Differential Computation Analysis

Hiding your White-Box Designs is Not Enough

Joppe W. Bos

Who am I

- ✓ Finished PhD@laboratory for cryptologic algorithms at EPFL, Lausanne, Switzerland under supervision of Prof. Arjen Lenstra in 2012
  - Computational cryptanalysis: RSA768, ECM, ECDLP112, …

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- ✓ Post-doctoral researcher in the Cryptography Research Group at Microsoft Research, Redmond, USA
  - Fully / Somewhat homomorphic encryption: YASHE
  - Fast curve crypto: Genus 2 cryptography

Microsoft Research

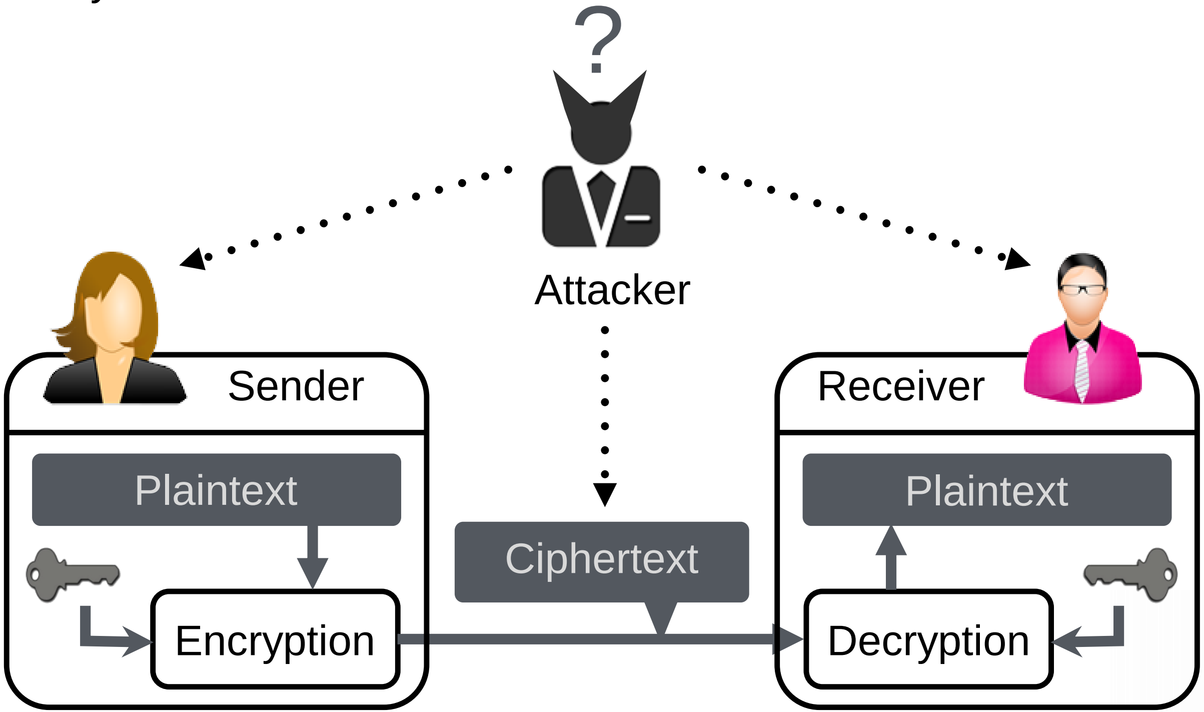
- ✓ From 2014, cryptographic researcher at NXP Semiconductors, Leuven
  - Applied cryptography: elliptic curves and **white-box cryptography**

NXP

NXP Semiconductors

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?

