White-Box Cryptography

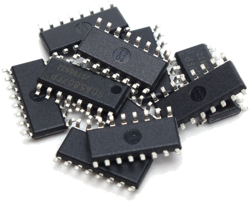
Matthieu Rivain

CARDIS 2017

CryptoExperts

How to protect a cryptographic key?

How to protect a cryptographic key?



Well, put it in a **smartcard** of course!

… or any piece of **secure hardware**

But...

- Secure hardware is **expensive** (production, integration, infrastructures...)
- Long lifecycle, limited updates
- Bugs, security flaws might occur
  - *e.g.* ROCA vulnerability (October 2017)

COMPLETELY BROKEN —

Millions of high-security crypto keys crippled by newly discovered flaw

Factorization weakness lets attackers impersonate key holders and decrypt their data.

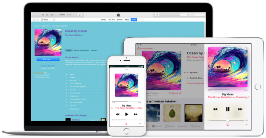
DAN GOODIN (US) - 16/10/2017, 16:30

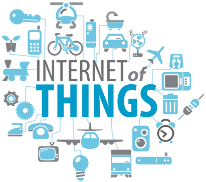
Pure software applications

- Advantages: cheaper, faster time-to-market, easier to update
- Big trend in ICTs: cloud service + mobile app
- HCE-based mobile payment
  - SE not available
  - Emulated SE in software
  - Short-term keys (tokens)
  - Regular authentication to server ("always on" paradigm)

Pure software applications

- IoT (without SE)
- Content protection, DRM
- OS / firmwares





Protecting keys in software?

- Potential threats:
  - malwares
  - co-hosted applications
  - users themselves
  - ...
- White-box adversary model
  - analyse the code
  - access the memory
  - tamper with execution
  - ...
- Ex: scan the memory for secret keys

Illustration: Shamir, van Someren. *Playing hide and seek with stored keys*.

White-box cryptography

General idea: hide the secret key in an obfuscated cryptographic implementation

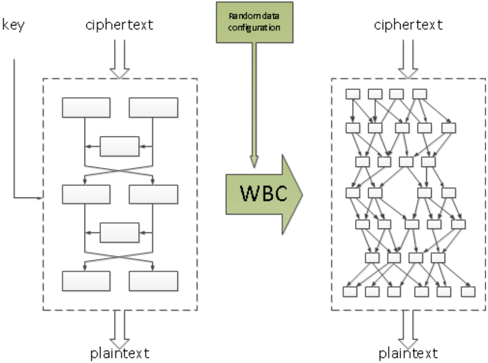
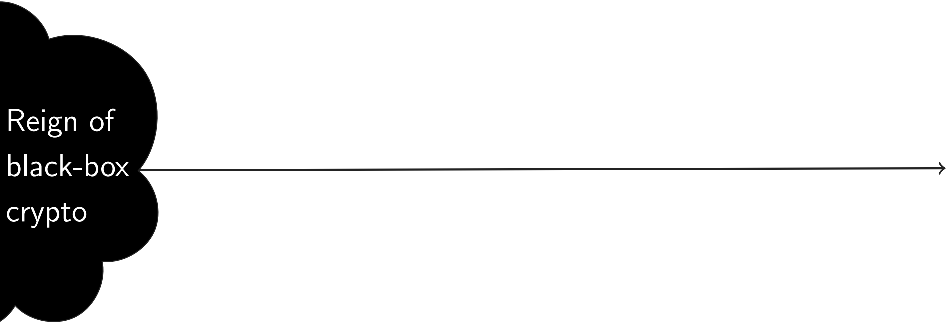
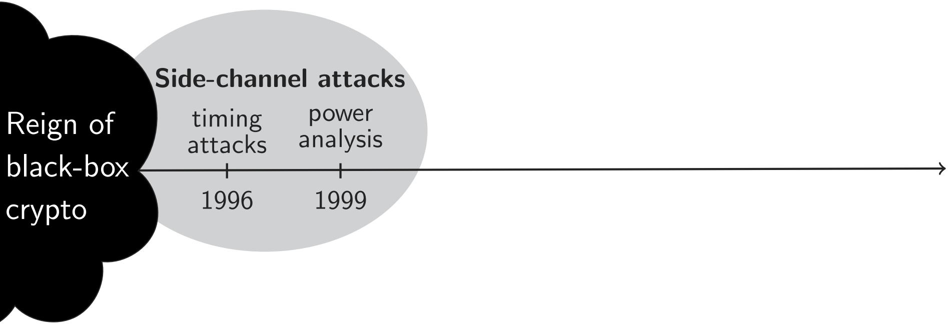


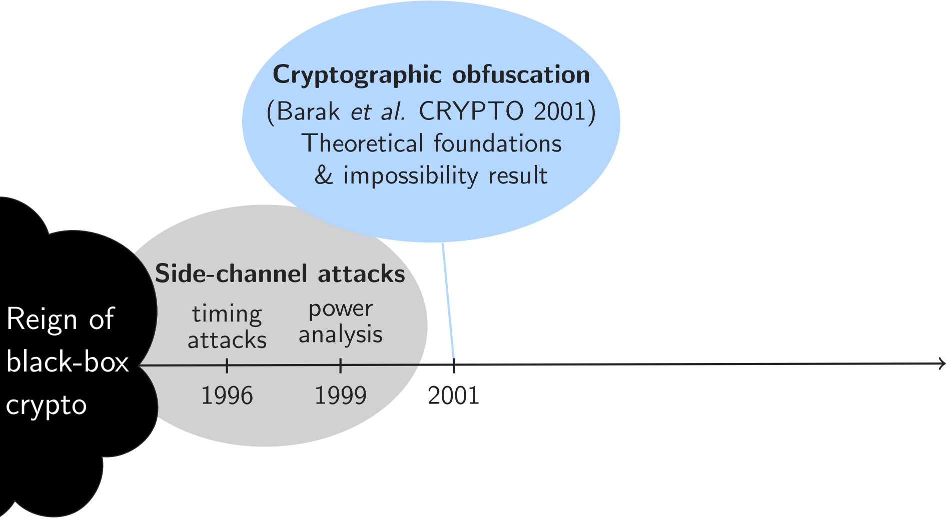
Illustration: `http://www.whiteboxcrypto.com/`

