internes
- Enregistre les traces de consommation
  correspondant à des messages différents





Differential Power Analysis



Seuls prérequis:

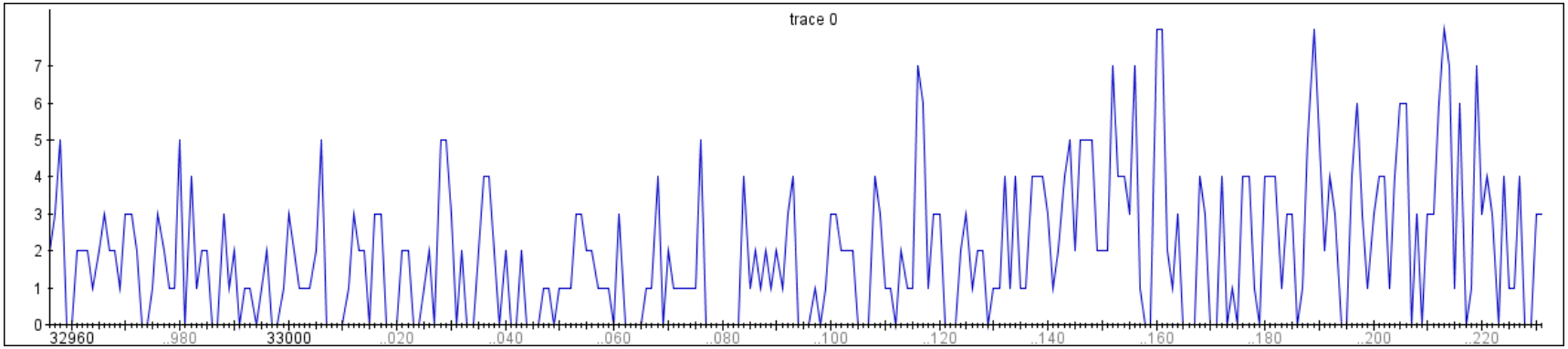
- Connaître l'entrée **ou** la sortie du chiffrement par blocs
- Pouvoir mesurer la consommation (ou les radiations EM)
- Présence de "fuites" (corrélations)

Differential Computation Analysis

Traces d'exécution → "traces de consommation" ?

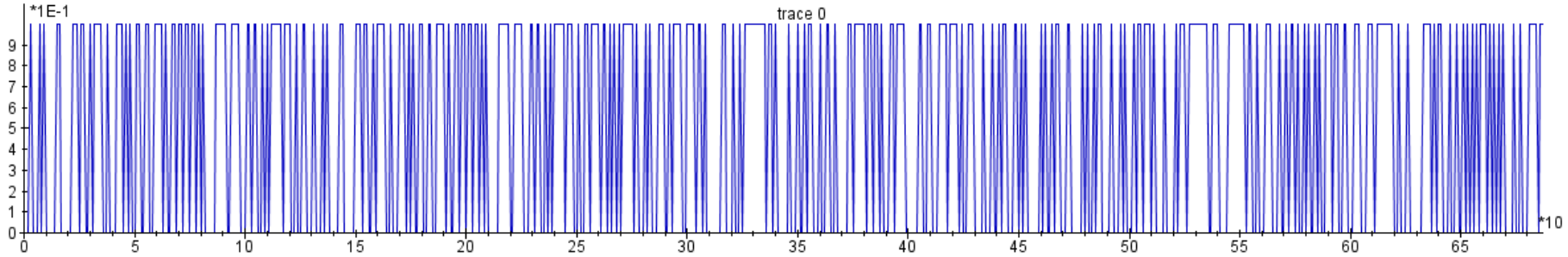
Adresses ou données des accès mémoire, écritures sur la pile,…

Exemple: octets lus → poids de Hamming ?



