Differential Computation Analysis

Hiding your White-Box Designs is Not Enough

Joppe W. Bos

Summer school on real-world crypto and privacy

Šibenik, Croatia

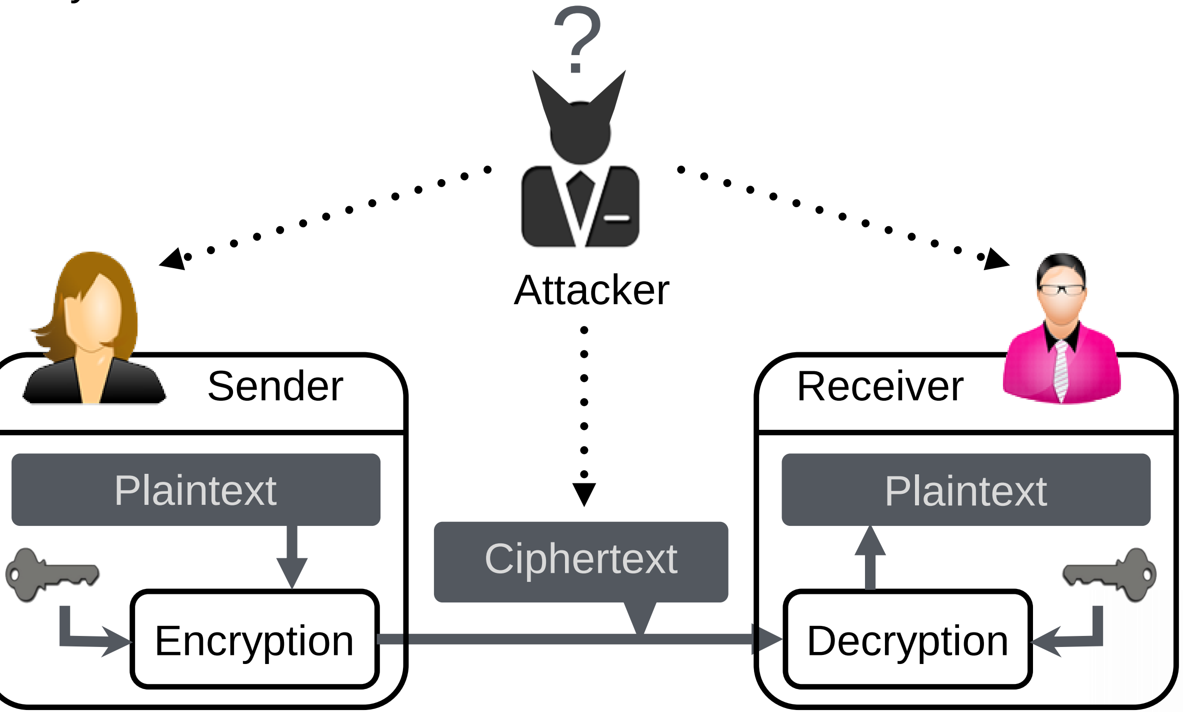
NXP Semiconductors

Operations in > 35 countries, more than 130 facilities
≈ 45,000 employees

Research & Development
≈ 11,200 engineers in 23 countries

The presence of an attacker

- Where should we assume the attacker to be? What is most realistic?
  - Is the attacker only eavesdropping on the communication channel?
  - Or did one of the (trusted/authorized) end-users become the attacker?
  - Or are there any malware/viruses installed on a trusted end-user's device?

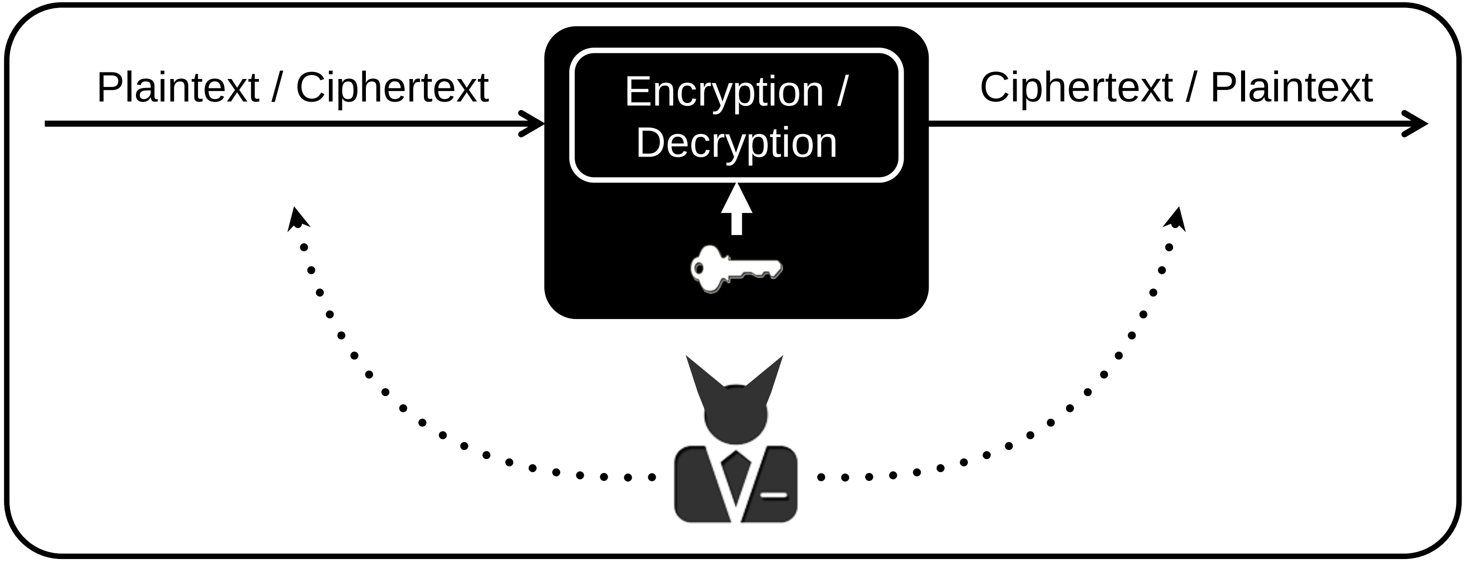


Cryptography & Security Notions

- In order to properly assess the security of (the implementation of) a cryptographic algorithm, one needs a clear definition of a *security notion*.

- **Security Notion = attacker's goal + attacker model.**
  - *Attacker's goal*: what does the attacker want to achieve?
    - This is not always key-extraction, the attacker is often satisfied with much less…
  - *Attacker model*: what are the capabilities of the attacker in order for him to achieve his goal?
    - Such a model tries to capture the capabilities of an attacker **as realistically as possible**, i.e., modeling the hostile environment in which the implementation of a cryptographic primitive is deployed.

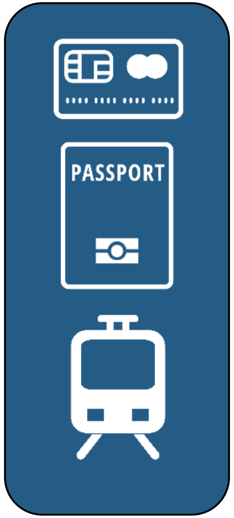
Black box model



Initial cryptographic security model from the 1980s

- Endpoints are trusted parties
- Attacker “observes” data being transferred

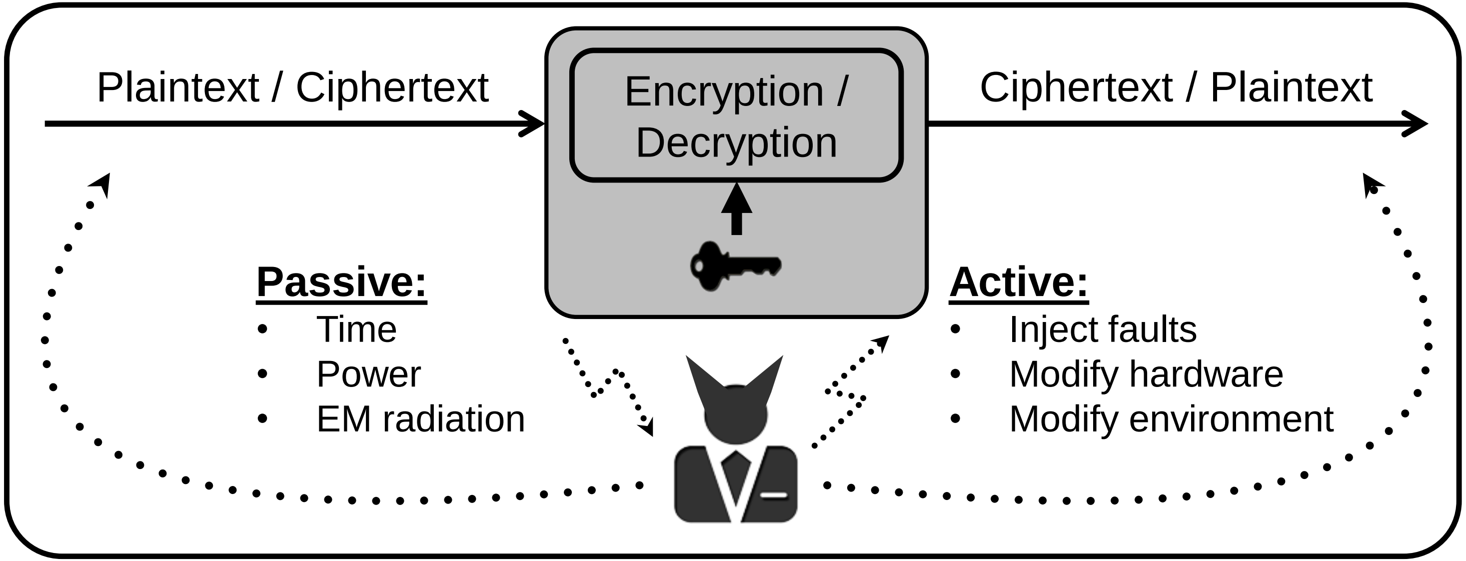
Black box model → grey box model



- When technology changed this model did not reflect reality any longer
- Cryptographic algorithms implemented in hardware were originally thought to form a secure environment
- In 1999 it was publicly shown that hardware implementations tend to leak key-correlated information

Kocher, Jaffe, Jun. Differential power analysis. In CRYPTO 1999

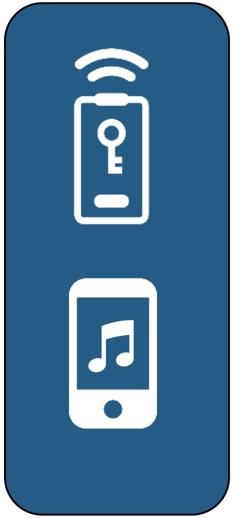
Grey box model



The research area of side-channel attacks and resistance has grown significantly: *fault injections, simple power analysis, differential power analysis, correlation power analysis, template attacks, higher-order correlation attacks, mutual information analysis, linear regression analysis, horizontal analysis, vertical analysis etc. etc.*