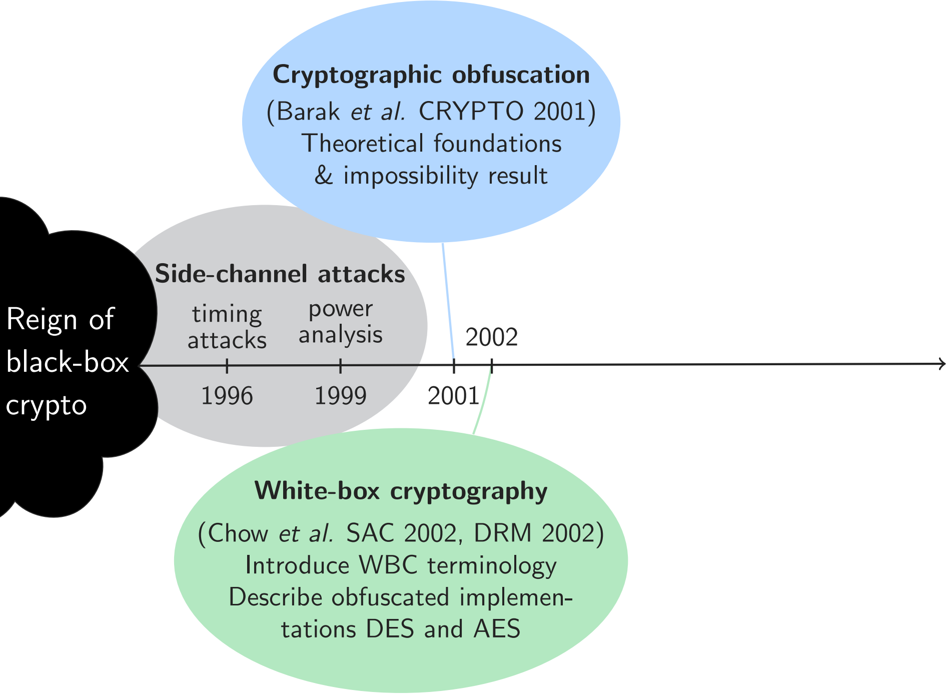
A scientific timeline

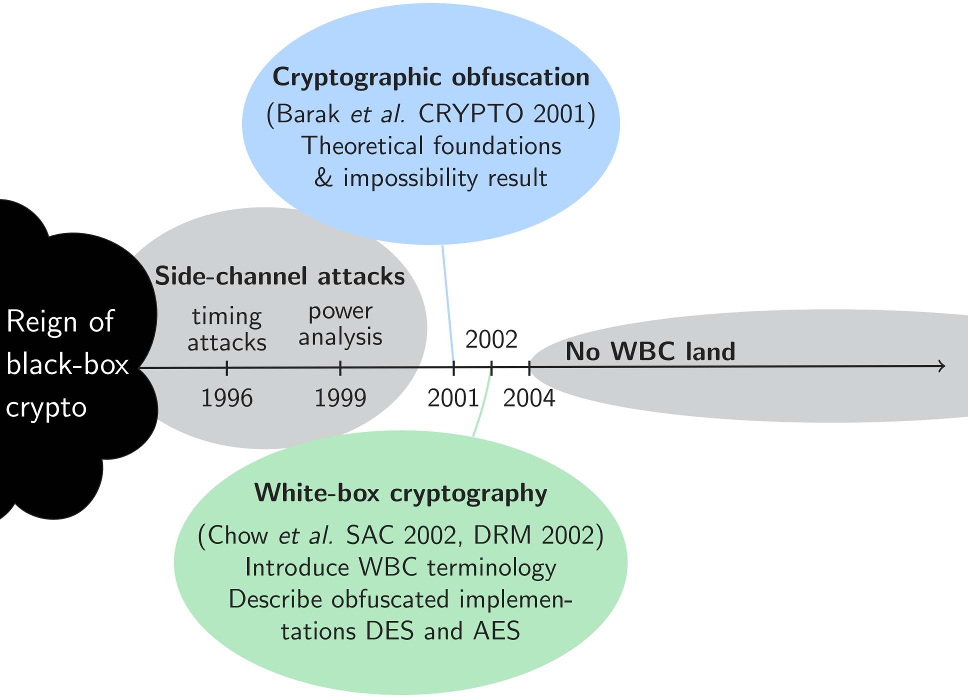


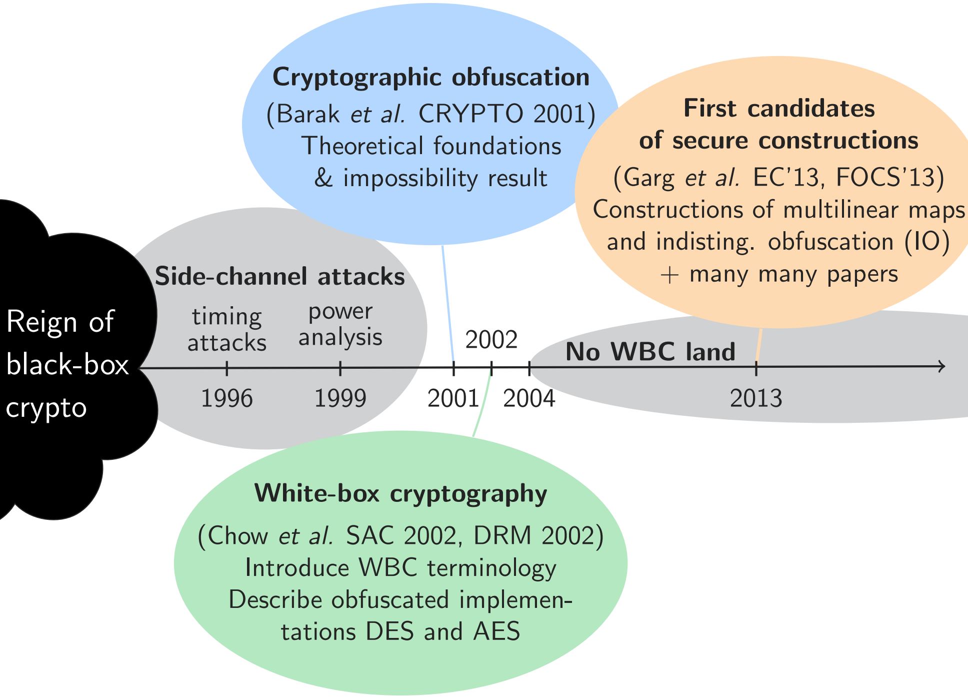
A scientific timeline

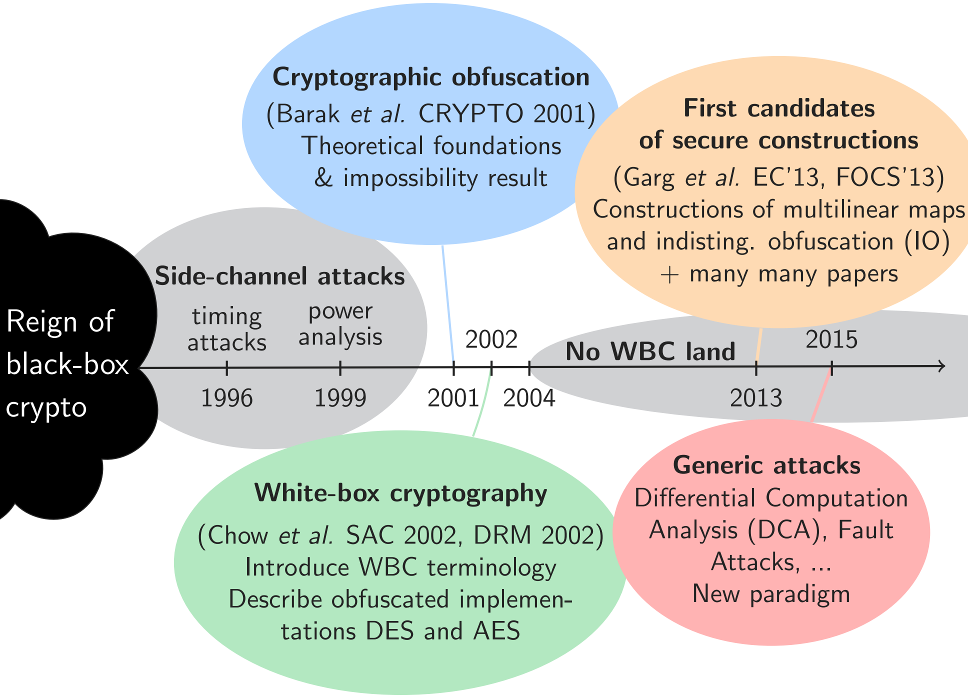


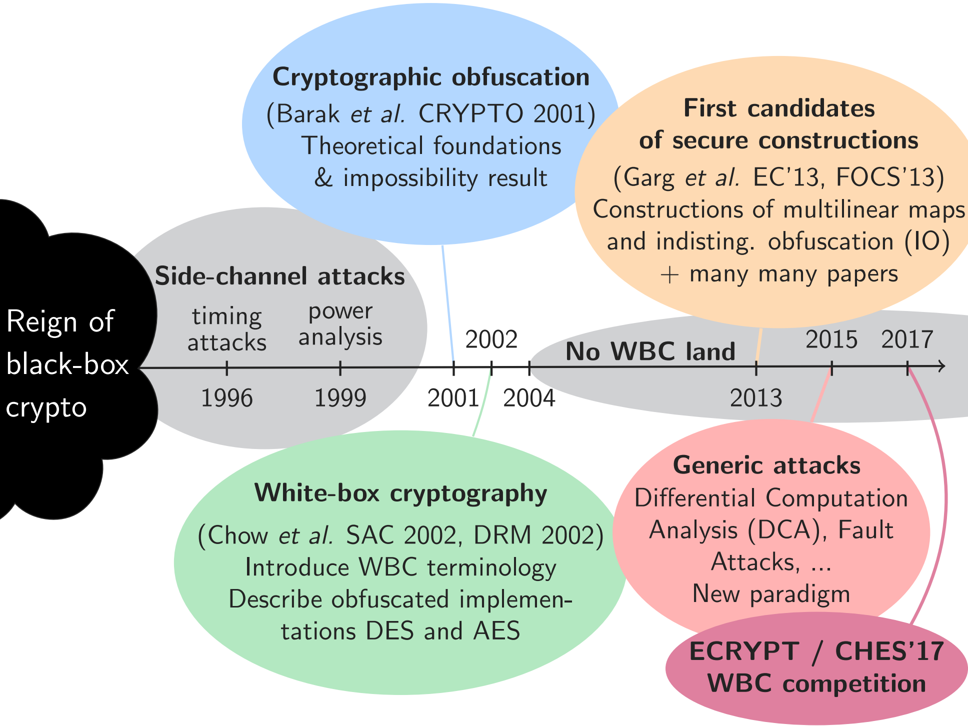

Cryptographic obfuscation
(Barak *et al.* CRYPTO 2001)
Theoretical foundations
& impossibility result
Side-channel attacks
timing attacks
power analysis
Reign of black-box crypto
1996
1999
2001


Cryptographic obfuscation
(Barak *et al.* CRYPTO 2001)
Theoretical foundations
& impossibility result
Side-channel attacks
timing attacks
power analysis
Reign of black-box crypto
1996
1999
2001
2002
White-box cryptography
(Chow *et al.* SAC 2002, DRM 2002)
Introduce WBC terminology
Describe obfuscated implementations DES and AES


Cryptographic obfuscation
(Barak et al. CRYPTO 2001)
Theoretical foundations
& impossibility result
Side-channel attacks
timing attacks
power analysis
Reign of black-box crypto
1996
1999
2001
2002
2004
No WBC land
White-box cryptography
(Chow et al. SAC 2002, DRM 2002)
Introduce WBC terminology
Describe obfuscated implementations DES and AES