Differential Computation Analysis

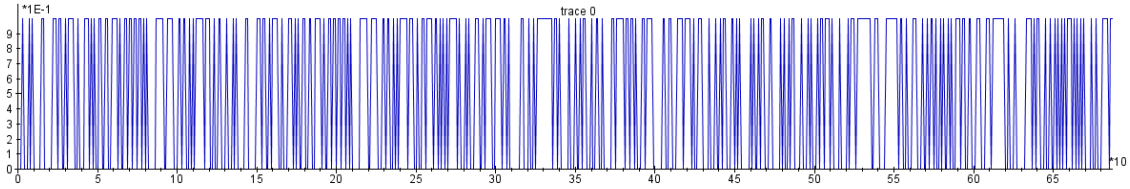
Octets → bits

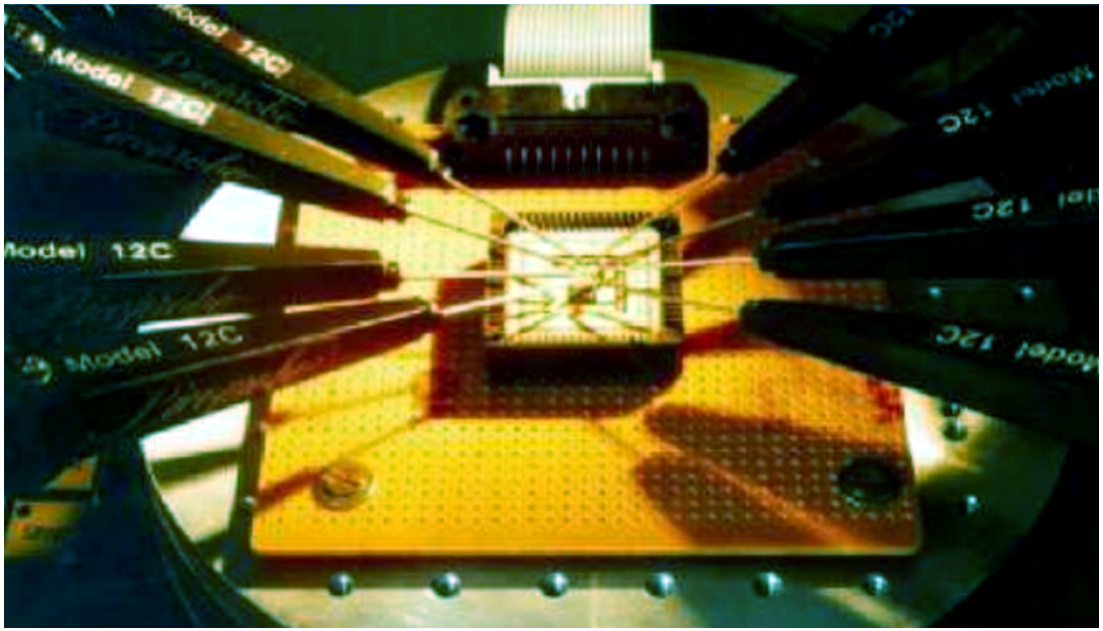


Yapluka → outils de DPA traditionnelle

Riscure Inspector, ChipWhisperer, Matlab ou… Daredevil!

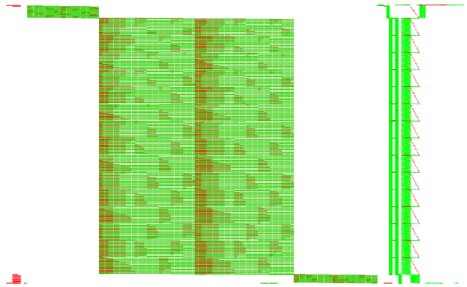
Differential Computation Analysis





Source: Brightsight

Challenge Wyseur



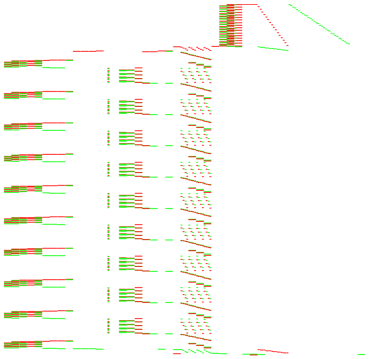
par Brecht Wyseur, 2007

implémentation WB-DES : Chow “plus some personal improvements”

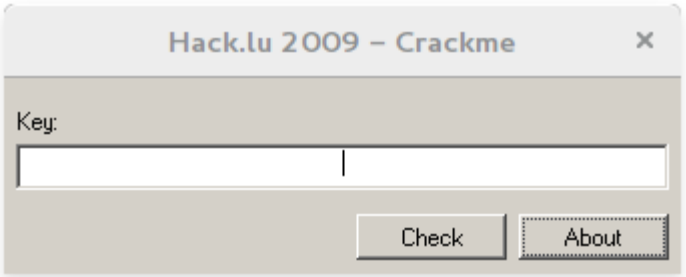
On télécharge…

1h et 65 traces plus tard, la clé est cassée !