See the CHES conference

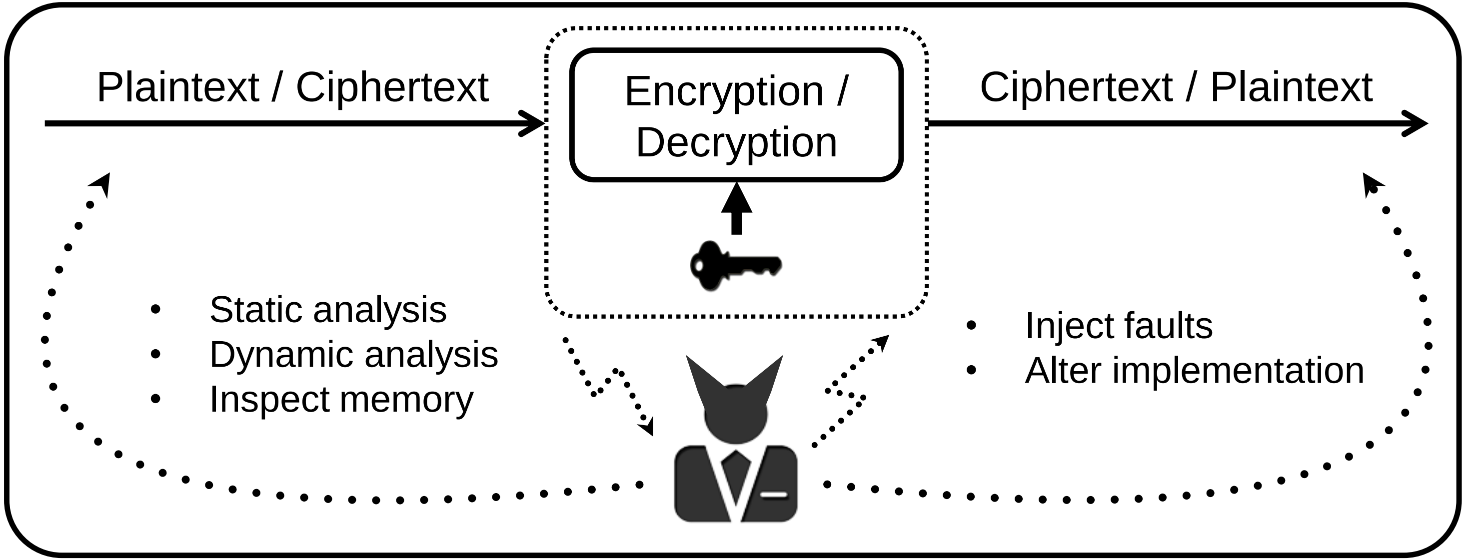
Grey box model → white box model



- When technology changed this model did not reflect reality any longer
- Increase in mobile devices without dedicated hardware support → need to rely on software solutions
- In 2002 the white-box model was introduced
  Initial focus on DRM applications.

Chow, Eisen, Johnson, van Oorschot. White-box cryptography and an AES implementation. In SAC 2002.
Chow, Eisen, Johnson, van Oorschot. A white-box DES implementation for DRM applications. In Security and Privacy in Digital Rights Management, 2003.

White box model



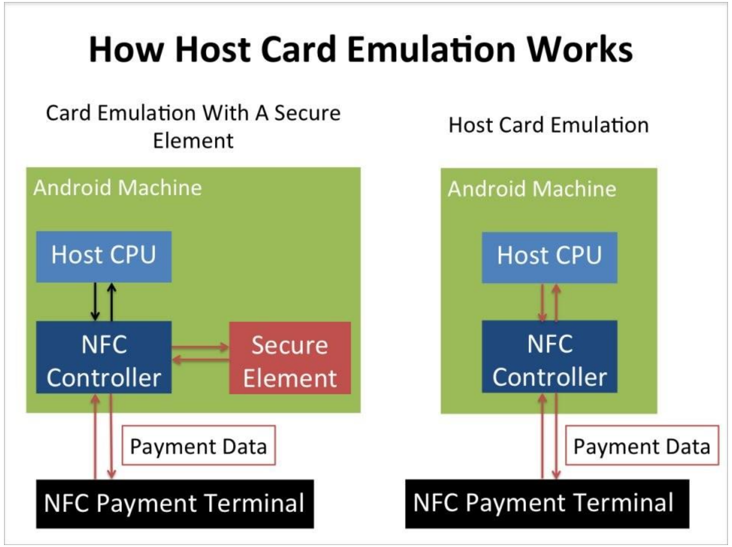
Adversary owns the device running the software. Powerful capabilities

- ✓ has full access to the source code
- ✓ inspect and alter the memory used
- ✓ perform static analysis
- ✓ alter intermediate results

White box crypto - applications

Applications of WB crypto has evolved to protection of

- digital assets
- mobile device (from an application store)
- Host Card Emulation (HCE)
- credentials for an authentication to the cloud



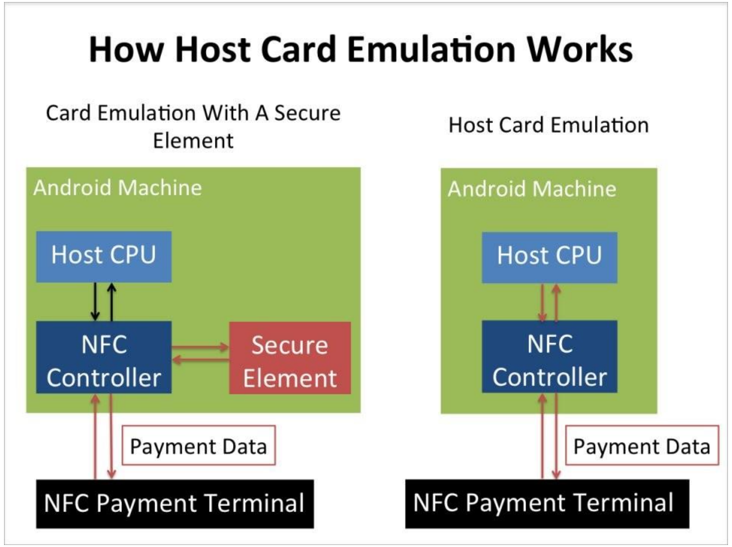
Source: Business Insider

Chow, Eisen, Johnson, van Oorschot. White-box cryptography and an AES implementation. In SAC 2002.

White box crypto - applications

Applications of WB crypto has evolved to protection of

- digital assets
- mobile device (from an application store)
- Host Card Emulation (HCE)
- credentials for an authentication to the cloud

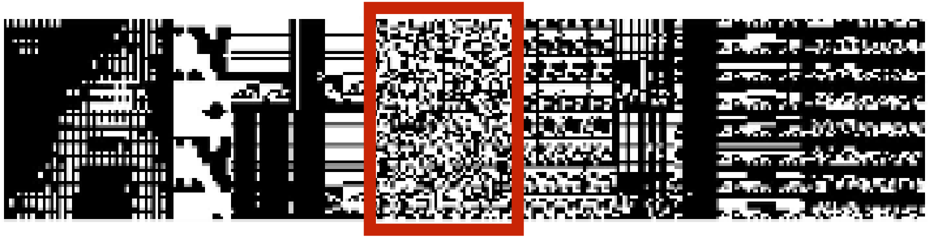


Source: Business Insider

How to realize a white-box implementation in practice?
“*when the attacker has internal information about a cryptographic implementation, choice of implementation is the sole remaining line of defense*”

Chow, Eisen, Johnson, van Oorschot. White-box cryptography and an AES implementation. In SAC 2002.

White-Box basic idea – Why?

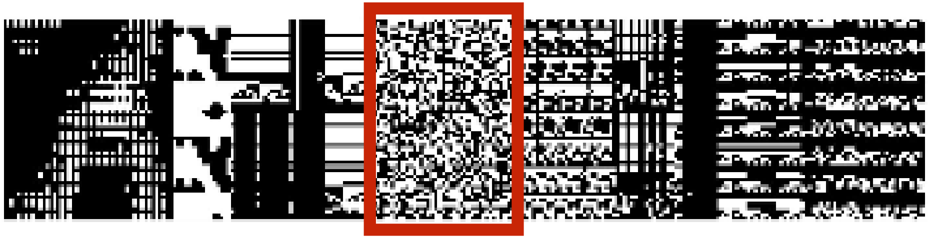


■ 0-bit □ 1-bit

- **Entropy attack**
  - Locate the unusual high entropy of the cryptographic key in a memory dump using sliding windows for example.

Shamir, van Someren: *Playing "Hide and Seek" with Stored Keys*. Financial Cryptography 1999

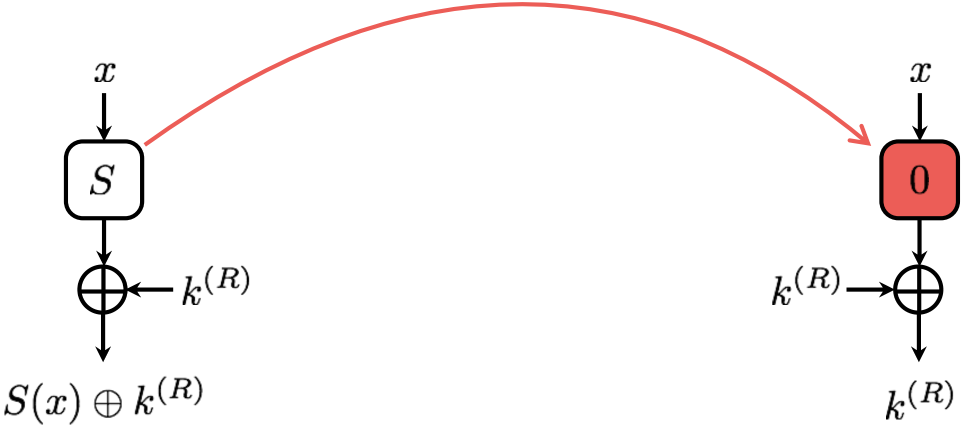
White-Box basic idea – Why?



■ 0-bit   □ 1-bit

- **Entropy attack**
  – Locate the unusual high entropy of the cryptographic key in a memory dump using sliding windows for example.

Shamir, van Someren: *Playing "Hide and Seek" with Stored Keys*. Financial Cryptography 1999



- **S-box blanking attack**
  – Locate the publicly defined S-boxes in the binary and overwrite it with all zeros such that $S(x)=0$ for any x.

Kerins, Kursawe: *A cautionary note on weak implementations of block ciphers*. *WISSec*, 2006

Security of WB solutions - Theory

White box can be seen as a form of code obfuscation

- It is known that obfuscation of **<u>any</u>** program is impossible

Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, Yang. On the (im)possibility of obfuscating programs. In CRYPTO 2001

- Unknown if a (sub)family of white-box functions can be obfuscated
- If secure WB solution exists then this is protected (by definition!) to **all** *current* and *future* side-channel and fault attacks!

Security of WB solutions - Theory

White box can be seen as a form of code obfuscation

- It is known that obfuscation of **<u>any</u>** program is impossible

Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, Yang. On the (im)possibility of obfuscating programs. In CRYPTO 2001

- Unknown if a (sub)family of white-box functions can be obfuscated
- If secure WB solution exists then this is protected (by definition!) to **all** *current* and *future* side-channel and fault attacks!