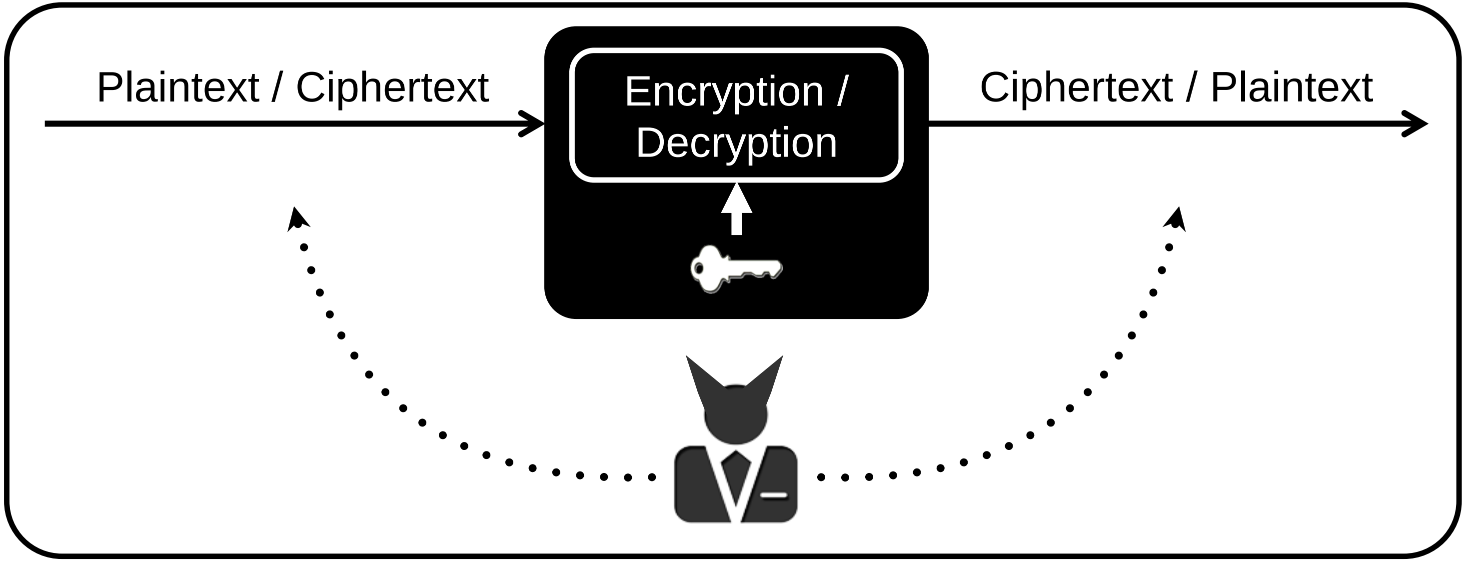


Cryptography & Security Notions

- 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.