Challenge Hack.lu 2009



Windows *crackme* par Jean-Baptiste Bédrune
Implémentation WB-AES : Chow



Paresse → Wine/Linux + xdotool (émulation clavier/souris)

16 traces

(challenge de CTF, pas d’encodages internes)

Challenge SSTIC 2012

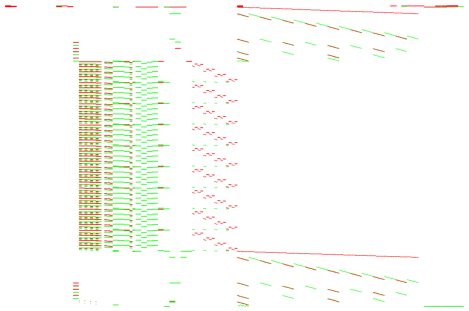
“white-box” en Python par Axel Tillequin
Implémentation WB-DES dans un objet “marshall”

→ instrumentation d’une classe externe “Bits”

16 traces

pas d’encodages internes

Karroumi



Dernière tentative académique de “réparer” Chow (2011)

Dual Ciphers, i.e. isomorphic AES ciphers:

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

김나리 생일 축하해

Notre propre challenge…

2000 traces, 500 traces après *tuning*

Quelques implémentations “white-box” propriétaires

DES & AES

Cassées en 200 à 2500 traces

Contre-mesures ?

Aléas dynamique?

- Pas de TRNG, juste le message d'entrée

Ajout de délais pseudo-aléatoires?

- Traces d'instructions → on réaligne

Bref, délicat…

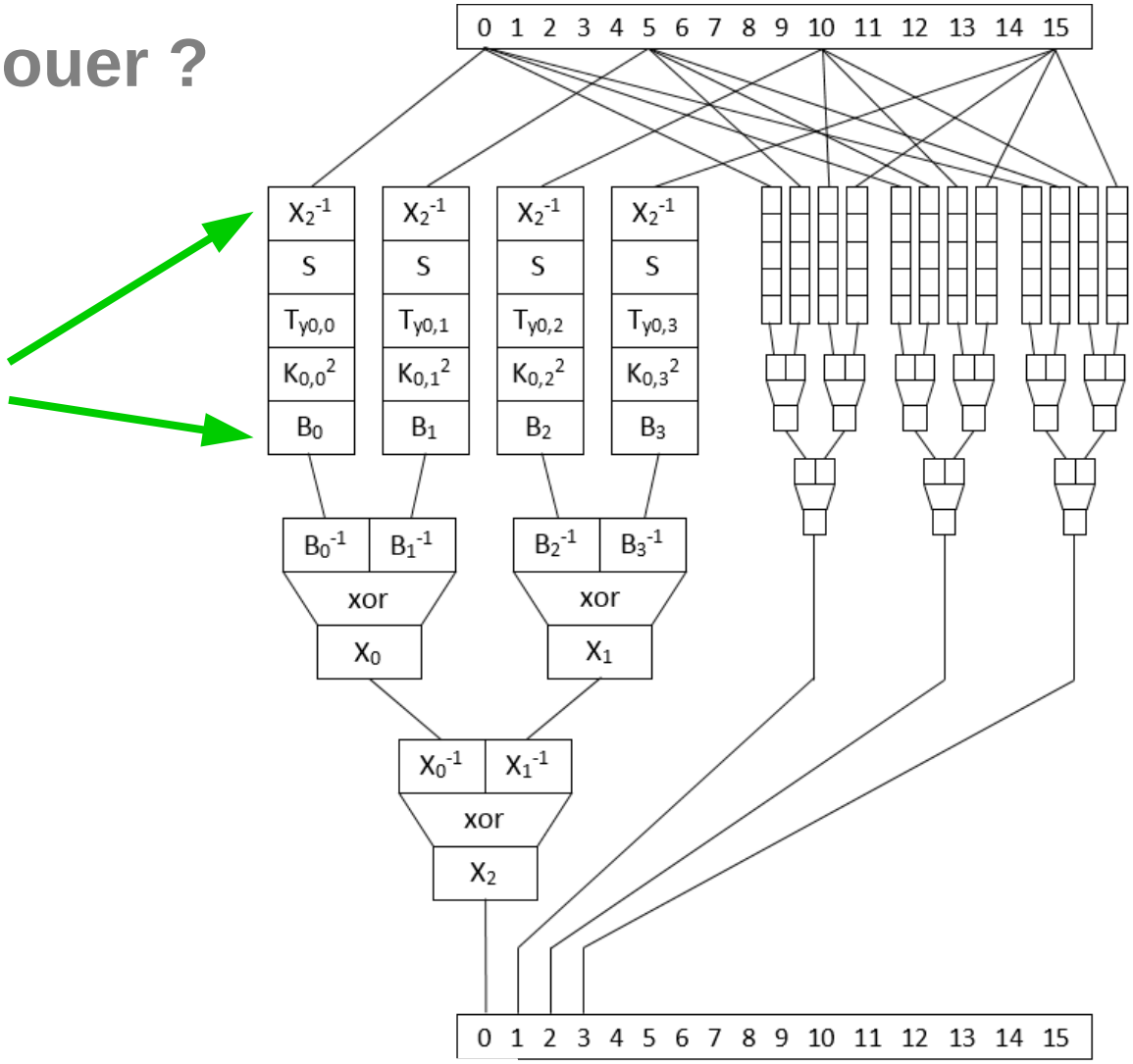
Une sécurité "parfaite" est illusoire, mais si le coût d'une attaque est supérieur au gain pour l'attaquant, vous avez réussi.

