White-Box Cryptography

Matthieu Rivain

CARDIS 2017
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)

![Image of a digital identity card](image-url)
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
  ▶ users themselves
  ▶ co-hosted applications
  ▶ ...

■ White-box adversary model
  ▶ analyse the code
  ▶ tamper with execution
  ▶ access the memory
  ▶ ...

■ Ex: scan the memory for secret keys

White-box cryptography

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

Illustration: http://www.whiteboxcrypto.com/
A scientific timeline

Reign of black-box crypto
A scientific timeline

Reign of black-box crypto

Side-channel attacks
- timing attacks
- power analysis

1996 1999
Side-channel attacks

Reign of black-box crypto

1996 1999 2001

timing attacks

power analysis

Cryptographic obfuscation
(Barak et al. CRYPTO 2001)
Theoretical foundations & impossibility result
Side-channel attacks

1996 - timing attacks
1999 - power analysis
2001

Cryptographic obfuscation
(Barak et al. CRYPTO 2001)
Theoretical foundations & impossibility result

Reign of black-box crypto

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

1996 1999 2001 2002

White-box cryptography (Chow et al. SAC 2002, DRM 2002)
- Introduce WBC terminology
- Describe obfuscated implementations DES and AES

No WBC land

Reign of black-box crypto

2004
Side-channel attacks
- Timing attacks (1996)
- Power analysis (1999)

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 indistinguishable obfuscation (IO)
  + Many many papers

Reign of black-box crypto

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

Side-channel attacks
- timing attacks
- power analysis

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

Reign of black-box crypto

No WBC land

**Side-channel attacks**

- **1996**: timing attacks
- **1999**: power analysis

**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 indistinguishable 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

**Reign of black-box crypto**

**No WBC land**

- 2002
- 2004
- 2013
- 2015
- 2017

**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 $P \in \mathcal{L}$

execute : $\mathcal{L} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$

$(P, \text{input}) \mapsto \text{output}$

(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:

\[
\text{randomness} \quad \downarrow \quad O(P) \equiv P
\]

- 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

\[ k \rightarrow P \equiv E_k(\cdot) \xrightarrow{O} [E_k] \]
What is an adversary?

- An algorithm:

  \[ O(P) \rightarrow \text{randomness} \rightarrow \{0, 1\} \]

  \[ \text{obfuscated program} \rightarrow \text{1 bit of information} \]

  Ex: msb of \( k \) if \( P \equiv \text{AES}_k(\cdot) \)

  Wlg: \( \not\exists \) 1-bit \( \bullet \) \( \Rightarrow \not\exists \) multi-bit \( \bullet \)
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

\[ \forall O(P) \text{ adversary} \rightarrow \{0, 1\} \]

\[ \exists P S \text{ simulator} \leftarrow \{0, 1\} \]

- \( O(P) \) reveals nothing more than the I/O behavior of \( P \)
Impossibility result

\[ P^* \text{ inputs secret keys } k^* \]
\[ = k^*_1 \]
\[ P(k^*_1, \bot) = k^*_2 \]

\( P^* \) cannot be VBB obfuscated:
- BB access to \( P^* \) reveals nothing
- But \( O(P^*)(0, O(P^*)) = k^*_1 \)
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 \(|P_1| = |P_2|\)

\[
\begin{align*}
O(P_1) &\rightarrow \text{emoji} \rightarrow \{0, 1\} \cong O(P_2) \\
&\rightarrow \text{emoji} \rightarrow \{0, 1\}
\end{align*}
\]

- i.e. \(O(P_1)\) and \(O(P_2)\) are indistinguishable
Why is IO meaningful?