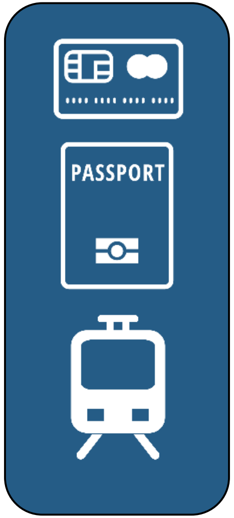
Black box model



Initial cryptographic security model from the 1980s

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

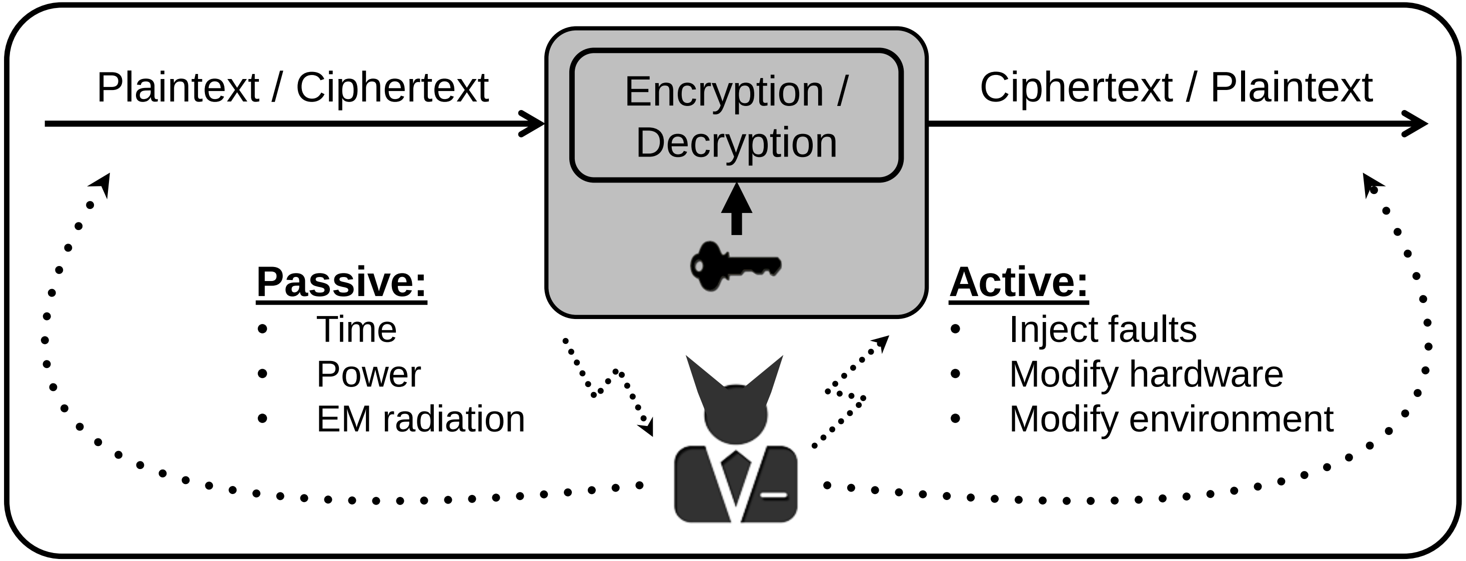
Black box model → grey box model



- 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

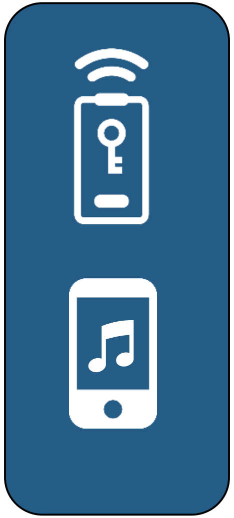
Kocher, Jaffe, Jun. Differential power analysis. In CRYPTO 1999