Oups, c'est la chute des prix...

La DCA peut-elle échouer ?

Oui !

Encodages “larges” (8x8) masquant la non-linérarité de la SBOX



La DCA peut-elle échouer ?

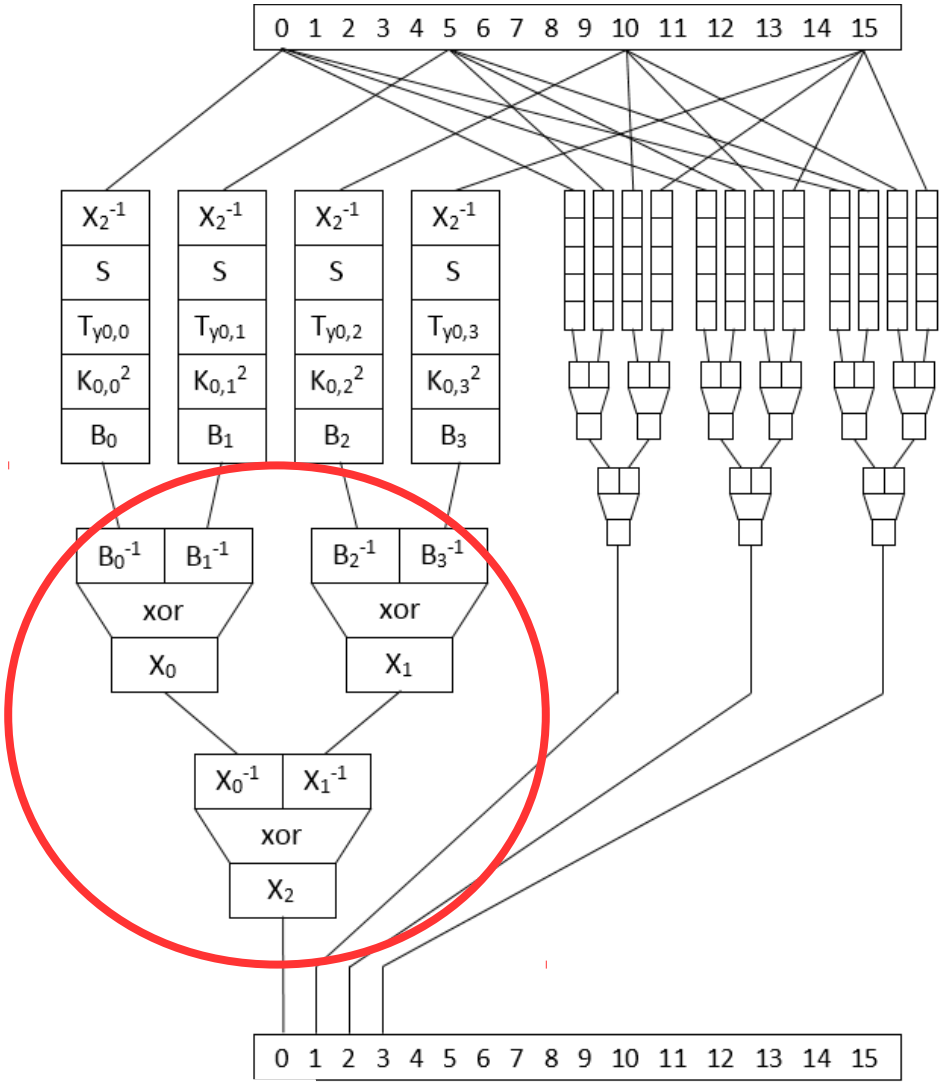
Oui !