Practice

- Only results known for symmetric crypto (all academic designs broken)
- Convert algorithms to sequence of LUTs
- Embed the secret key in the LUTs
- Obfuscate the LUTs by using encodings

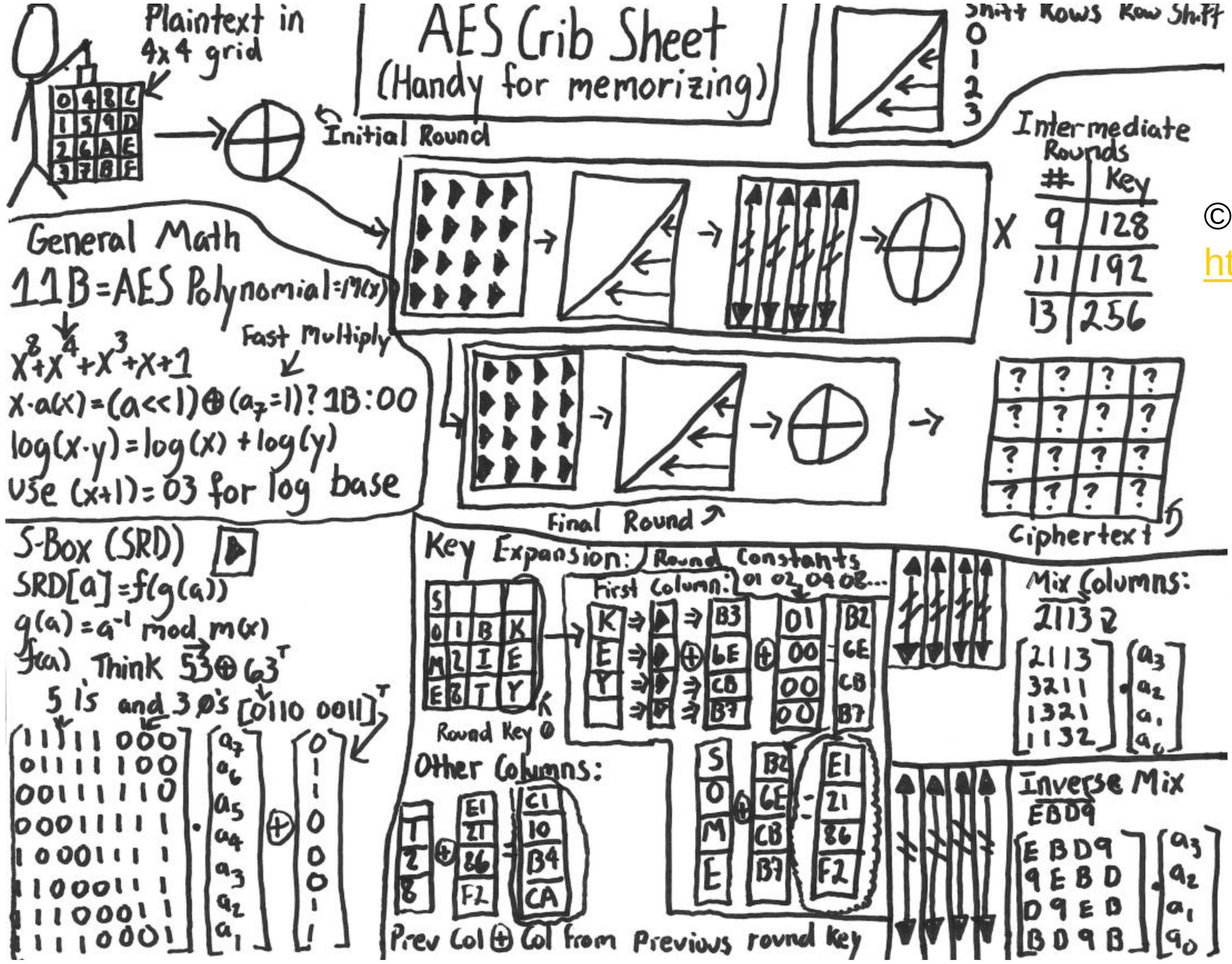
WB Impossible?

No! “Ideal” WB AES implementation
One big lookup table → 2^{92} TB storage required

<u>Practical WB AES?</u>
Network of smaller tables: ≈ 700 kB
Encoding on intermediate values using ideas by Chow

Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. White-box cryptography and an AES implementation, in SAC 2002.

<u>Generic idea.</u>
Transform a cipher into a network of randomized key-instantiated look-up tables


AES Crib Sheet
(Handy for memorizing)
Plaintext in 4x4 grid
Initial Round
Shift Rows
Row Shift
0
1
2
3
Intermediate Rounds
| Key
9 | 128
11 | 192
13 | 256
X
General Math
11B = AES Polynomial = m(x)
x^8+x^4+x^3+x+1
Fast Multiply
x·a(x) = (a<<1) ⊕ (a7=1) ? 1B : 00
log(x·y) = log(x) + log(y)
Use (x+1) = 03 for log base
Final Round
Ciphertext
S-Box (SRD)
SRD[a] = f(g(a))
g(a) = a^-1 mod m(x)
f(a) Think 53 ⊕ 63^T
5 1's and 3 0's [0110 0011]^T
Key Expansion:
Round Constants
01 02 04 08 ...
First Column:
Round Key 0
Other Columns:
Prev Col ⊕ Col from Previous round key
Mix Columns:
2113
Inverse Mix
EBD9

AES with look-up tables

- The key addition and S-box operations are merged into a single operation (8 bit $\rightarrow$ 8 bit table $\rightarrow$ 256 byte)

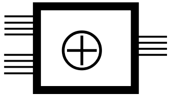
$$c_{i,j} = Sbox(a_{i,j} \oplus k_{i,j}) = T_{i,j}(a_{i,j})$$

- To simplify: we omit ShiftRow operation
  - Corresponds to renumbering of indices

- The MixColumn operation can be split into four byte-to-32-bit (8 bit $\rightarrow$ 32 bit table $\rightarrow$ 1024 byte) operations:

$$\bar{d}_j = M_0 T_{0,j}(a_{0,j}) \oplus M_1 T_{1,j}(a_{1,j}) \oplus M_2 T_{2,j}(a_{2,j}) \oplus M_3 T_{3,j}(a_{3,j})$$

- We can now implement a round by only using the following 2 types of lookup tables:



AES with look-up tables + obfuscation

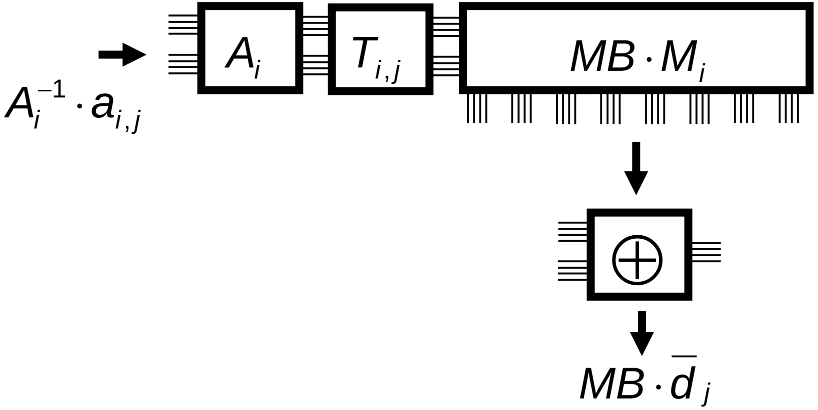
- Since S-boxes and matrix M are known, the key can easily be extracted from the lookup tables.
- **Solution**: obfuscating lookup tables by encoding their input and output.

AES with look-up tables + obfuscation

- Since S-boxes and matrix M are known, the key can easily be extracted from the lookup tables.

- **Solution**: obfuscating lookup tables by encoding their input and output.

- First, we apply **linear** encodings:
  - A_i: random 8-bit linear mapping
  - MB: random 32-bit linear mapping

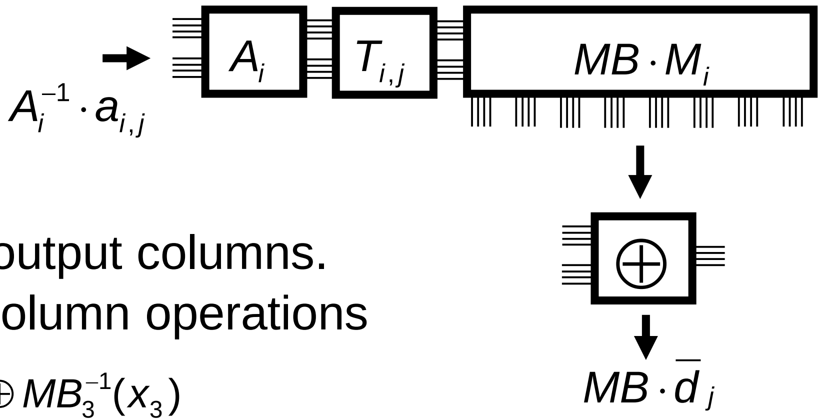


AES with look-up tables + obfuscation

- Since S-boxes and matrix M are known, the key can easily be extracted from the lookup tables.

- **Solution**: obfuscating lookup tables by encoding their input and output.

- First, we apply **linear** encodings:
  - A_i: random 8-bit linear mapping
  - MB: random 32-bit linear mapping

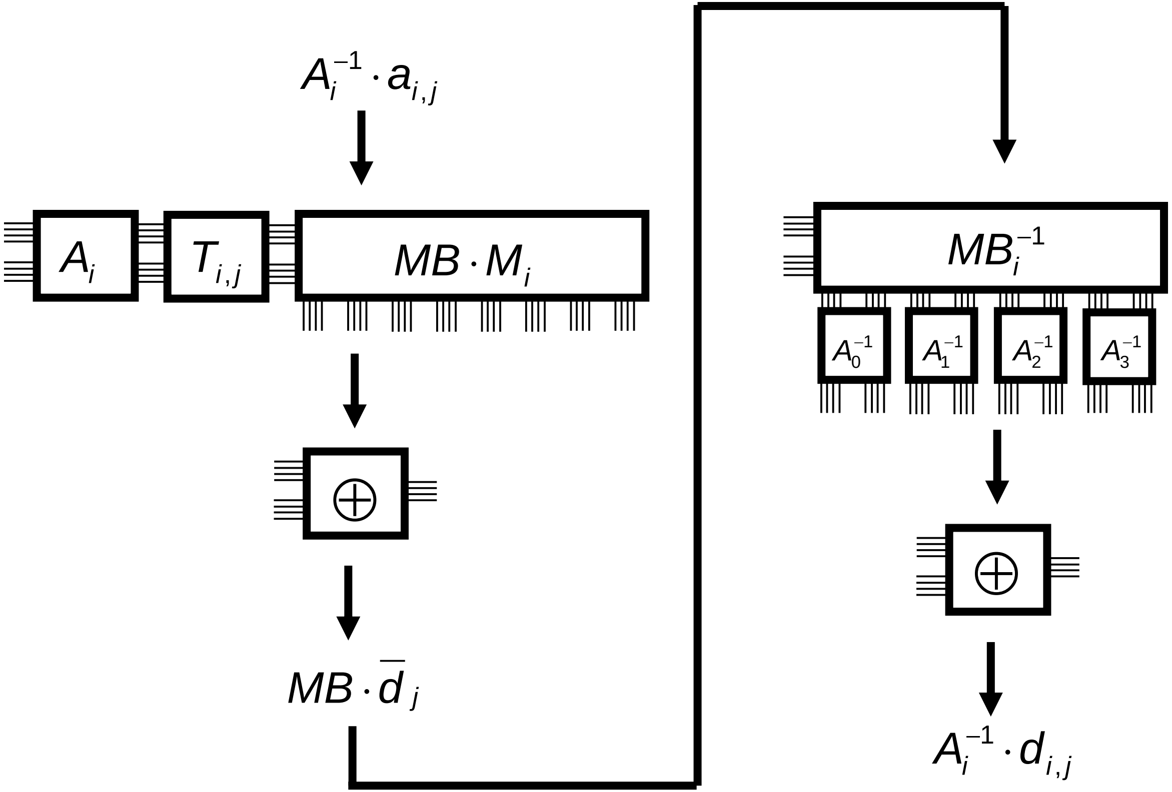


- Matrix MB is removed from the computed output columns. Implemented in the same way as the MixColumn operations

$$MB^{-1}(\bar{x}) = MB_0^{-1}(x_0) \oplus MB_1^{-1}(x_1) \oplus MB_2^{-1}(x_2) \oplus MB_3^{-1}(x_3)$$