Cryptographic obfuscation
(Barak et al. CRYPTO 2001)
Theoretical foundations
& impossibility result
First candidates
of secure constructions
(Garg et al. EC'13, FOCS'13)
Constructions of multilinear maps
and indisting. obfuscation (IO)
+ many many papers
Reign of
black-box
crypto
Side-channel attacks
timing
attacks
power
analysis
2002
No WBC land
1996
1999
2001
2004
2013
White-box cryptography
(Chow et al. SAC 2002, DRM 2002)
Introduce WBC terminology
Describe obfuscated implemen-
tations DES and AES


Reign of black-box crypto
Side-channel attacks
timing attacks
power analysis
1996
1999
2001
2002
2004
2013
2015
No WBC land
Cryptographic obfuscation
(Barak et al. CRYPTO 2001)
Theoretical foundations
& impossibility result
First candidates of secure constructions
(Garg et al. EC'13, FOCS'13)
Constructions of multilinear maps and indisting. obfuscation (IO)
+ many many papers
White-box cryptography
(Chow et al. SAC 2002, DRM 2002)
Introduce WBC terminology
Describe obfuscated implementations DES and AES
Generic attacks
Differential Computation Analysis (DCA), Fault Attacks, ...
New paradigm


Cryptographic obfuscation
(Barak et al. CRYPTO 2001)
Theoretical foundations
& impossibility result
First candidates
of secure constructions
(Garg et al. EC'13, FOCS'13)
Constructions of multilinear maps
and indisting. obfuscation (IO)
+ many many papers
Side-channel attacks
timing
attacks
power
analysis
Reign of
black-box
crypto
2002
No WBC land
2015
2017
1996
1999
2001
2004
2013
White-box cryptography
(Chow et al. SAC 2002, DRM 2002)
Introduce WBC terminology
Describe obfuscated implemen-
tations DES and AES
Generic attacks
Differential Computation
Analysis (DCA), Fault
Attacks, ...
New paradigm
ECRYPT / CHES'17
WBC competition

Overview of this talk

- White-box crypto theory
  - Formal definitions & security notions
- White-box crypto practice
  - Practical constructions & attacks
- White-box crypto competition
  - Wrap-up, break of challenge 777

White-Box Crypto Theory

What is a program?

- A word in a formal language $P \in \mathcal{L}$

$$\begin{aligned} \text{execute}: \mathcal{L} \times \{0,1\}^* &\to \{0,1\}^* \\ (P, input) &\mapsto output \end{aligned}$$

(Universal Turing Machine)

- $|P|$: size of $P \in \mathcal{L}$
- time(P): # operations for execute$(P, \cdot)$

What is a program?

- $P \equiv f$ *(P implements f)*

$$\forall x \ : \ \text{execute}(P, x) = f(x)$$

- $P_1 \equiv P_2$ *(functional equivalence)*

$$\forall x \ : \ \text{execute}(P_1, x) = \text{execute}(P_2, x)$$

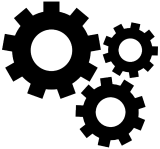
- Straight-line programs
  - no conditional statements, no loops
  - $|P| = \text{time}(P)$

What is an obfuscator?

- An algorithm:

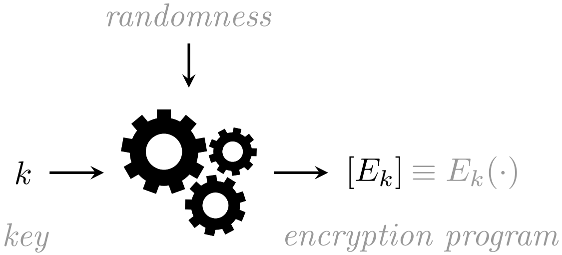
$$randomness$$

$$\downarrow$$

$P \longrightarrow$  $\longrightarrow O(P) \equiv P$

- Size and execution time increase
(hopefully not too much)

What is a white-box compiler?

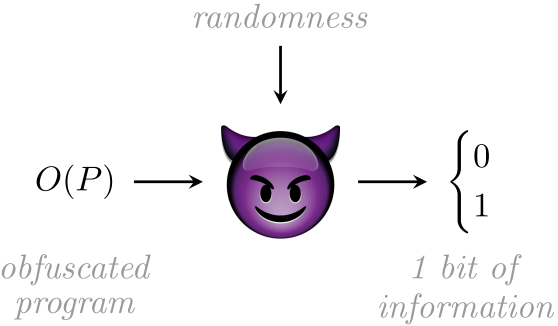


- Specific to an encryption function E
- Can be constructed from an obfuscator

$$k \to P \equiv E_k(\cdot) \xrightarrow{O} [E_k]$$

What is an adversary?

- An algorithm:



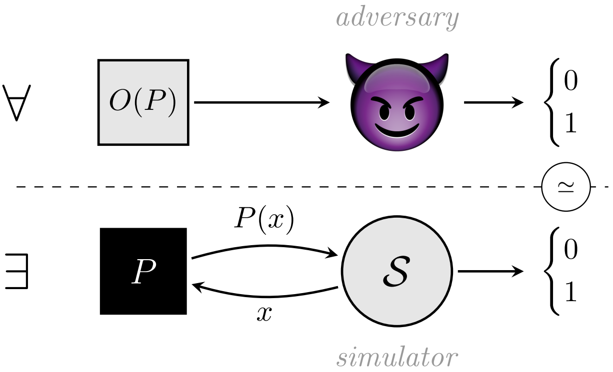
- Ex: msb of k if $P \equiv \mathrm{AES}_k(\cdot)$
- Wlg: $\nexists$ 1-bit ♉ $\Rightarrow$ $\nexists$ multi-bit ♉

[Barak *et al.* – CRYPTO 2001]

On the (Im)possibility of Obfuscating Programs

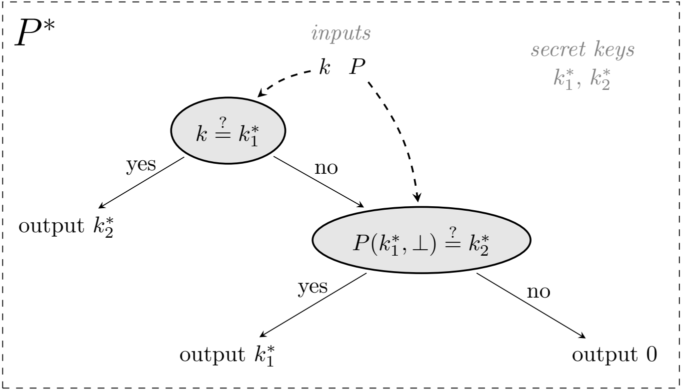
- Virtual Black Box (VBB) security notion
- Impossibility result: VBB cannot be achieved for all programs (counterexample)
- Indistinguishability Obfuscation (IO)

VBB security notion



- $O(P)$ reveals nothing more than the I/O behavior of P

Impossibility result

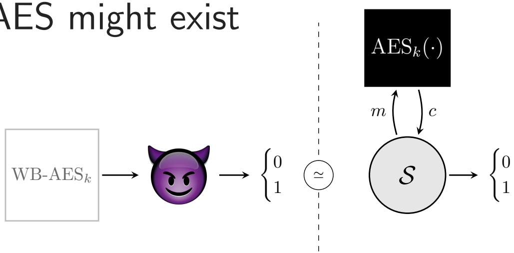


P^* cannot be VBB obfuscated:

- BB access to P^* reveals nothing
- But $O(P^*)\big(0, O(P^*)\big) = k_1^*$

The good news

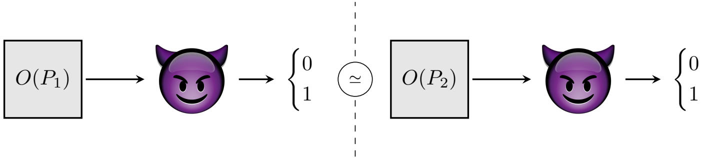
- The impossibility result does not apply to a given encryption algorithm
- VBB AES might exist



- The bad news: seems very hard to achieve

Indistinguishability Obfuscation (IO)

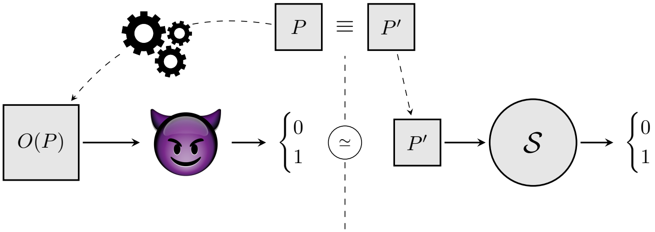
- Notion restricted to straight-line programs
- For any (P_1, P_2) st $P_1 \equiv P_2$ and $|P_1| = |P_2|$



- *i.e.* $O(P_1)$ and $O(P_2)$ are indistinguishable

Why is IO meaningful?