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

\[
O(P) \rightarrow \{0, 1\} \rightarrow S \rightarrow \{0, 1\}
\]

- $O(P)$ 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

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

- Nevertheless

  \[
  \exists P^* \equiv \text{AES}_k(\cdot) \text{ secure} \\
  \implies \\
  \forall P \equiv \text{AES}_k(\cdot) \text{ with } |P| \geq |P^*|: IO(P) \text{ secure}
  \]
simple AES

AES

iO

AES

VBB AES

Obfuscation scale

further white-box security notions
White-box security notions

- **Unbreakability**: resistance to key extraction
  
  \[ \text{WB-AES}_k \rightarrow \text{Devil} \rightarrow k \]

- Basic requirement but insufficient in practice

- Other security notions
  
  - [SWP09] *Towards Security Notions for White-Box Cryptography* (ISC 2009)
  
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

\[
WB-AES_k, id \xrightarrow{\text{rom}} \Pi \equiv AES_k(\cdot) \xrightarrow{T} \text{id}
\]
Security features

- **Traceability**: WB implem traceable

\[
\text{WB-AES}_{k,\text{id}_1} \quad \text{WB-AES}_{k,\text{id}_2} \quad \ldots \quad \text{WB-AES}_{k,\text{id}_t} \\
\Pi \equiv \text{AES}_k(\cdot) \\
\mathcal{T} \quad \text{id} \in \{\text{id}_1, \text{id}_2, \ldots, \text{id}_t\}
\]
Security features

- **Traceability**: WB implem traceable

  \[
  \text{WB-AES}_{k, id_1} \rightarrow \text{WB-AES}_{k, id_2} \rightarrow \ldots \rightarrow \text{WB-AES}_{k, id_t} \rightarrow \Pi \equiv \text{AES}_k(\cdot) \rightarrow \mathcal{T} \rightarrow \text{id} \in \{id_1, id_2, \ldots, id_t\}
  \]

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

  \[
  \text{WB-AES}_{k, \pi} \rightarrow \hat{\pi} \rightarrow m \rightarrow \text{if } (\hat{\pi} == \pi) \text{ return } \text{AES}_k(m) \text{ else return } \bot \rightarrow c = \text{AES}_k(m) \rightarrow \text{max proba } 2^{-|\pi|}
  \]
Some relations

- [DLPR13] Perturbation-Value Hiding notion:
  \[ \text{PVH} \Rightarrow \text{traceability} \]
- If the underlying encryption scheme is secure:

\[
\begin{align*}
\text{INC} & \Downarrow \\
\text{OW} & \Rightarrow \text{UBK} \\
\text{UBK} & \Leftarrow \text{PVH}
\end{align*}
\]

No UBK construction known for AES \Rightarrow no OW/INC/PVH/VBB construction neither
Some relations

- [DLPR13] Perturbation-Value Hiding notion: PVH $\Rightarrow$ traceability
- If the underlying encryption scheme is secure:

\[
\begin{align*}
\text{INC} & \Downarrow \\
\text{VBB} & \Rightarrow \text{OW} \Rightarrow \text{UBK} \Leftarrow \text{PVH} \Leftarrow \text{VBB}
\end{align*}
\]
Some relations

- [DLPR13] Perturbation-Value Hiding notion:
  \[ \text{PVH} \Rightarrow \text{traceability} \]

- If the underlying encryption scheme is secure:
  \[
  \begin{align*}
  \text{VBB} & \Downarrow \Downarrow \\
  \text{INC} & \\
  \text{VBB} & \Rightarrow \text{OW} \Rightarrow \text{UBK} \leftarrow \text{PVH} \leftarrow \text{VBB}
  \end{align*}
  \]

- No UBK construction known for AES \[\Rightarrow\] no OW/INC/PVH/VBB construction neither
Some relations

- [DLPR13] Perturbation-Value Hiding notion:
  \[ \text{PVH} \Rightarrow \text{traceability} \]

- If the underlying encryption scheme is secure:
  \[ \text{VBB} \quad \uparrow \quad \text{INC} \quad \downarrow \quad \text{VBB} \]
  \[ \text{VBB} \Rightarrow \text{OW} \Rightarrow \text{UBK} \Leftarrow \text{PVH} \Leftarrow \text{VBB} \]

- No UBK construction known for AES
  \[ \Rightarrow \quad \text{no OW/INC/PVH/VBB construction neither} \]
IO constructions

- Very active research field
  - 18 papers in 2017 (IACR conferences)
  - 22 papers in 2016 (IACR conferences)