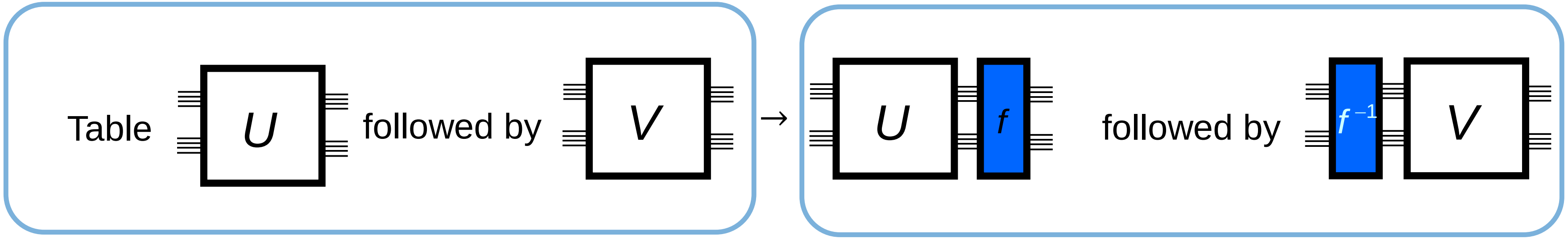
- Merge the MB_i-tables by the linear encodings used in the next round.

AES with look-up tables + obfuscation - Flow



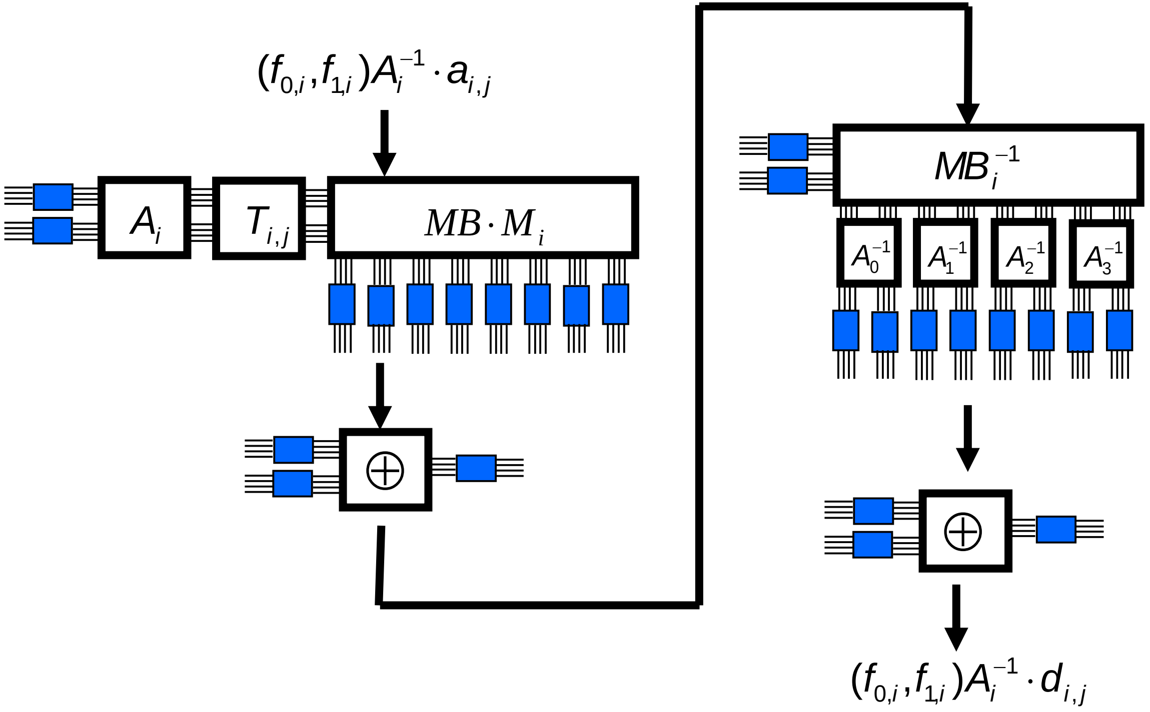
Obfuscation, obfuscation, obfuscation

- In addition to the *linear* encodings, also add **non-linear** encodings f.



Obfuscation, obfuscation, obfuscation

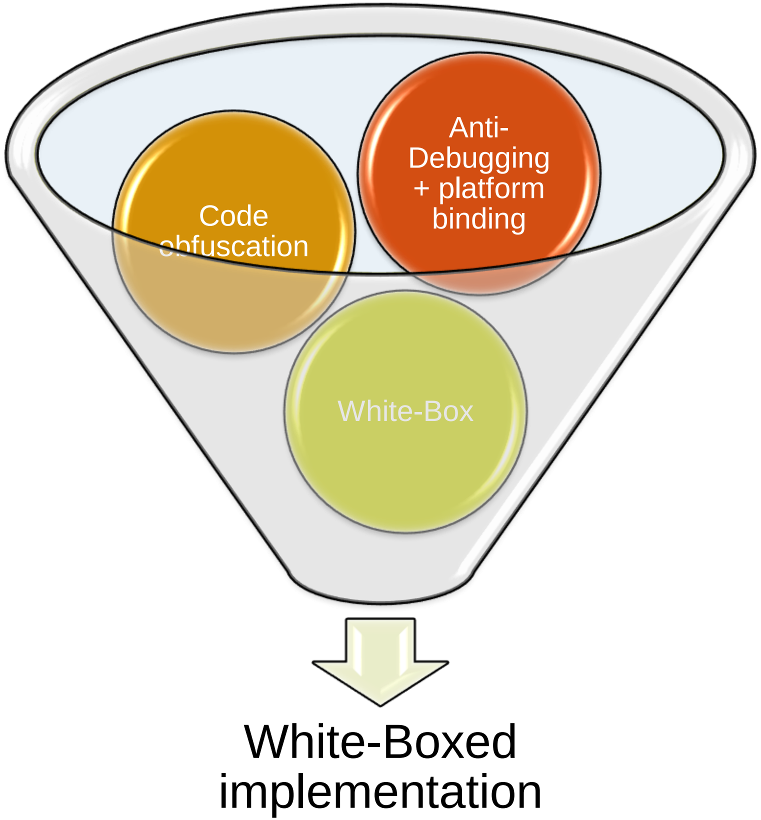
- In addition to the *linear* encodings, also add **non-linear** encodings f.



Size implementation: ≈ 700 kB

Chow, Eisen, Johnson, van Oorschot. White-box cryptography and an AES implementation. In SAC 2002.

White box crypto - practice



In practice the white box is the most essential but a **small part** of the entire software implementation

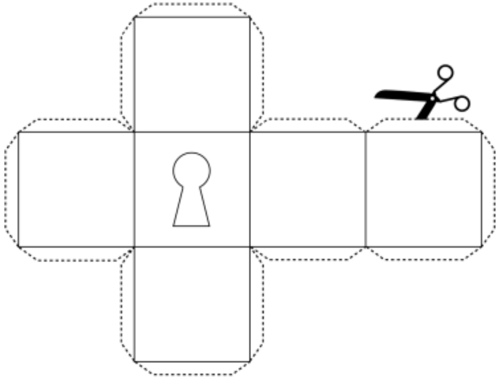
- Strong code obfuscation
- Binary is “glued”” to the environment
  - Prevent code-lifting
- Support for traitor tracing
- Mechanism for frequent updating

More details see the invited talk at EC 2016 *Engineering Code Obfuscation* by Christian Collberg

Remainder of the talk

<u>Focus on the white-box only</u>

White box crypto - practice



- White-box “solutions” are known for standard symmetric crypto only

- All published (academic) designs have been theoretically broken

So what can do to improve the attacks?

Effort and expertise required

- Previous WB attacks were WB approach specific which means
  - know the type of encodings that are applied on the *intermediate results*
  - know which *cipher operations* are implemented by which *(network of) lookup tables*
  - This implies **time-consuming reverse-engineering** of the code and then applying a sophisticated (algebraic) attack

Effort and expertise required

- Previous WB attacks were WB approach specific which means
  - know the type of encodings that are applied on the *intermediate results*
  - know which *cipher operations* are implemented by which *(network of) lookup tables*
  - This implies **time-consuming reverse-engineering** of the code and then applying a sophisticated (algebraic) attack
- Our attack allows to assess the security of a WB implementation
  - Automatically
  - Without knowledge of the implementation choices / details of the underlying scheme (just the scheme itself)
  - Ignores all (attempts) at code-obfuscation
  - No expertise required (execute our scripts and wait for the key)

SOFTWARE TRACES

Tracing binaries

- Academic attacks are on open design
- In practice: what you get is a binary blob
  - No design documents
  - No algorithm specification
  - No source code

Idea: Does the WB leak info?

In other words:
Verify if one can correlate guesses to some intermediate results (which use the key) using software traces with the help of

dynamic binary instrumentation tools

Tracing binaries

- Academic attacks are on open design
- In practice: what you get is a binary blob

Idea: create software traces using *dynamic binary instrumentation* tools

- Record all instructions and memory accesses.