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.*

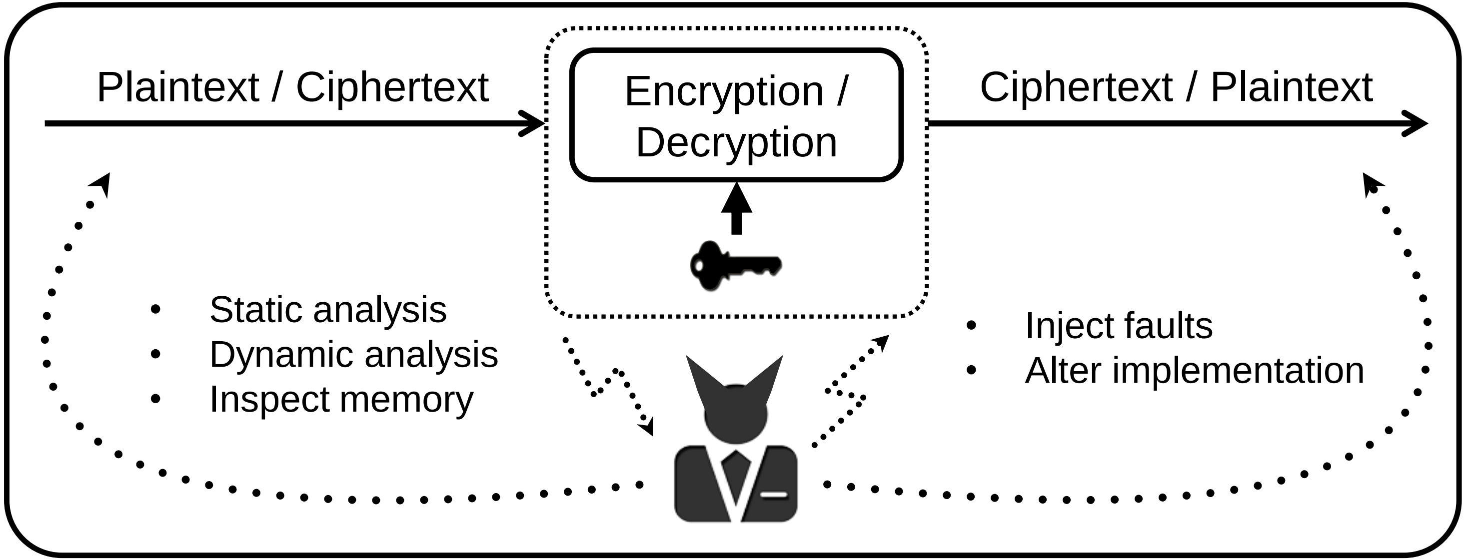
Grey box model → white box model



- 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



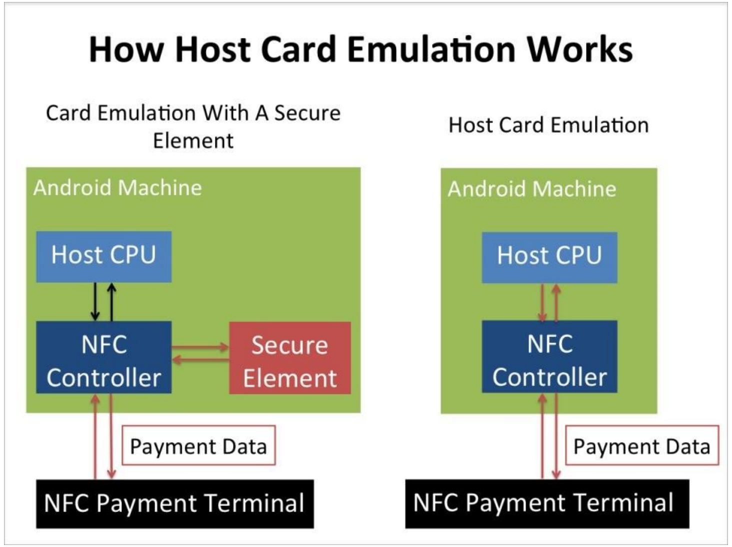
Adversary owns the device running the software. Powerful capabilities

- ✓ 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

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*”

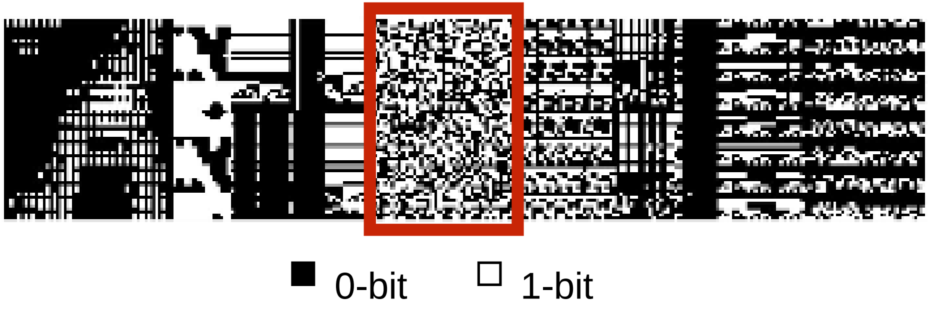
Chow, Eisen, Johnson, van Oorschot. White-box cryptography and an AES implementation. In SAC 2002.

White-Box basic idea

Embed the secret key in the implementation.

White-Box basic idea

Embed the secret key in the implementation.



- **Entropy attack** by Shamir and van Someren (1999)
  - Locate the unusual high entropy of the cryptographic key in a memory dumb using sliding windows for example.

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

Security of WB solutions - Theory

White box can be seen as a form of code obfuscation

- It is known that obfuscation of **any** 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 **any** 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