- Most constructions rely on multilinear maps

\[ e : \left( g_1^{e_1}, g_2^{e_2}, \ldots, g_d^{e_d} \right) \mapsto g_T^{e_1 \cdot e_2 \cdots e_d} \]

(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 \( \rightarrow \) 32 bits

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

- XOR table: 8 bits \( \rightarrow \) 4 bits

\[ T_{xor}[x_0||x_1] = x_0 \oplus x_1 \]
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 $T$, $R$
  
  $$T' = g \circ T \circ f^{-1}$$
  $$R' = h \circ R \circ g^{-1} \quad \Rightarrow \quad R' \circ T' = h \circ (R \circ T) \circ f^{-1}$$
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 $\sim 2^{22}$
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

- 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

- Apply DPA techniques to computation traces
Differential Computation Analysis

predictions

\[ S(x_1 \oplus k) \]
\[ S(x_2 \oplus k) \]
\[ \vdots \]
\[ S(x_N \oplus k) \]

correlation

\[ \rho(\cdot, \cdot) \]

computation traces

\[ k \neq k^* \]
\[ k = k^* \]
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
    \[ \rho(x_i, Enc(x)_j) \gg 0 \]
  - especially with 4-bit encodings
    \[ Enc(x_0 || x_1) = Enc(x_0) || Enc(x_1) \]
DCA experiment

- Random 4-bit encoding Enc
- Correlation $\rho(S(x \oplus k)_0, Enc(S(x \oplus k^*)_j))$
DCA experiment

- With another (4-bit) encoding

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

- Natural approach: use known SCA/FA countermeasures

AES_k

\[ \Rightarrow \]

AES_k

\[ \Rightarrow \]

RNG

AES_k

\[ \Rightarrow \]

RNG

\[ m \rightarrow m \rightarrow m \rightarrow m \rightarrow m \rightarrow m \]

AES_k

masking, shuffling, ...

AES_k

masking, shuffling, ...

AES_k

masking, shuffling, ...

AES_k

masking, shuffling, ...

AES_k

masking, shuffling, ...

error
detection

c
c
c
c
c
c

c
c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

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 ≤ 50MB
  - executable ≤ 20MB
  - RAM consumption ≤ 20MB
  - running time ≤ 1sc

- Attackers could try to recover the keys of submitted implems
Score system

■ Unbroken implementation on day $n$

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

■ 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

No implementation got more than 1 strawberry before 08/20
Strawberry scores over time

No implementation got more than 1 before 08/20

Everything was broken in the end!
Strawberry scores over time

No implementation got more than 1 before 08/20

Everything was broken in the end!

Outstanding winner

No implementation got more than 1 before 08/20
Strawberry scores over time

No implementation got more than 1 before 08/20

Everything was broken in the end!

Outstanding winner

Several challenging implementations

No implementation got more than 1 before 08/20
Results

- 94 submitted implementations
- \( \sim 870 \) breaks
- Scoreboard:

<table>
<thead>
<tr>
<th>id</th>
<th>designer</th>
<th>breaker</th>
<th>score</th>
<th># days</th>
<th># breaks</th>
</tr>
</thead>
<tbody>
<tr>
<td>777</td>
<td>cryptolux</td>
<td>team_cryptexperts</td>
<td>406</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>815</td>
<td>grothendieck</td>
<td>cryptolux</td>
<td>78</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>753</td>
<td>sebastien-riou</td>
<td>cryptolux</td>
<td>66</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>877</td>
<td>chaes</td>
<td>You!</td>
<td>55</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>845</td>
<td>team4</td>
<td>cryptolux</td>
<td>36</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