- IO $\Leftrightarrow$ Best Possible Obfuscation
- For any P':



- $O(P)$ doesn't reveal anything more than the *best obfuscated* program P'

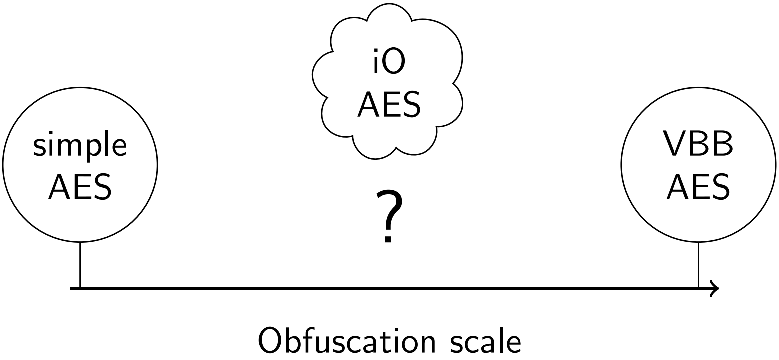
Is IO meaningful for WBC?

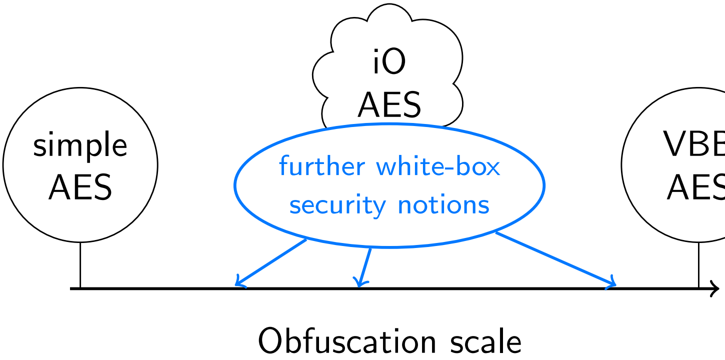
- IO does not imply resistance to key extraction
- For instance

$$\text{Any prog } P \equiv \mathsf{AES}_k(\cdot) \longmapsto \text{Ref implem of } \mathsf{AES}_k(\cdot)$$

- Nevertheless

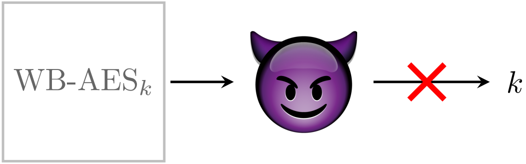
$$\exists P^* \equiv \mathsf{AES}_k(\cdot) \text{ secure}$$
$$\Rightarrow$$
$$\forall P \equiv \mathsf{AES}_k(\cdot) \text{ with } |P| \geq |P^*|\text{: } IO(P) \text{ secure}$$


iO
AES
simple
AES
?
VBB
AES
Obfuscation scale


iO
AES
simple
AES
VBB
AES
further white-box
security notions
Obfuscation scale

White-box security notions

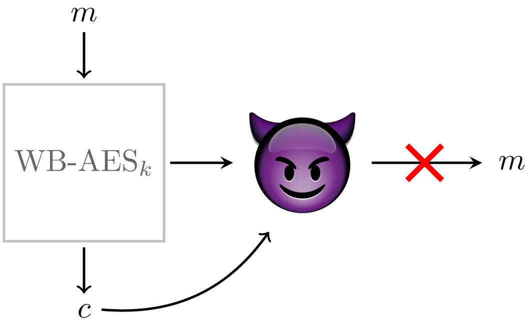
- **Unbreakability**: resistance to key extraction



- Basic requirement but insufficient in practice
- Other security notions
  - [SWP09] *Towards Security Notions for White-Box Cryptography* (ISC 2009)
  - [DLPR13] *White-Box Security Notions for Symmetric Encryption Schemes* (SAC 2013)

One-wayness

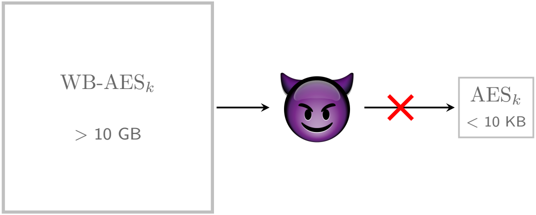
- **One-wayness**: hardness of inversion



- Turns AES into a public-key cryptosystem
- PK crypto with light-weight private operations

Incompressibility

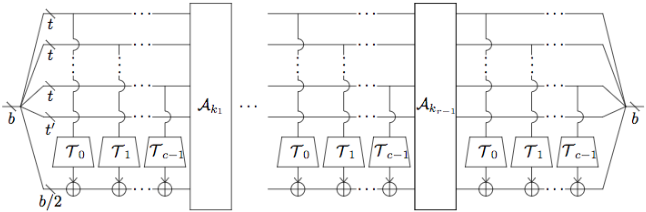
- **Incompressibility**: hardness of compression



- Makes the implementation less convenient to share at a large scale

Incompressibility

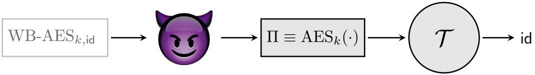
- Incompressible primitives recently proposed
  - Bogdanov *et al.* (CCS 2015, Asiacrypt 2016)
  - Fouque *et al.* (Asiacrypt 2016)



- But no white-box implementations of a standard cipher (*e.g.* AES)

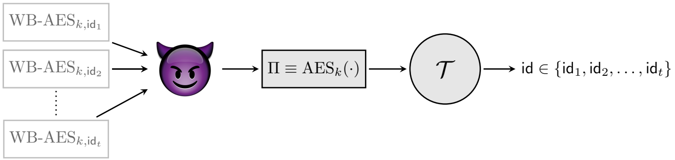
Security features

- **Traceability**: WB implem traceable



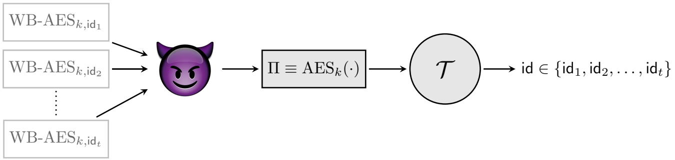
Security features

- **Traceability**: WB implem traceable

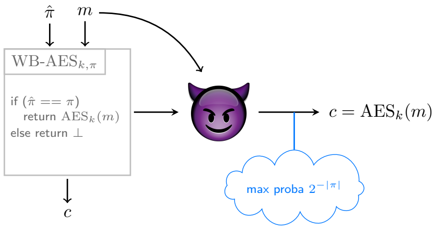


Security features

- **Traceability**: WB implem traceable



- **Password**: WB implem locked by password



Some relations

- [DLPR13] Perturbation-Value Hiding notion:

$$\text{PVH} \Rightarrow \text{traceability}$$

- If the underlying encryption scheme is secure:

$$\begin{array}{ccccc} & & \text{INC} & & \\ & & \Downarrow & & \\ \text{OW} & \Rightarrow & \text{UBK} & \Leftarrow & \text{PVH} \end{array}$$

Some relations

- [DLPR13] Perturbation-Value Hiding notion:

$$\text{PVH} \Rightarrow \text{traceability}$$

- If the underlying encryption scheme is secure:

$$\begin{array}{ccccccccc} & & & & \text{INC} & & & & \\ & & & & \Downarrow & & & & \\ \text{VBB} & \Rightarrow & \text{OW} & \Rightarrow & \text{UBK} & \Leftarrow & \text{PVH} & \Leftarrow & \text{VBB} \end{array}$$

Some relations

- [DLPR13] Perturbation-Value Hiding notion:

$$\text{PVH} \Rightarrow \text{traceability}$$

- If the underlying encryption scheme is secure:

$$\begin{array}{ccccccccc} & & & & \text{VBB} & & & & \\ & & & & \not\Downarrow & & & & \\ & & & & \text{INC} & & & & \\ & & & & \Downarrow & & & & \\ \text{VBB} & \Rightarrow & \text{OW} & \Rightarrow & \text{UBK} & \Leftarrow & \text{PVH} & \Leftarrow & \text{VBB} \end{array}$$

Some relations

- [DLPR13] Perturbation-Value Hiding notion:

$$\text{PVH} \Rightarrow \text{traceability}$$

- If the underlying encryption scheme is secure:

$$\begin{array}{ccccccccc} & & & & \text{VBB} & & & & \\ & & & & \not\Downarrow & & & & \\ & & & & \text{INC} & & & & \\ & & & & \Downarrow & & & & \\ \text{VBB} & \Rightarrow & \text{OW} & \Rightarrow & \text{UBK} & \Leftarrow & \text{PVH} & \Leftarrow & \text{VBB} \end{array}$$

- No UBK construction known for AES

  $\Rightarrow$ no OW/INC/PVH/VBB construction neither

IO constructions

- Very active research field
  - 18 papers in 2017 (IACR conferences)
  - 22 papers in 2016 (IACR conferences)
- Most constructions rely on *multilinear maps*

$$e : (g_1^{e_1}, g_2^{e_2}, \ldots, g_d^{e_d}) \longmapsto g_T^{e_1 \cdot e_2 \cdots e_d}$$

(or noisy variants)

- Many breaks, security still under investigation
- Performances far beyond practical applications