Examples of the tools we extended / modified

  - Intel PIN (x86, x86-64, Linux, Windows, Wine/Linux)
  - Valgrind (idem+ARM, Android)

Tracing binaries

- Academic attacks are on open design
- In practice: what you get is a binary blob

Idea: create software traces using *dynamic binary instrumentation* tools

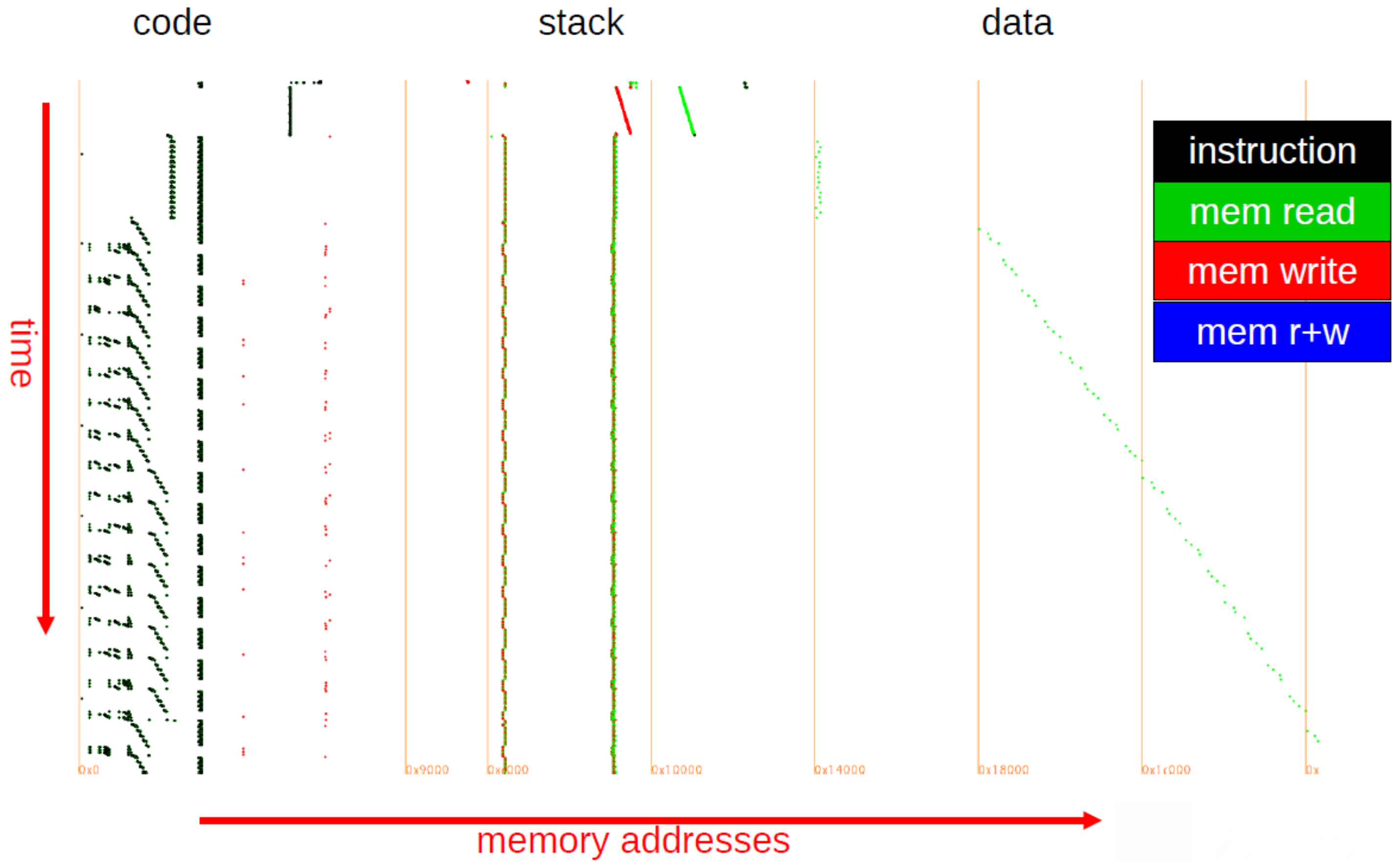
- Record all instructions and memory accesses.

Examples of the tools we extended / modified

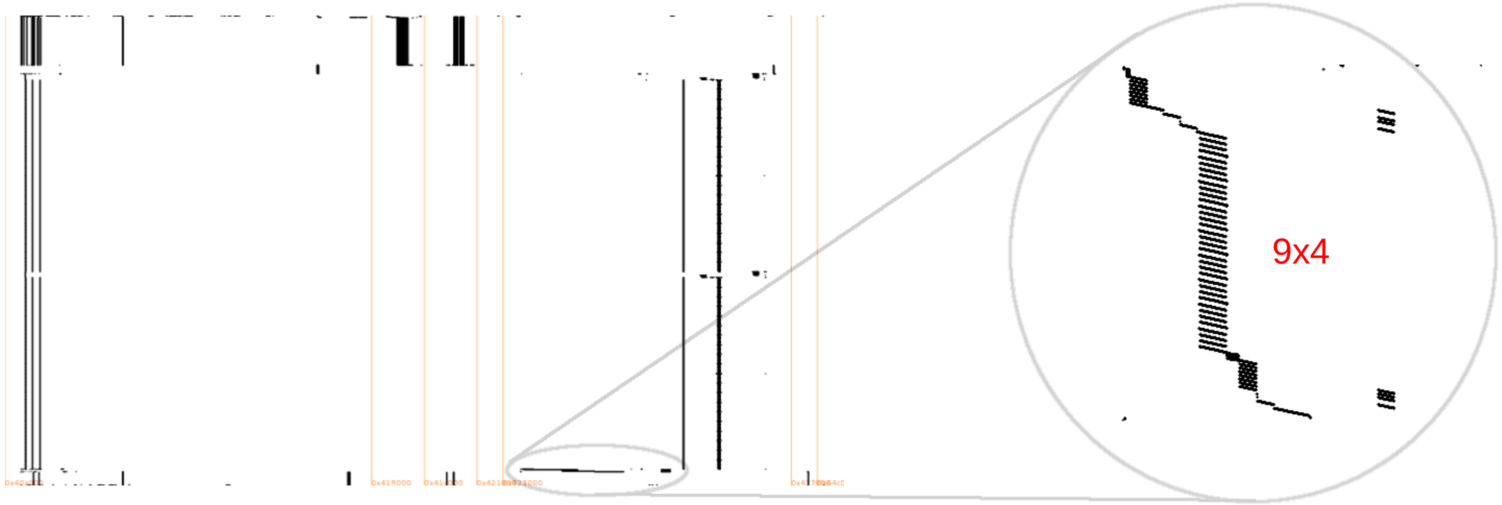
  - Intel PIN (x86, x86-64, Linux, Windows, Wine/Linux)
  - Valgrind (idem+ARM, Android)

- Using traces:
  1. One trace: Visual identification of white-box, code-/table-lifting
  2. Few traces: data correlation, standard deviation, etc
  3. More traces: DPA-based attack

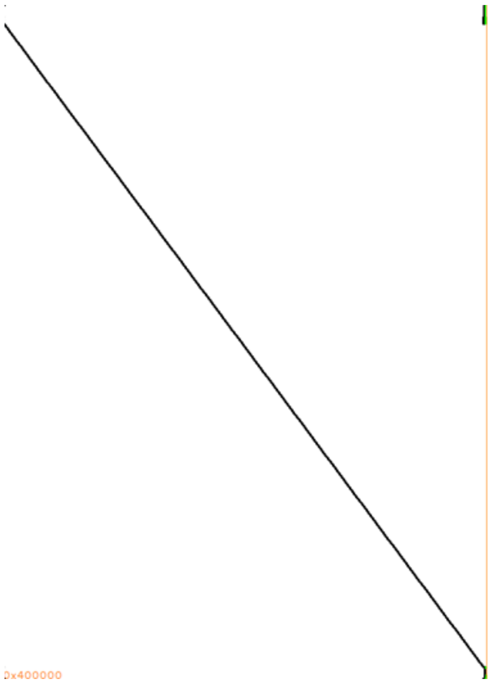
Trace visualization convention: pTra waterfall