cryptolux: Biryukov, Udovenko

team_cryptexperts: 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)
1020 functions of the form

```c
void xSnEq (uint UMNsVLp, uint KtFY, uint vzJZq) {
    if (nIlajqqq () == 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)
- Duplicates of 21 different functions
  - memory reading/writing
  - bitwise operations, bit shifts
  - goto, conditional jump
De-virtualisation

Simulation $\Rightarrow$ 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: $64 \times 64$ 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
Put it in Static Single Assignment (SSA) form:

\[
\begin{align*}
  x &= \ldots \\
  y &= \ldots \\
  t &= \neg x \\
  x &= t \oplus y \\
  y &= y \land t \\
  t &= x \lor y \\
  \vdots
\end{align*}
\]
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:

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

■ Several rounds: $\sim 600K \Rightarrow \sim 280K$ 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):

S-boxes?
Data dependency analysis

Data dependency graph (5% of the circuit):

S-boxes?
MixColumn?
Data dependency analysis

Data dependency graph (5% of the circuit):

- S-boxes?
- MixColumn?
- Initial pseudo-randomness generation?
Data dependency analysis

- Cluster analysis ⇒ gates within one “s-box”
- Identify all the outgoing variables: $s_1, s_2, \ldots, s_n$
- Likely hypothesis:

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

for some deterministic decoding function
Key recovery

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

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

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

\[
\begin{bmatrix}
   s_1^{(1)} & s_2^{(1)} & \cdots & s_m^{(1)} \\
   s_1^{(2)} & s_2^{(2)} & \cdots & s_m^{(2)} \\
   \vdots & \vdots & \ddots & \vdots \\
   s_1^{(n)} & s_2^{(n)} & \cdots & s_m^{(n)}
\end{bmatrix}
\]

\[
\begin{bmatrix}
   S_j(x^{(1)} \oplus k) \\
   S_j(x^{(2)} \oplus k) \\
   \vdots \\
   S_j(x^{(n)} \oplus k)
\end{bmatrix}
\]

Linear system solvable for $k = k^*$
Key recovery

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

\[
\begin{bmatrix}
    s_1^{(1)} & s_2^{(1)} & \cdots & s_m^{(1)} \\
    s_1^{(2)} & s_2^{(2)} & \cdots & s_m^{(2)} \\
    \vdots & \vdots & \ddots & \vdots \\
    s_1^{(n)} & s_2^{(n)} & \cdots & s_m^{(n)}
\end{bmatrix}
\]

\[
\begin{bmatrix}
    S_j(x^{(1)} \oplus k) \\
    S_j(x^{(2)} \oplus k) \\
    \vdots \\
    S_j(x^{(n)} \oplus k)
\end{bmatrix},
\]
Key recovery

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

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

\[\begin{align*}
  j = 0: & \quad 0,0,0,0,0,0,1,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 1: & \quad 0,0,0,0,0,1,0,0,1,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 2: & \quad 0,0,0,0,0,1,0,1,0,0,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 3: & \quad 0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 4: & \quad 0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 5: & \quad 0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 6: & \quad 0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 7: & \quad 0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
\end{align*}\]
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 $k \neq k^*$

\[
\begin{align*}
j &= 0: 0,0,0,0,0,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
j &= 1: 0,0,0,0,0,0,1,0,0,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
j &= 2: 0,0,0,0,0,0,0,1,0,1,0,0,0,1,1,1,1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0 \\
j &= 3: 0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
j &= 4: 0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0 \\
j &= 5: 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
j &= 6: 0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
j &= 7: 0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
\end{align*}
\]
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 $k \neq k^*$

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

- Decoding

$$\begin{align*}
(s_7, s_8, \ldots, s_{21}) \quad \text{× Bin. Mat.} \quad (S_0(x \oplus k^*), \ldots, S_7(x \oplus k^*))
\end{align*}$$

15 outgoing bits

8 s-box coordinates
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 $k \neq k^*$

\[
\begin{align*}
  j = 0: & \quad 0,0,0,0,0,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 1: & \quad 0,0,0,0,0,1,0,0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 2: & \quad 0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 3: & \quad 0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 4: & \quad 0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 5: & \quad 0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 6: & \quad 0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \\
  j = 7: & \quad 0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
\end{align*}
\]

Decoding

\[
\begin{align*}
  (s_7, s_8, \ldots, s_{21}) \times \text{Bin. Mat.} \rightarrow (S_0(x \oplus k^*), \ldots, S_7(x \oplus k^*))
\end{align*}
\]

15 outgoing bits

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?