White-Box Crypto Practice

Original white-box AES

- SAC 2002: *"White-Box Cryptography and an AES Implementation"* (Chow *et al.*)
- First step: network of look-up tables
- Each round split in 4 *sub-rounds*

$$(x_0, x_5, x_{10}, x_{15}) \mapsto \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \otimes \begin{pmatrix} S(x_0 \oplus k_0) \\ S(x_5 \oplus k_5) \\ S(x_{10} \oplus k_{10}) \\ S(x_{15} \oplus k_{15}) \end{pmatrix}$$

Original white-box AES

- Computed as

$$T_0[x_0] \oplus T_5[x_5] \oplus T_{10}[x_{10}] \oplus T_{15}[x_{15}]$$

- Tables T_i : 8 bits → 32 bits

$$\begin{aligned}
T_0[x] &= S(x \oplus k_0) \times (02\ 01\ 01\ 03)^T \\
T_5[x] &= S(x \oplus k_5) \times (03\ 02\ 01\ 01)^T \\
T_{10}[x] &= S(x \oplus k_{10}) \times (01\ 03\ 02\ 01)^T \\
T_{15}[x] &= S(x \oplus k_{15}) \times (01\ 01\ 03\ 02)^T
\end{aligned}$$

- XOR table: 8 bits → 4 bits

$$T_{\text{xor}}[x_0 || x_1] = x_0 \oplus x_1$$

Original white-box AES

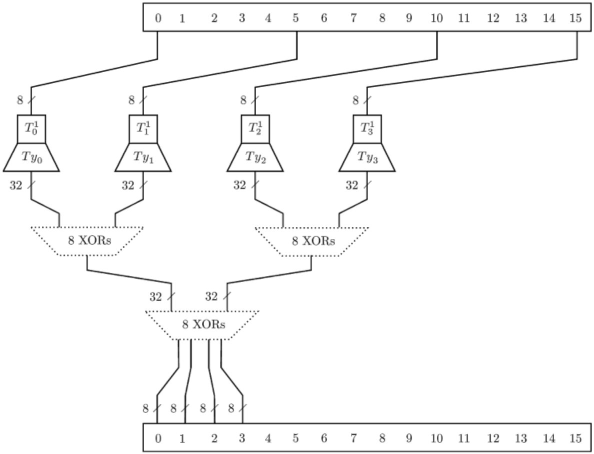


Illustration: J. Muir *"A Tutorial on White-box AES"* (ePrint 2013)

Original white-box AES

- Second step: randomize look-up tables
- Each table T is replaced by

$$T' = g \circ T \circ f^{-1}$$

where f, g are **random encodings**

- For two *connected* tables T, R

$$\begin{aligned} T' &= g \circ T \circ f^{-1} \\ R' &= h \circ R \circ g^{-1} \end{aligned} \quad \Rightarrow \quad R' \circ T' = h \circ (R \circ T) \circ f^{-1}$$

Original white-box AES

- Intuition: encoded tables bring no information
- True for a single (bijective) table $g \circ T \circ f^{-1}$
- Not for the large picture

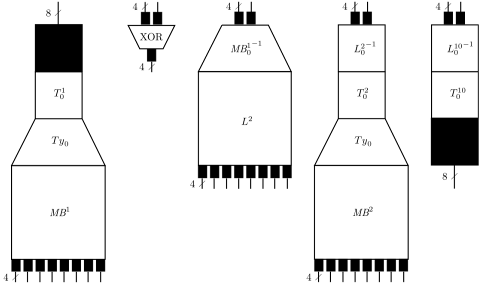
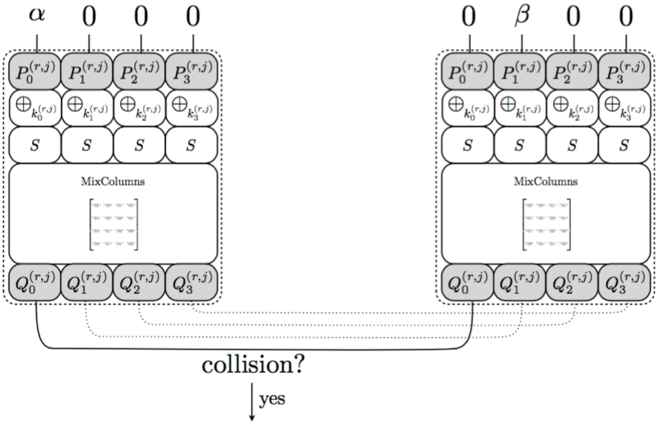


Illustration: J. Muir *"A Tutorial on White-box AES"* (ePrint 2013)

Many breaks

- First break: BGE attack
  - Billet *et al. Cryptanalysis of a White Box AES Implementation* (SAC 2004)
- Generic attack on WB SPN ciphers
  - Michiels *et al. Cryptanalysis of a Generic Class of White-Box Implementations* (SAC 2008)
- Collision attack & improved BGE attack
  - Lepoint *et al. Two Attacks on a White-Box AES Implementation* (SAC 2013)
- Attack complexity ~ 2^{22}

Example: collision attack



$$02 \cdot S_0(\alpha) \oplus 03 \cdot S_1(0) = 02 \cdot S_0(0) \oplus 03 \cdot S_1(\beta)$$

where $S_0(x) = S(P_0(x) \oplus k_0)$ and $S_1(x) = S(P_1(x) \oplus k_1)$

Patches and variants

- Perturbed WB-AES using MV crypto (Bringer *et al.* ePrint 2006)
  ⇒ broken (De Mulder *et al.* INDOCRYPT 2010)

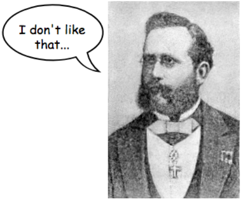
- WB-AES based on wide linear encodings (Xiao-Lai, CSA 2009)
  ⇒ broken (De Mulder *et al.* SAC 2012)

- WB-AES based on dual AES ciphers (Karroumi, ICISC 2010)
  ⇒ broken (Lepoint *et al.* SAC 2013)

- Same situation with DES

Secret design paradigm

- Industrial need
  - home-made solutions
  - mix of several obfuscation techniques
  - secret designs

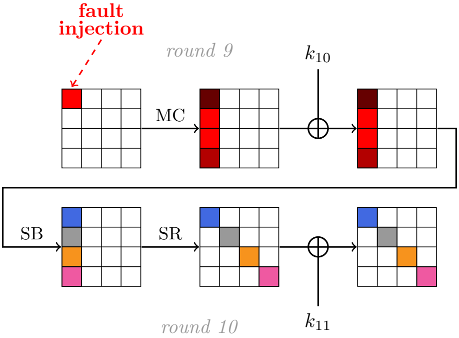


Auguste Kerckhoffs

- Security evaluations by ITSEF labs
- Development of generic attacks
  - Fault attacks, DCA
  - Avoid costly reverse engineering effort

Fault attacks

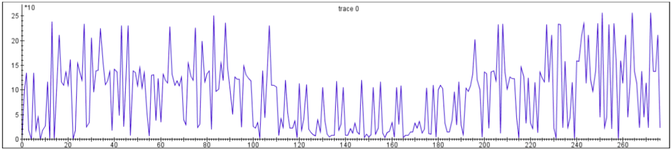
- Easy fault injection in the white-box context
- Plenty of efficient FA techniques (on *e.g.* AES)



- Original white-box AES vulnerable to this attack

Differential Computation Analysis

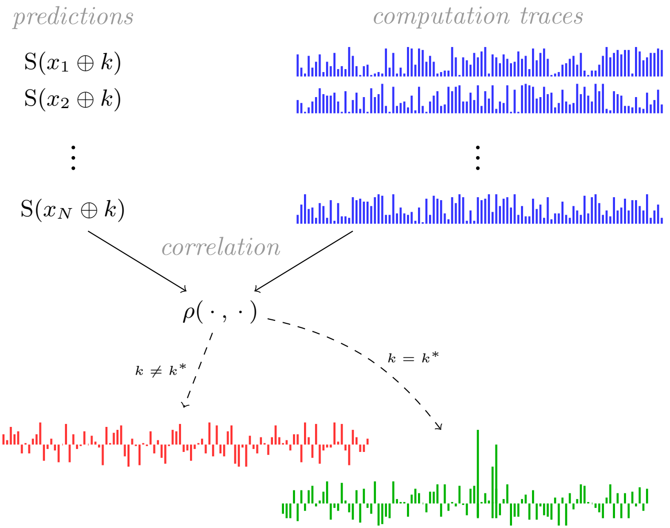
- Suggested by NXP / Riscure
  - Presentation at BalckHat 2015
  - Best paper award CHES 2016
- Record data-dependent information at execution ⇒ *computation trace*



Trace: J. Bos (presentation CHES 2016)

- Apply DPA techniques to computation traces

Differential Computation Analysis



DCA in presence of encodings

- DCA can break the original white-box AES
  - [Bos *et al.* CHES 2016] *Differential Computation Analysis*