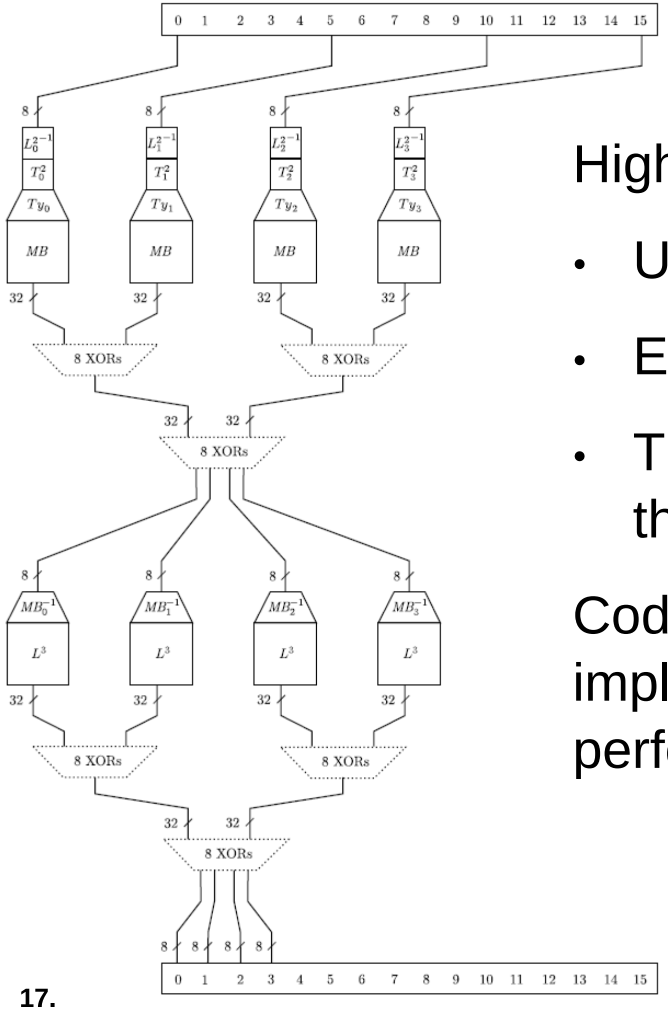
WB Impossible?

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

Practical WB AES?
Network of smaller tables: 752 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.

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



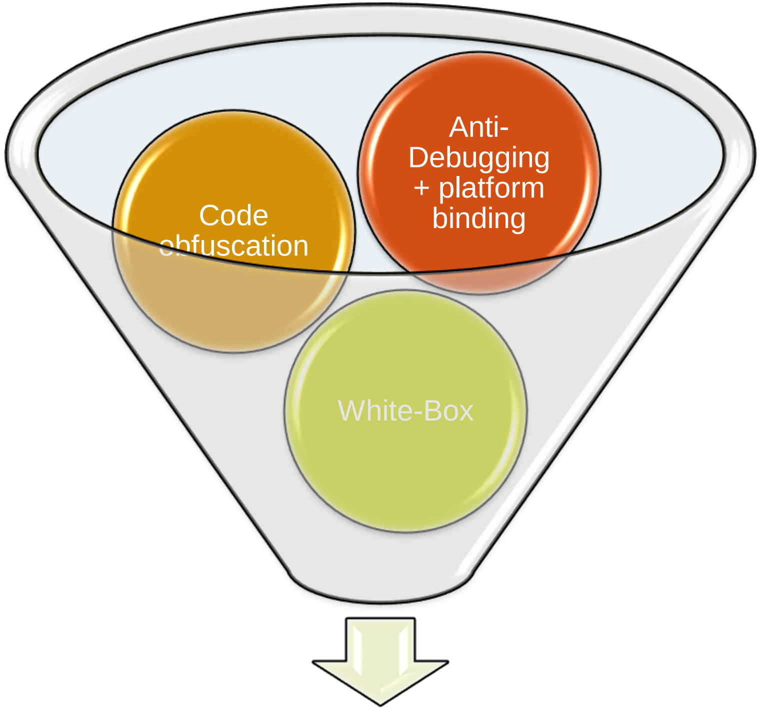
High-level approach

- Use tables rather than individual steps
- Encode tables with random bijections ($g \circ L_i \circ f^{-1}$)
- The usage of a fixed secret key is embedded in these tables

Code generator generates random white-box implementations → lots (10s of thousands) LUTs → performance impact

Figure from: A Tutorial on White-box AES, James A. Muir, Cryptology ePrint Archive, Report 2013/104

White box crypto - practice



White-Boxed implementation

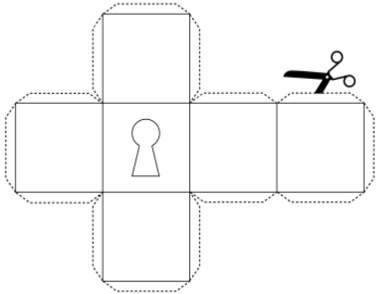
In practice the white box is the most essential but a **small part** of the entire software implementation

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

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
  - High-level of expertise required
  - Need to know which approach is implemented
  - Targets specific s-box or range of LUTs
- Our new attack allows to assess the security of a WB implementation
  - Automatically
  - Without knowledge of the underlying scheme
  - Ignores all (attempts) at code-obfuscation, anti-debugging etc
  - No expertise required

SOFTWARE TRACES

Tracing binaries

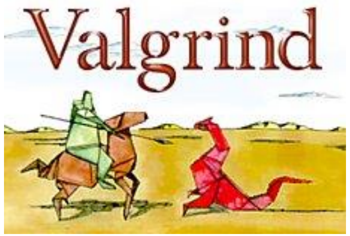
- Academic attacks are on open design
- In practice: what you get is a binary blob

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

- Record all instructions and memory accesses.