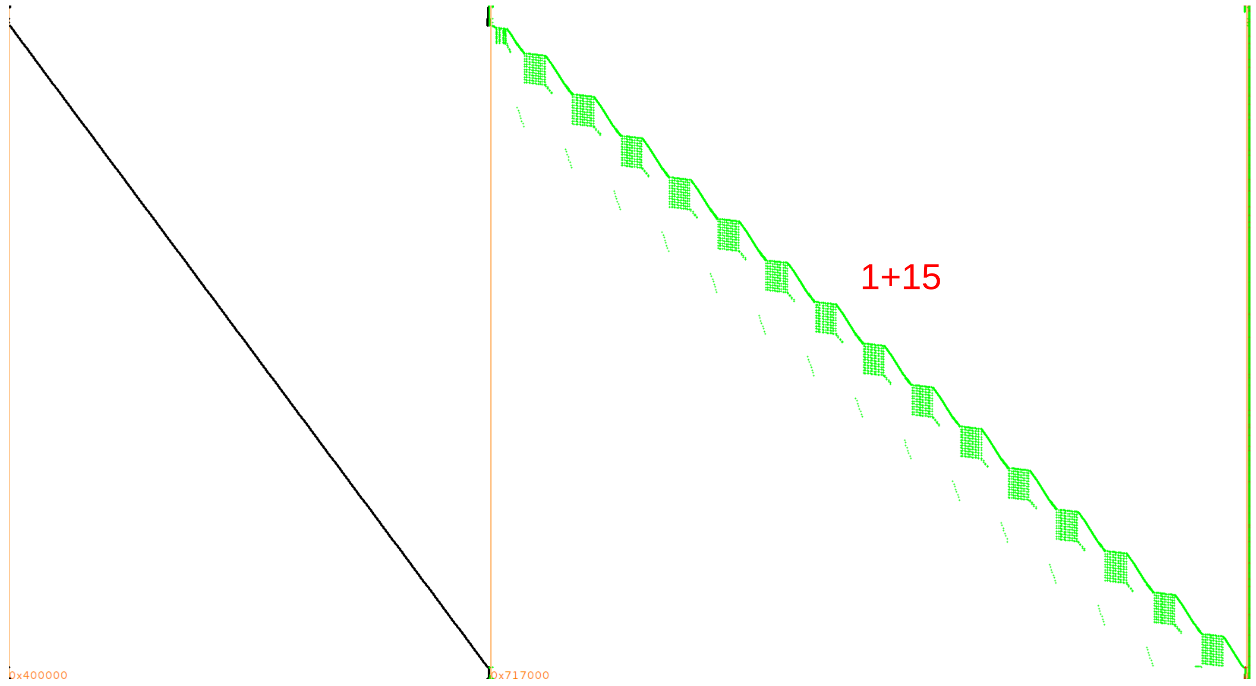
Visual crypto identification: code



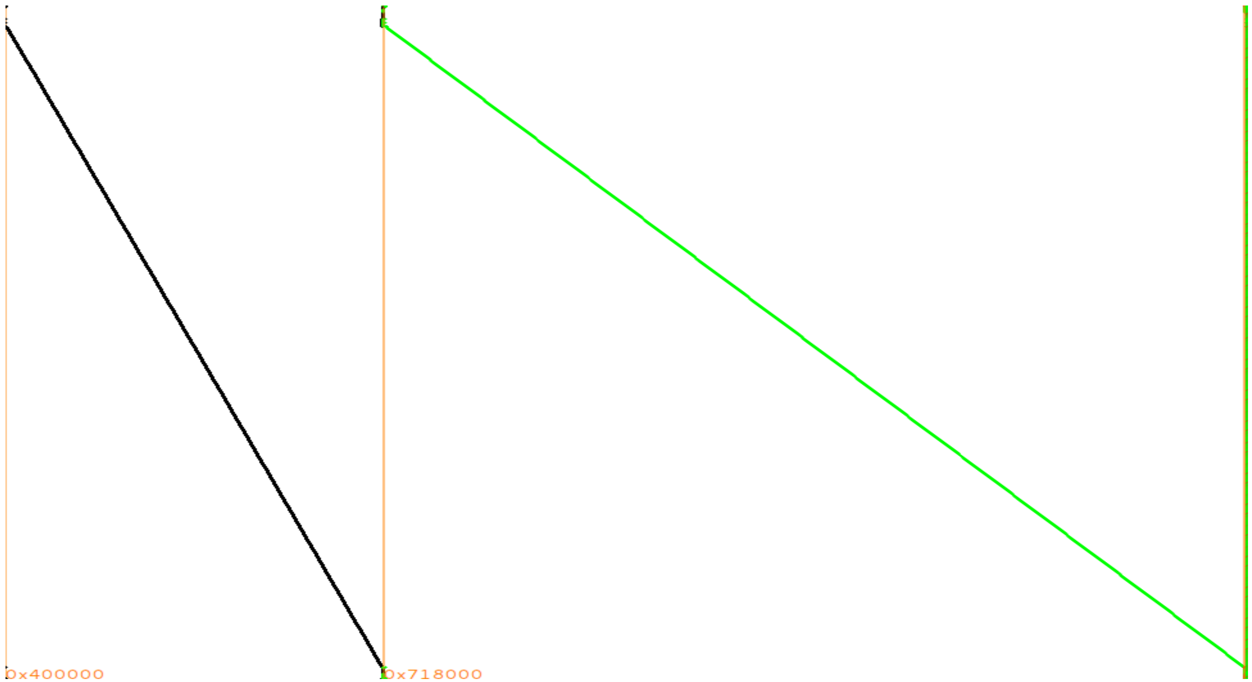
Visual crypto identification: code?



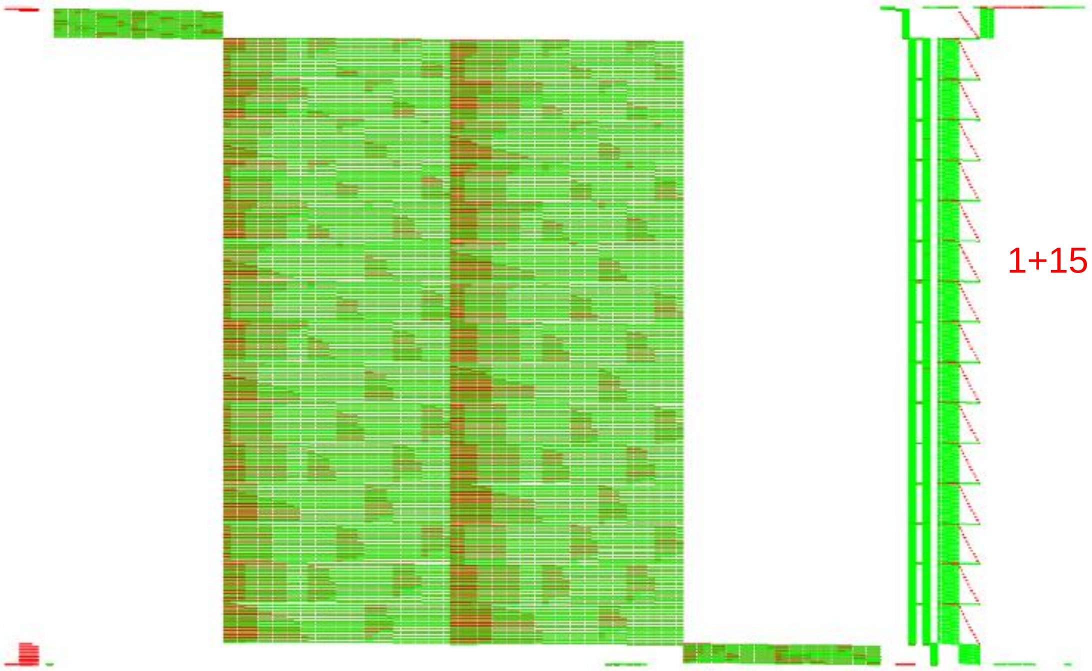
Visual crypto identification: code? data!



Visual crypto identification: code? data?



Visual crypto identification: stack!



Where is the key?

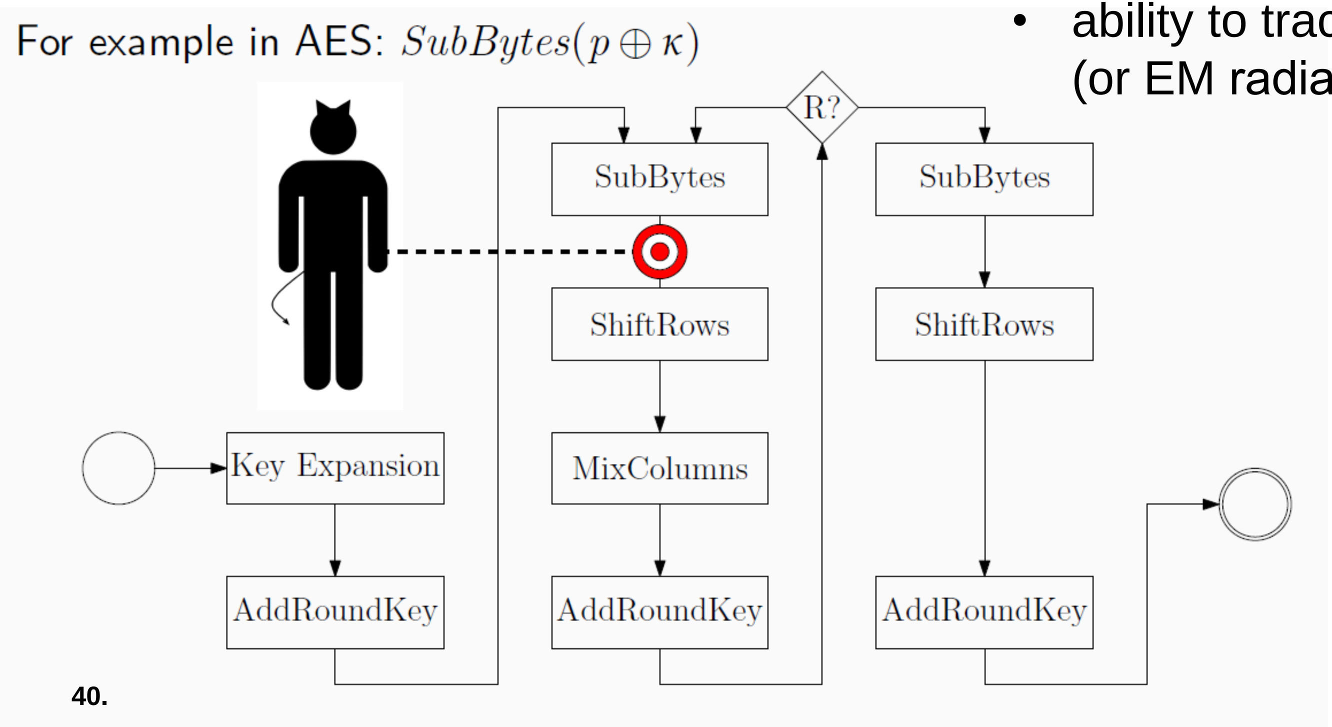
Differential Power Analysis and friends

P. C. Kocher, J. Jaffe, and B. Jun: *Differential power analysis*. CRYPTO'99

Very powerful grey box attack!

Requirements

- known input or known output
- ability to trace power consumption (or EM radiations, or …)



Differential Computation Analysis

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

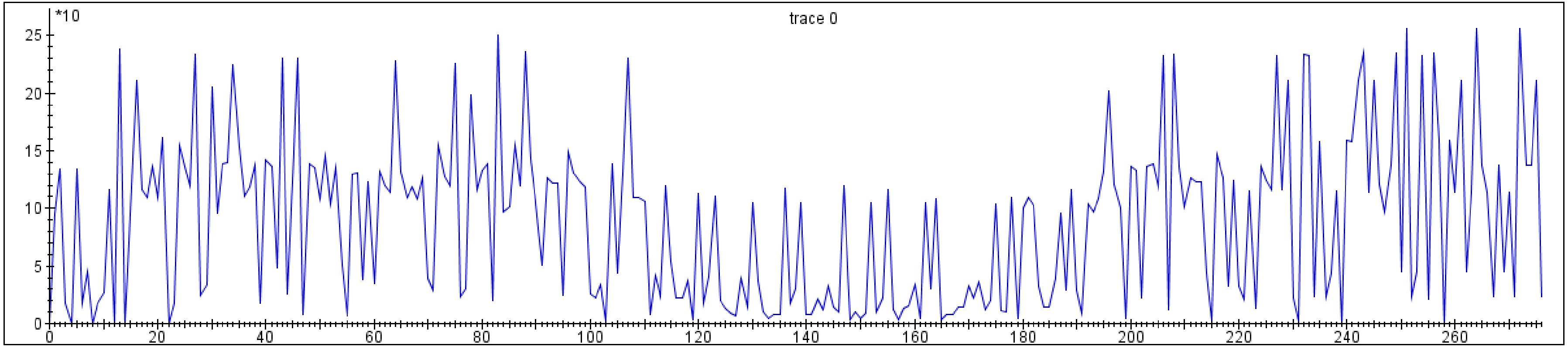
Differential Computation Analysis

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

Make pseudo power traces from our software execution traces

→ this are lists of memory accesses / data + stack writes / …

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

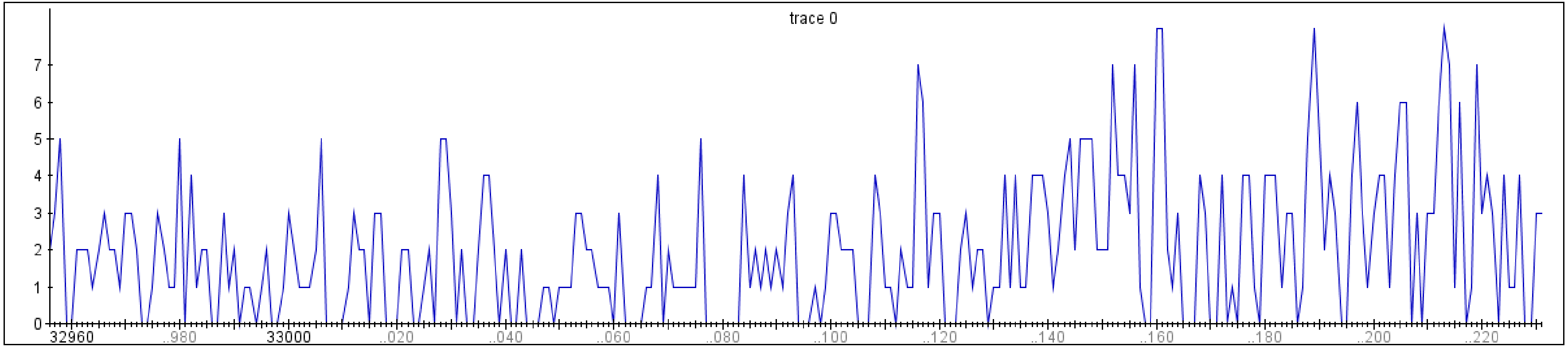


→ 256 possible discrete values

Differential Computation Analysis

256 possible discrete values but bit values dominated by the MSB

→ Build Hamming weight traces?