- Why?
  - random encodings are hardcoded
  - for some Enc, we might have

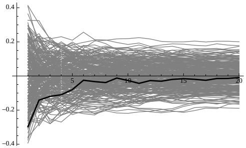
$$\rho(x_i, \mathsf{Enc}(x)_j) \gg 0$$

  - especially with 4-bit encodings

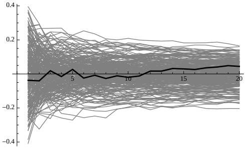
$$\mathsf{Enc}(x_0 \,\|\, x_1) = \mathsf{Enc}(x_0) \,\|\, \mathsf{Enc}(x_1)$$

DCA experiment

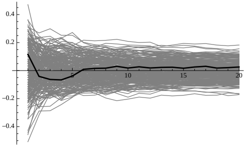
- Random 4-bit encoding Enc
- Correlation $\rho\big(\mathrm{S}(x \oplus k)_0, \, \mathsf{Enc}(\mathrm{S}(x \oplus k^*))_j\big)$



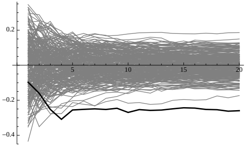
Bit 0



Bit 1



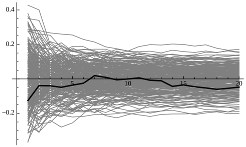
Bit 2



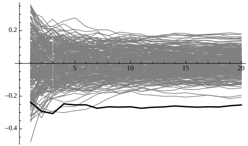
Bit 3

DCA experiment

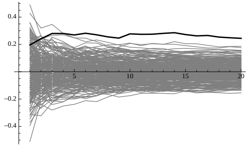
- With another (4-bit) encoding



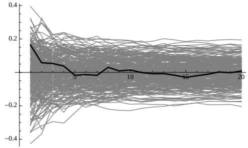
Bit 0



Bit 1



Bit 2

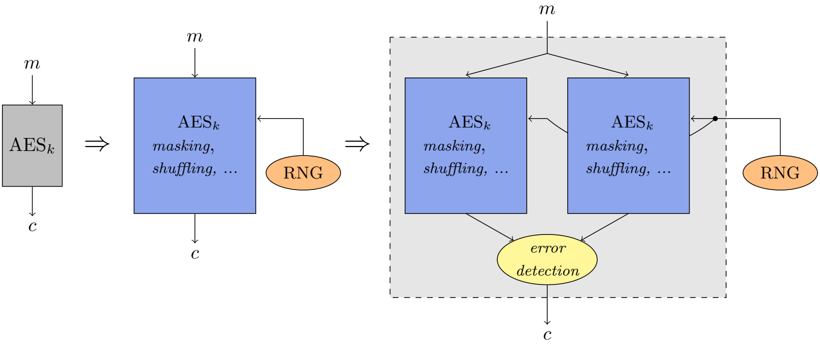


Bit 3

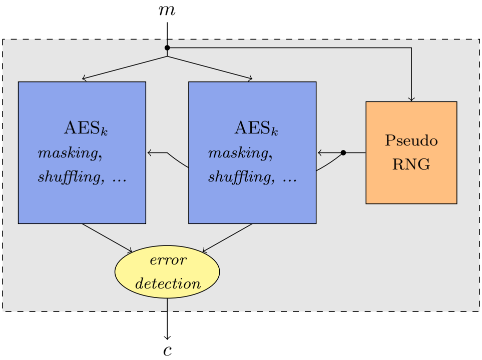
- Most of the time 1, 2, or 3 bits leak

Countermeasures?

- Natural approach: use known SCA/FA countermeasures

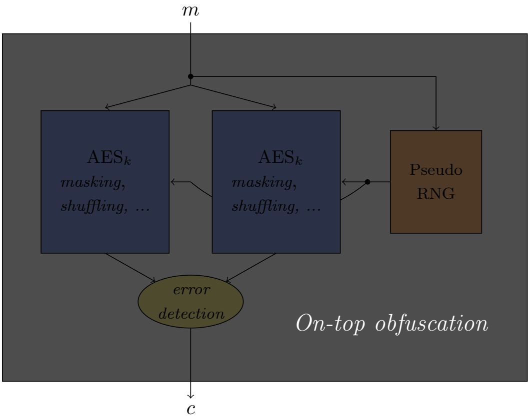


Countermeasures?



- Pseudo-randomness from m
- PRNG should be somehow secret

Countermeasures?



- Countermeasures hard to remove
- P-randomness / redundancy hard to detect

Open problems

- How to obfuscate the countermeasures?
- How to generate pseudo-randomness?
- Resistance to higher-order DCA, multiple FA?

White-Box Crypto Competition


CHES 2017
Capture the Flag
Challenge
The WhibOx Contest
An ECRYPT White-Box Cryptography Competition

WhibOx Contest

- Goal: confront designers and attackers in the **secret design paradigm**
- Designers could submit WB AES implems:
  - C source code $\leq$ 50MB
  - executable $\leq$ 20MB
  - RAM consumption $\leq$ 20MB
  - running time $\leq$ 1sc
- Attackers could try to recover the keys of submitted implems

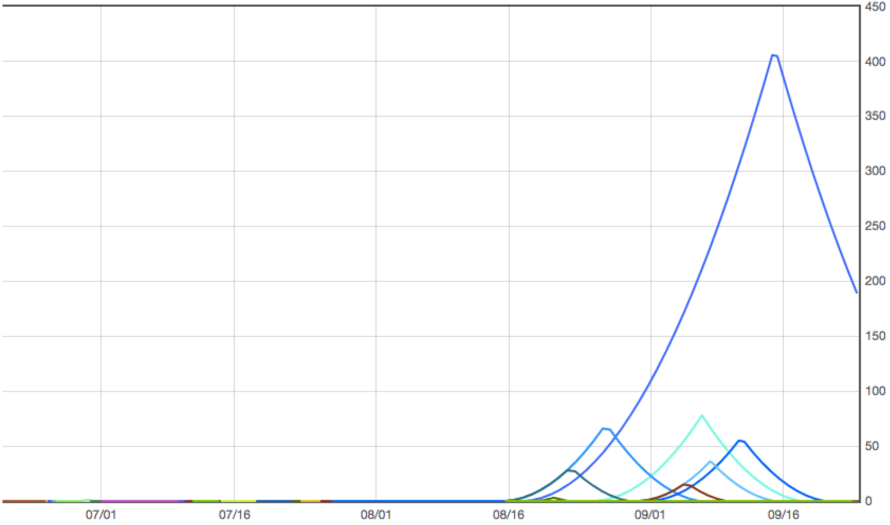
Score system

- Unbroken implem on day n

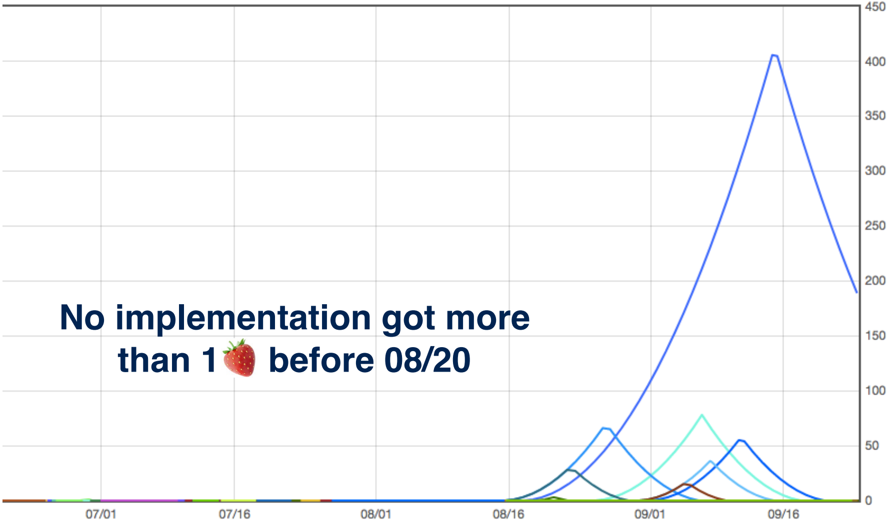
$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}\ \textbf{ST}$$

- Break on day n
  - Designer gets $\frac{n(n+1)}{2}$ **ST** points
  - Attacker gets $\frac{n(n+1)}{2}$ **BN** points
  - Challenge score starts decreasing symmetrically