Encodages “larges” (8x8) masquant la non-linérarité de la SBOX

→ grandes tables !

→ tendance à les réutiliser

→ autres types d’attaque



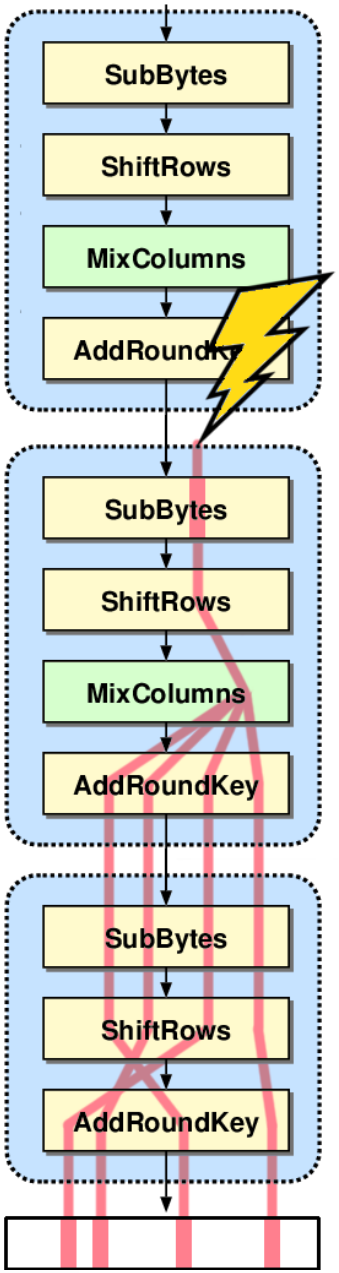
cf. write-ups
NoSuchCon 2013 et CHES 2015
http://wiki.yobi.be/wiki/CHES2015_Writeup

Autres attaques “grey box” à portée de main : Higher order DPA, CPA, DFA,…

La DFA, parlons-en...

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

- Prérequis: rejouer l'entrée, voir la sortie en clair
- Injection: source ; bin statique ; bin dynamique
- Injection à l'aveugle, voire à la bourrin
- 8 "bonnes" fautes suffisent (sur AES-128 enc ou dec)
- Temps d'analyse: qqs secondes



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)

Michael Eder (TUM, Fraunhofer AISEC)

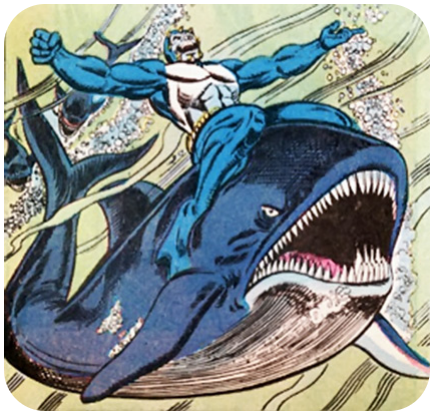
Paul Bottinelli (EPFL)

Philippe Teuwen (Quarkslab)

Van Huynh Le (U.Twente, NXP)

Wil Michiels (NXP, TU/e)

Sans oublier...



Orka
- images Docker

Image source: “A Beautiful Mind”

DEMO