→ 8 possible discrete values

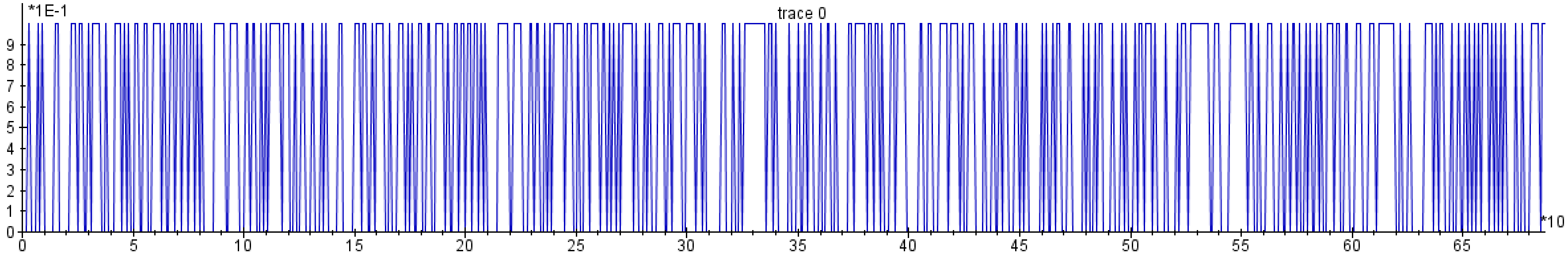
That works but we can do better…

recall: Hamming weight was a **hardware model** for combined bit leaks

Differential Computation Analysis

Each bit of those bytes is equally important
address bits represent a different way to partition the look-up tables

→ Serialize bytes in a succession of bits



→ 2 possible discrete values: 0's and 1's

DCA: DPA on software traces

HW analogue: this is like probing each bus-line individually ***without any error***

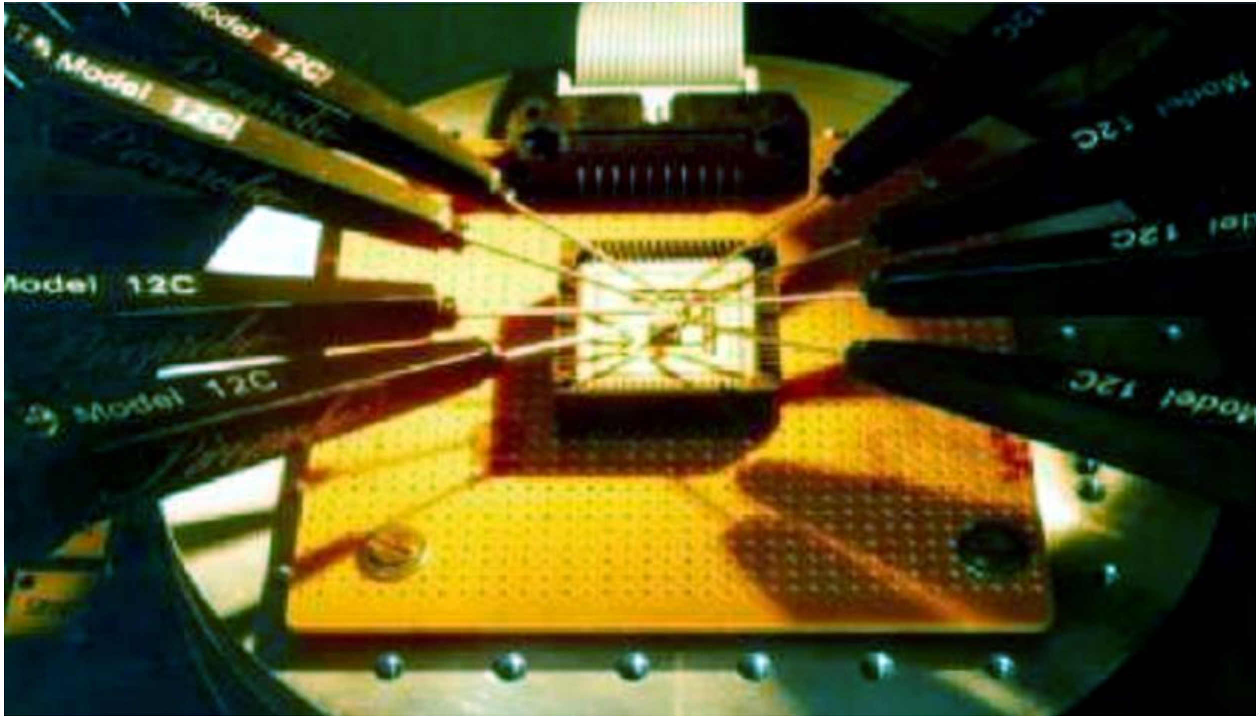


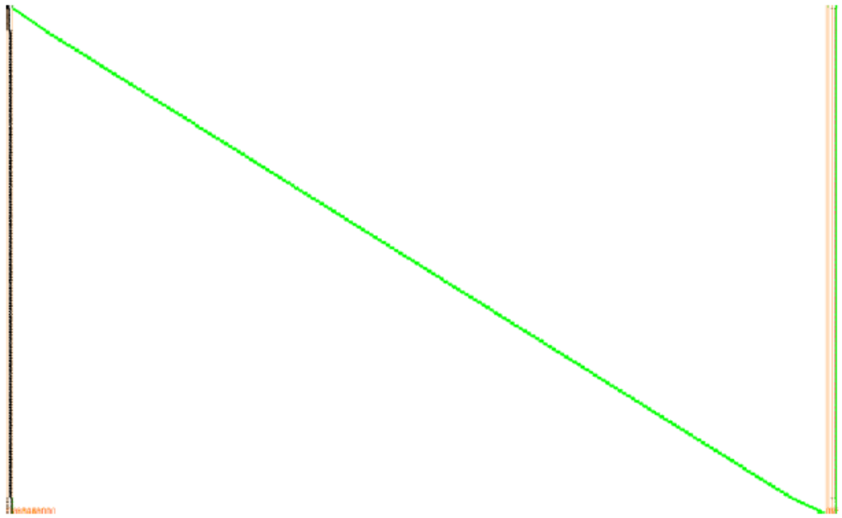
Image source: Brightsight

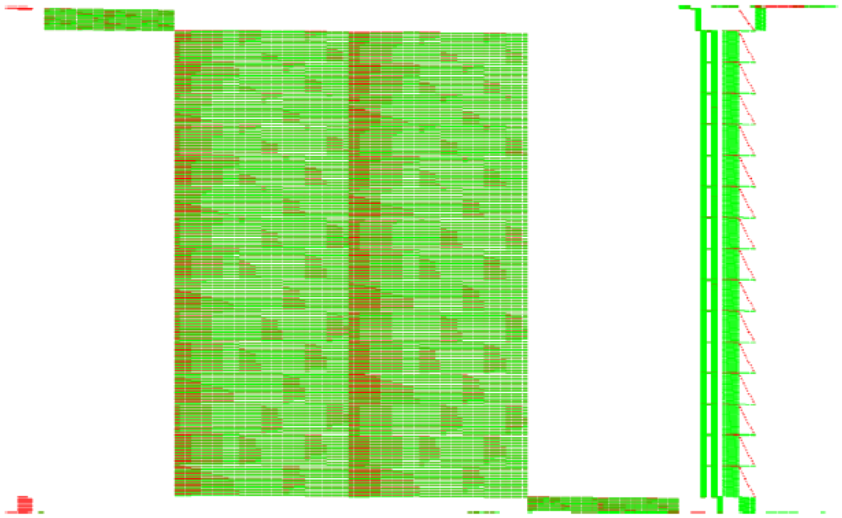
Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let’s check!

WB implementation	Algorithm	#traces
Wyseur challenge, 2007		
Hack.lu challenge, 2009		
SSTIC challenge, 2012		
Klinec implementation, 2013		

Wyseur challenge





Chow+: Chow-based plus personal improvements by Brecht Wyseur

Chow, Eisen, Johnson, van Oorschot. A white-box DES implementation for DRM applications. In Security and Privacy in Digital Rights Management, 2003.
E. Link and W. D. Neumann. Clarifying obfuscation: Improving the security of white-box DES. In International Symposium on Information Technology: Coding and Computing (ITCC 2005)

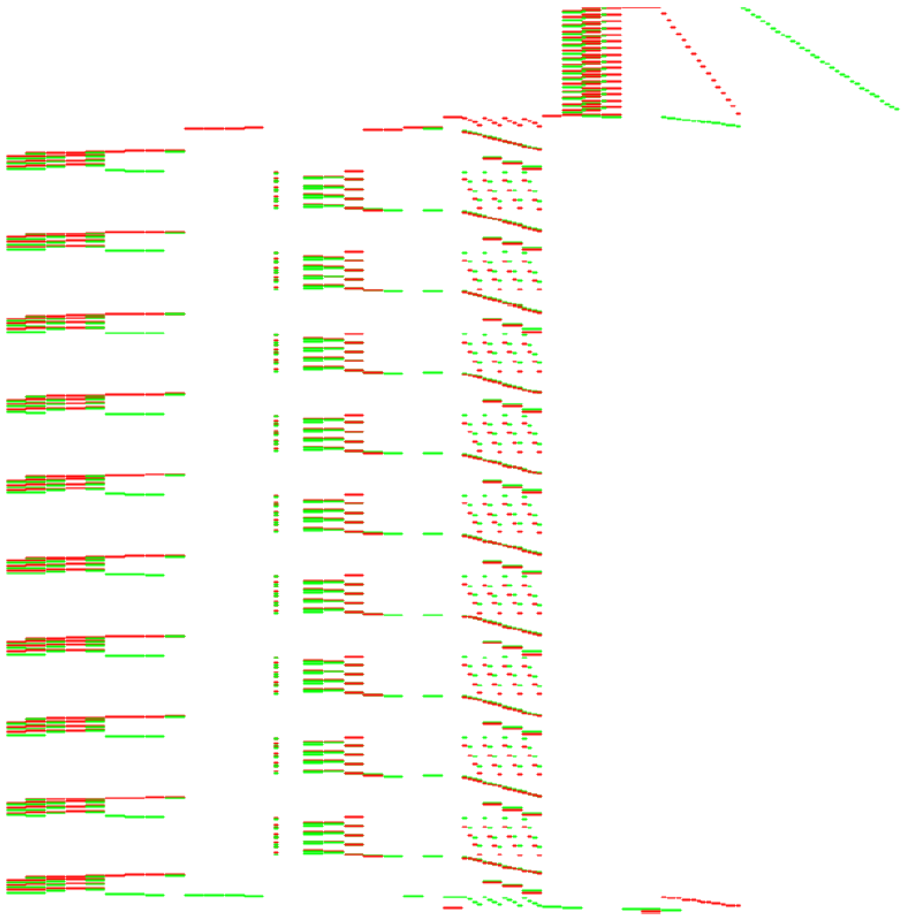
Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let's check!