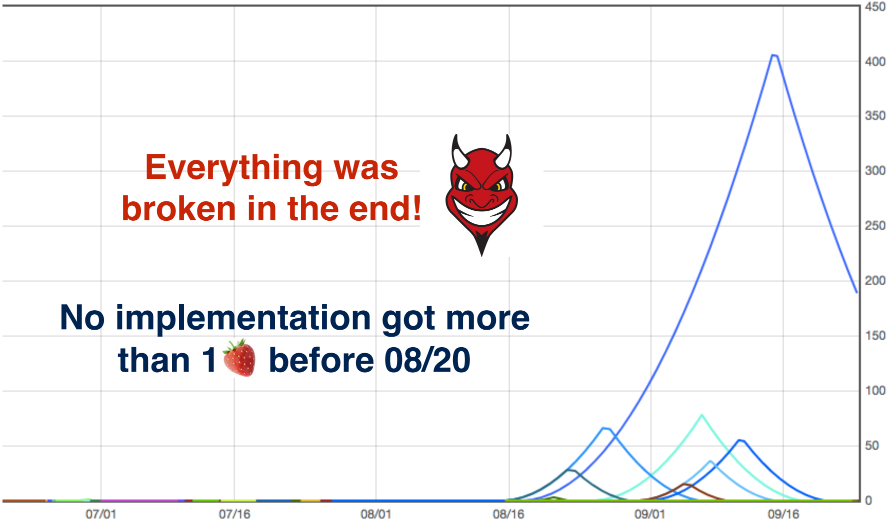
Strawberry scores over time



Strawberry scores over time



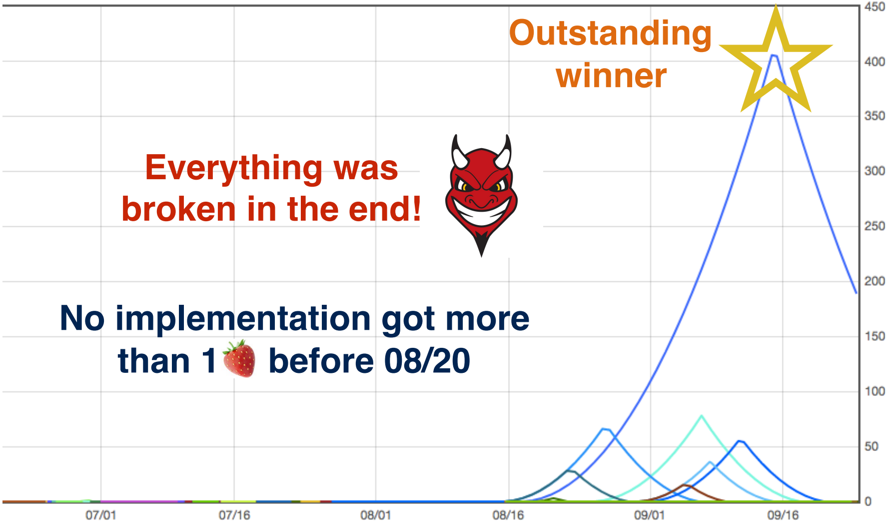
Strawberry scores over time



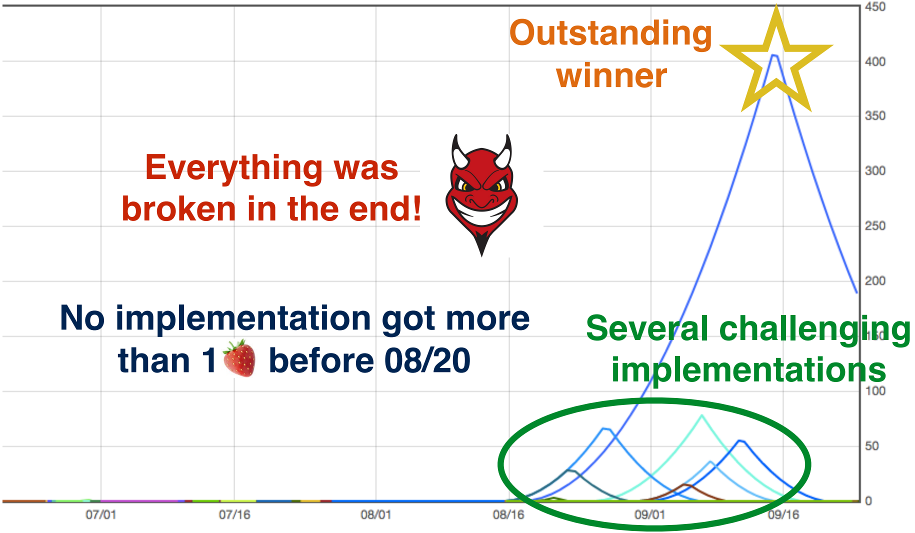
Everything was broken in the end!

No implementation got more than 1🍓 before 08/20

Strawberry scores over time



Strawberry scores over time



Results

- 94 submitted implementations
- ~ 870 breaks
- Socreboard:

id	designer	breaker	score	# days	# breaks
777	cryptolux	team_cryptoexperts	406	28	1
815	grothendieck	cryptolux	78	12	1
753	sebastien-riou	cryptolux	66	11	3
877	chaes	You!	55	10	2
845	team4	cryptolux	36	8	2

cryptolux: Biryukov, Udovenko

team_cryptoexperts: Goubin, Paillier, Rivain, Wang

Implementation 777

- Several obfuscation layers
  - Encoded Boolean circuit
  - Bitslicing, error detection, dummy operations
  - Virtualization, naming obfuscation
- Code size: 28 MB
- Code lines: 2.3 K
- 12 global variables
  - `pDeoW`: computation state (2.1 MB)
  - `JGNNvi`: program bytecode (15.3 MB)

Implementation 777

- 1020 functions of the form

```
void xSnEq (uint UMNsVLp, uint KtFY, uint vzJZq) {
  if (nIlajqq () == IFWBUN (UMNsVLp, KtFY))
    EWwon (vzJZq);
}

void rNUiPyD (uint hFqeIO, uint jvXpt) {
  xkpRp[hFqeIO] = MXRIWZQ (jvXpt);
}

void cQnB (uint QRFOf, uint CoCiI, uint aLPxnn) {
  ooGoRv[(kIKfgI + QRFOf) & 97603] =
    ooGoRv[(kIKfgI + CoCiI) | 173937] & ooGoRv[(kIKfgI + aLPxnn) | 39896];
}

uint dLJT (uint RouDUC, uint TSCaTl) {
  return ooGoRv[763216 ul] | qscwtK (RouDUC + (kIKfgI << 17), TSCaTl);
}
```

Analysis of functions

- Table of function pointers indexed by bytecode
- Only 210 functions are called (over 1020)
- Ducplicates of 21 different functions
    - memory reading/writing
    - bitwise operations, bit shifts
    - goto, conditional jump

De-virtualisation

```
PROGRAM  = ...  // bytecode 15.3 MB
FUNC_PTR = ...  // 210 function pointers

interpretor()
{
    pc = 0;
    while(pc < eop)
    {
        nb_arg = PROGRAM[pc]; pc++;
        func_index = PROGRAM[pc]; pc++;
        function = FUNC_PTR[func_index];
        for (i=0; i<nb_arg; i++)
        {
            arg[i] = PROGRAM[pc]; pc++;
        }
        function(arg[0], ...);
    }
}
```

Simulation ⇒ equivalent program with do-while loops
of arithmetic instructions

Human reverse engineering

- Remove some dummy loops
- Get sequence of 64-loops of 64-bit instructions
  - First part: 64×64 bitslice program
  - 3 instances with the input plaintext
  - rest with hardcoded values
  - Second part: (probably) error detection and extraction of the ciphertext
- Extract a Boolean circuit with $\sim$ **600K gates**

SSA form

- Put it in Static Single Assignment (SSA) form:

$$
\begin{array}{lcl}
x & = & \ldots \\
y & = & \ldots \\
t & = & \neg x \\
x & = & t \oplus y \\
y & = & y \wedge t \\
t & = & x \vee y \\
 & \vdots &
\end{array}
\quad \Rightarrow \quad
\begin{array}{lcl}
x & = & \ldots \\
y & = & \ldots \\
t_1 & = & \neg x \\
t_2 & = & t_1 \oplus y \\
t_3 & = & y \wedge t_1 \\
t_4 & = & t_2 \vee t_3 \\
 & \vdots &
\end{array}
$$

Circuit minimization

Detect (over many executions) and remove:

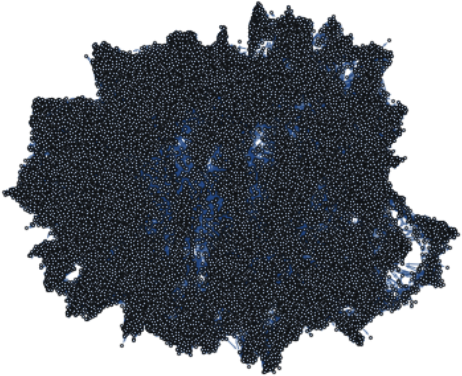
- Dummy variable: t_i never used?
- Constant: $t_i = 0$? $t_1 = 1$?
- Duplicate: $t_i = t_j$?
- Pseudo-randomness:

$$(t_i \to t_i \oplus 1) \;\Rightarrow\; \text{same result?}$$

- Several rounds: ∼600K ⇒ ∼**280K** gates

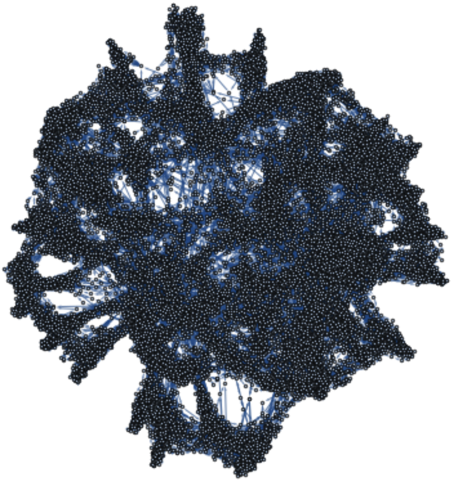
Data dependency analysis

Data dependency graph (20% of the circuit):