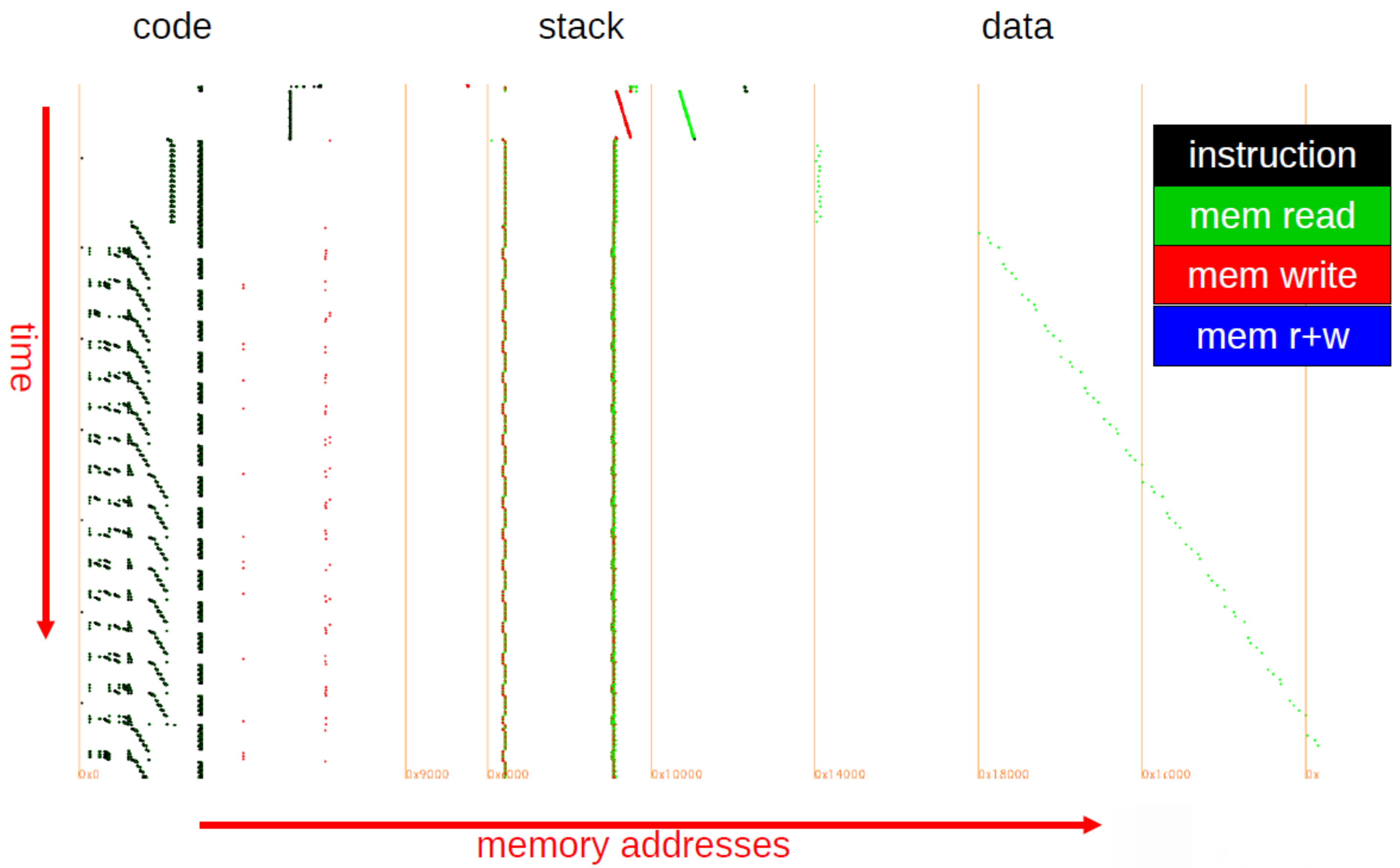
Examples of the tools we extended / modified