WB implementation	Algorithm	#traces
Wyseur challenge, 2007	DES (Chow+)	65
Hack.lu challenge, 2009		
SSTIC challenge, 2012		
Klinec implementation, 2013		

Hack.lu challenge

Zoom on the stack



- ✓ AES-128
- ✓ Very easy to break
  (designed for a one-day challenge)

Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let's check!

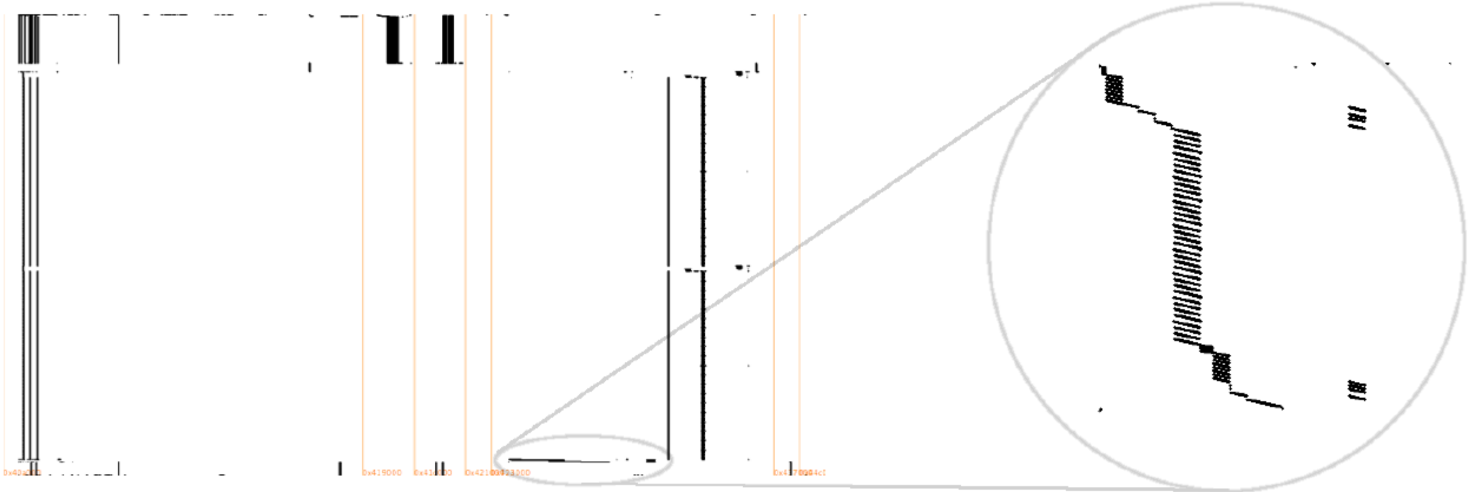
WB implementation	Algorithm	#traces
Wyseur challenge, 2007	DES (Chow+)	65
Hack.lu challenge, 2009	AES (Chow)	16 (no encodings)
SSTIC challenge, 2012		
Klinec implementation, 2013		

Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let's check!

WB implementation	Algorithm	#traces
Wyseur challenge, 2007	DES (Chow+)	65
Hack.lu challenge, 2009	AES (Chow)	16 (no encodings)
SSTIC challenge, 2012	DES	16 (no encodings)
Klinec implementation, 2013		

Klinec



- AES using Karroumi's approach (using dual ciphers)
- More difficult, not all correct key bytes are #1

M. Karroumi. Protecting white-box AES with dual ciphers. In ICISC 2011

Klinec

- Balanced encodings?
  - It may become the *least* candidate, this is still standing out!

target bit	key byte 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	256	255	256	255	256	253	1	256	256	239	256	1	1	1	255
1	1	256	256	256	1	255	256	1	1	5	1	256	1	1	1	1
2	256	1	255	256	1	256	226	256	256	256	1	256	22	1	256	256
3	256	255	251	1	1	1	254	1	1	256	256	253	254	256	255	256
4	256	256	74	256	256	256	255	256	254	256	256	256	1	1	256	1
5	1	1	1	1	1	1	50	256	253	1	251	256	253	1	256	256
6	254	1	1	256	254	256	248	256	252	256	1	14	255	256	250	1
7	1	256	1	1	252	256	253	256	256	255	256	1	251	1	254	1
All	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

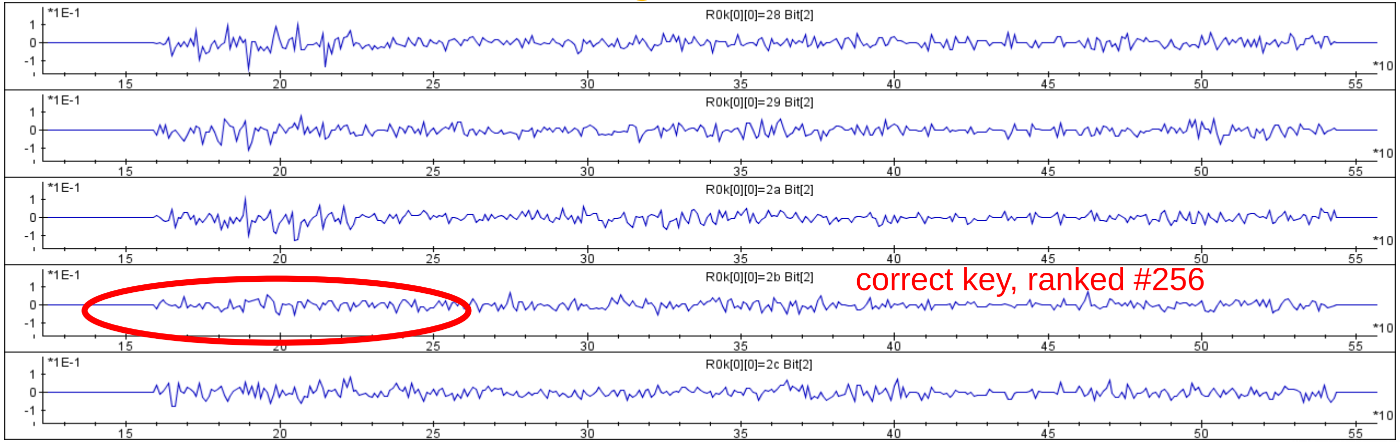


Table 1. DCA ranking for a Karroumi white-box implementation when targeting the output of the *SubBytes* step in the first round based on the least significant address byte on memory reads.

		key byte															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
target bit	**0**	1	256	255	256	255	256	253	1	256	256	239	256	1	1	1	255
	1	1	256	256	256	1	255	256	1	1	5	1	256	1	1	1	1
	2	256	1	255	256	1	256	226	256	256	256	1	256	22	1	256	256
	3	256	255	251	1	1	1	254	1	1	256	256	253	254	256	255	256
	4	256	256	74	256	256	256	255	256	254	256	256	256	1	1	256	1
	5	1	1	1	1	1	1	50	256	253	1	251	256	253	1	256	256
	6	254	1	1	256	254	256	248	256	252	256	1	14	255	256	250	1
	7	1	256	1	1	252	256	253	256	256	255	256	1	251	1	254	1
	All	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2. DCA ranking for a Karroumi white-box implementation when targeting the output of the multiplicative inversion inside the *SubBytes* step in the first round based on the least significant address byte on memory reads.

		key byte															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
target bit	**0**	256	256	1	1	1	256	256	256	254	1	1	1	255	256	256	1
	1	1	1	253	1	1	256	249	256	256	256	226	1	254	256	256	256
	2	256	256	1	1	255	256	256	256	251	1	255	256	1	1	254	256
	3	254	1	69	1	1	1	1	1	252	256	1	256	1	256	256	256
	4	254	1	255	256	256	1	255	256	1	1	256	256	238	256	253	256
	5	254	256	250	1	241	256	255	3	1	1	256	256	231	256	208	254
	6	256	256	256	256	233	256	1	256	1	1	256	256	1	1	241	1
	7	63	256	1	256	1	255	231	256	255	1	255	256	255	1	1	1
	All	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let's check!

WB implementation	Algorithm	#traces
Wyseur challenge, 2007	DES (Chow+)	65
Hack.lu challenge, 2009	AES (Chow)	16 (no encodings)
SSTIC challenge, 2012	DES	16 (no encodings)
Klinec implementation, 2013	AES (Karroumi, dual ciphers)	2000 → 500

Countermeasures?

Academic remedies

- Cannot rely on random data in the white-box attack model
- Use static random data within the white-box itself?
- Use ideas from threshold implementation?
  - masking scheme based on secret sharing and multi-party computation
    S. Nikova, C. Rechberger, and V. Rijmen. Threshold implementations against side-channel attacks and glitches. In Information and Communications Security, 2006.

Practical remedy

- simply strengthen other measures
  - anti-debug / detect DBI frameworks, code-obfuscation (?), integrity checks, platform binding, etc

Side-Channel Marvels

SCA-related projects

https://github.com/SideChannelMarvels

All help to make our DCA / DFA or CPA tools more powerful is highly appreciated!

Deadpool

C ★ 25 ⑂ 6

Repository of various public white-box cryptographic implementations and their practical attacks.

Updated 10 days ago

Tracer

C++ ★ 25 ⑂ 7

Set of Dynamic Binary Instrumentation and visualization tools for execution traces.

Updated on Apr 24

JeanGrey

Python ★ 0 ⑂ 0

A tool to perform differential fault analysis attacks (DFA).

Updated on Apr 18

Orka

★ 4 ⑂ 1

Repository of the official Docker image for SideChannelMarvels.

Updated on Apr 14

Daredevil

C++ ★ 10 ⑂ 4

A tool to perform (higher-order) correlation power analysis attacks (CPA).

Updated on Apr 11

Conclusions and future work

- Software-only solutions are becoming more popular
  - white-box crypto
- Use-cases shifted from DRM to HCE (payment, transit, …)
- Level of security / maturity of many (all?) WB schemes is questionable
  - Open problem to construct asymmetric WB crypto
  - Industry keeps design secret
- DCA is an automated attack which can be carried out without any expertise
  - Counterpart of the SCA from the crypto HW community
- We used DPA, what about FA, CPA, higher-order attacks etc?
  - See our github and the great work by Riscure (see the training this afternoon)

References

- Joppe W. Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen: *Differential Computation Analysis: Hiding your White-Box Designs is Not Enough*. Cryptology ePrint Archive, Report 2015/260, IACR, 2015. (to appear at CHES 2016)
- Eloi Sanfelix Gonzalez, Cristofaro Mune, Job de Haas: *Unboxing the White-Box: Practical Attacks Against Obfuscated Ciphers*. Black Hat Europe 2015.

SECURE CONNECTIONS
FOR A SMARTER WORLD