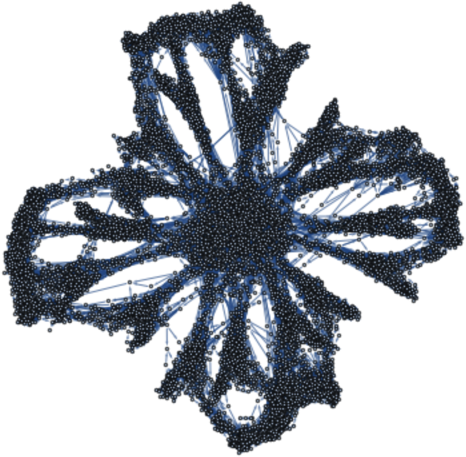
Data dependency analysis

Data dependency graph (10% of the circuit):



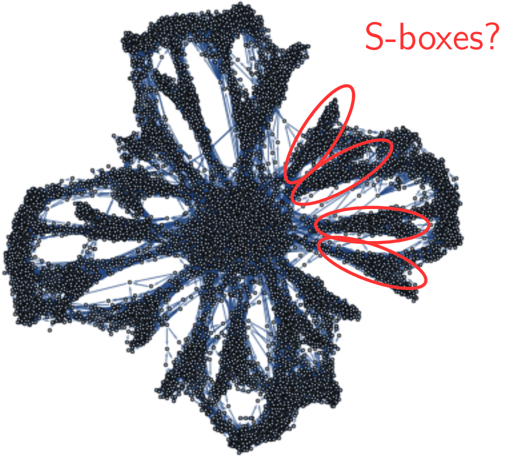
Data dependency analysis

Data dependency graph (5% of the circuit):



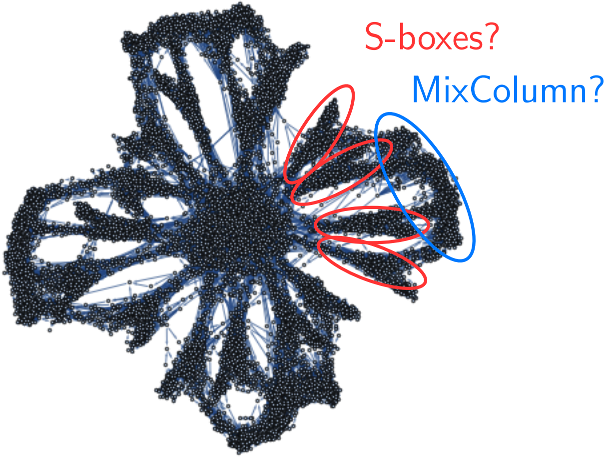
Data dependency analysis

Data dependency graph (5% of the circuit):



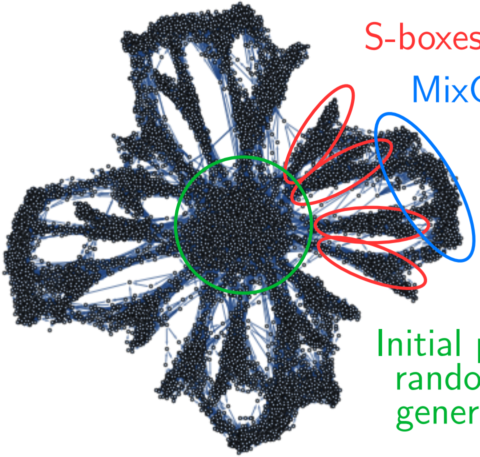
Data dependency analysis

Data dependency graph (5% of the circuit):



Data dependency analysis

Data dependency graph (5% of the circuit):



Data dependency analysis

- Cluster analysis $\Rightarrow$ gates within one "s-box"
- Identify all the outgoing variables:

$$s_1, s_2, \ldots, s_n$$

- Likely hypothesis:

$$S(x \oplus k^*) = \mathsf{Dec}(s_1, s_2, \ldots, s_m)$$

for some deterministic decoding function

Key recovery

- Hypothesis: linear decoding function
- Record the s_i's over n executions

$$\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & s_m^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ s_1^{(n)} & s_2^{(n)} & \cdots & s_m^{(n)} \end{bmatrix} \qquad \begin{bmatrix} S_j(x^{(1)} \oplus k) \\ S_j(x^{(2)} \oplus k) \\ \vdots \\ S_j(x^{(n)} \oplus k) \end{bmatrix},$$

Key recovery

- Hypothesis: linear decoding function
- Record the s_i's over n executions

$$\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & s_m^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ s_1^{(n)} & s_2^{(n)} & \cdots & s_m^{(n)} \end{bmatrix} \qquad \begin{bmatrix} S_j(x^{(1)} \oplus k) \\ S_j(x^{(2)} \oplus k) \\ \vdots \\ S_j(x^{(n)} \oplus k) \end{bmatrix},$$

Key recovery

- Hypothesis: linear decoding function
- Record the s_i's over n executions

$$\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & s_m^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ s_1^{(n)} & s_2^{(n)} & \cdots & s_m^{(n)} \end{bmatrix} \qquad \begin{bmatrix} S_j(x^{(1)} \oplus k) \\ S_j(x^{(2)} \oplus k) \\ \vdots \\ S_j(x^{(n)} \oplus k) \end{bmatrix},$$

Key recovery

- Hypothesis: linear decoding function
- Record the s_i's over n executions

$$\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & s_m^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ s_1^{(n)} & s_2^{(n)} & \cdots & s_m^{(n)} \end{bmatrix} \qquad \begin{bmatrix} S_j(x^{(1)} \oplus k) \\ S_j(x^{(2)} \oplus k) \\ \vdots \\ S_j(x^{(n)} \oplus k) \end{bmatrix},$$

Key recovery

- Hypothesis: linear decoding function
- Record the s_i's over n executions

$$\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & s_m^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ s_1^{(n)} & s_2^{(n)} & \cdots & s_m^{(n)} \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} S_j(x^{(1)} \oplus k) \\ S_j(x^{(2)} \oplus k) \\ \vdots \\ S_j(x^{(n)} \oplus k) \end{bmatrix},$$

- Linear system solvable for $k = k^*$

Key recovery

- And it works! For example:
  - s-box cluster with $n = 34$ outgoing variables
  - using $T = 50$ executions traces
  - one solution per S_j for $k = k^*$
  - no solutions for $k \neq k^*$

$j = 0$: 0,0,0,0,0,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 1$: 0,0,0,0,0,0,1,0,0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 2$: 0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 3$: 0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 4$: 0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 5$: 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 6$: 0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 7$: 0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Key recovery

- And it works! For example:
  - s-box cluster with $n = 34$ outgoing variables
  - using $T = 50$ executions traces
  - one solution per S_j for $k = k^*$
  - no solutions for $k \neq k^*$

j = 0: 0,0,0,0,0,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
j = 1: 0,0,0,0,0,0,1,0,0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
j = 2: 0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
j = 3: 0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
j = 4: 0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
j = 5: 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
j = 6: 0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
j = 7: 0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Key recovery

- And it works! For example:
  - s-box cluster with $n = 34$ outgoing variables
  - using $T = 50$ executions traces
  - one solution per S_j for $k = k^*$
  - no solutions for $k \neq k^*$

$j = 0$: 0,0,0,0,0,0,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 1$: 0,0,0,0,0,0,0,1,0,0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 2$: 0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 3$: 0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 4$: 0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 5$: 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 6$: 0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 7$: 0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

- Decoding

$$\underbrace{(s_7, s_8, \ldots, s_{21})}_{\text{15 outgoing bits}} \xrightarrow{\times \text{ Bin. Mat.}} \underbrace{(S_0(x \oplus k^*), \ldots, S_7(x \oplus k^*))}_{\text{8 s-box coordinates}}$$

Key recovery

- And it works! For example:
  - s-box cluster with $n = 34$ outgoing variables
  - using $T = 50$ executions traces
  - one solution per S_j for $k = k^*$
  - no solutions for $k \neq k^*$

$j = 0$: 0,0,0,0,0,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 1$: 0,0,0,0,0,0,1,0,0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 2$: 0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 3$: 0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 4$: 0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 5$: 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 6$: 0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
$j = 7$: 0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

- Decoding

$$\underbrace{(s_7, s_8, \ldots, s_{21})}_{\text{15 outgoing bits}} \xrightarrow{\times \text{ Bin. Mat.}} \underbrace{(S_0(x \oplus k^*), \ldots, S_7(x \oplus k^*))}_{\text{8 s-box coordinates}}$$

Conclusion

- Theory:
  - No provably secure constructions
  - More work needed on security models & notions
- Practice:
  - Everything broken in the literature
  - Moving toward a secret design paradigm
  - More work needed on generic attacks and countermeasures in the white-box context
- ECRYPT / CHES'17 competition:
  - Nothing stood > 28 days
  - Can obscurity really bring (a bit of) security?