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

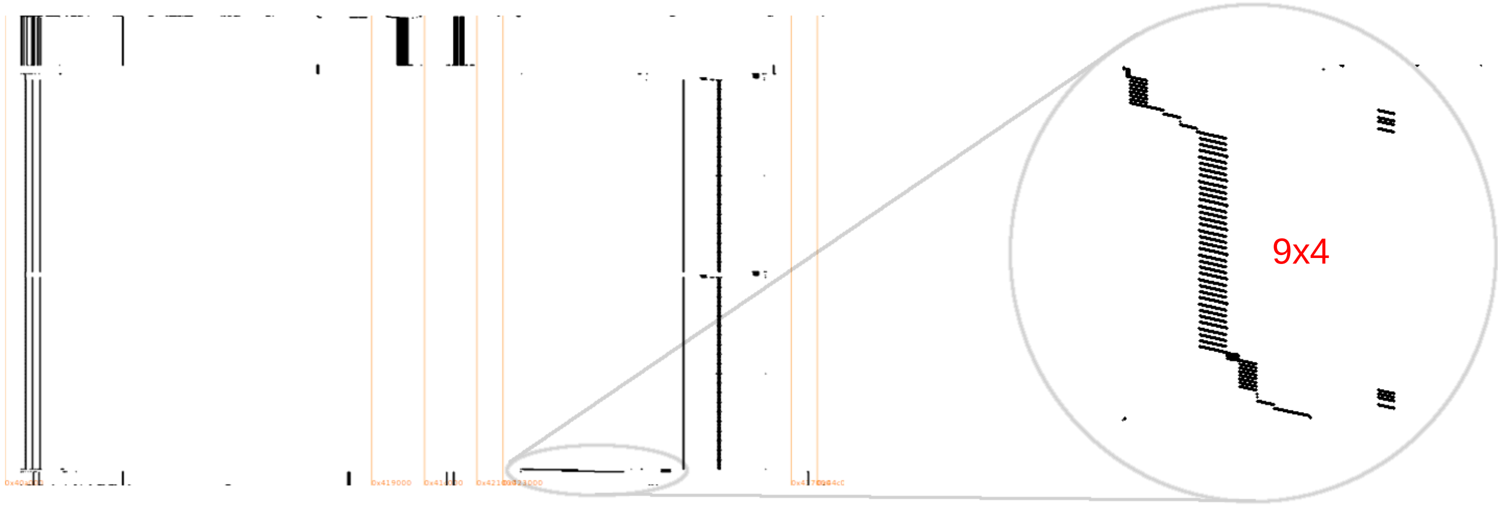


- Using traces:
  1. One trace: Visual identification of white-box, code-/table-lifting
  2. Few traces: data correlation, standard deviation, etc
  3. More traces: DPA-based attack

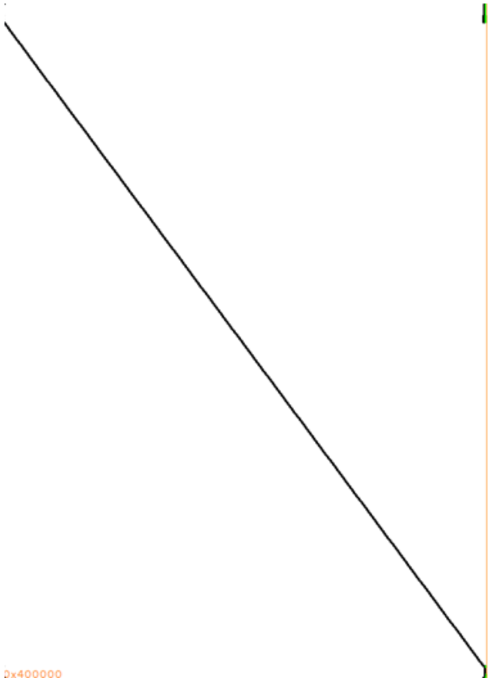
Trace visualization convention: pTra waterfall



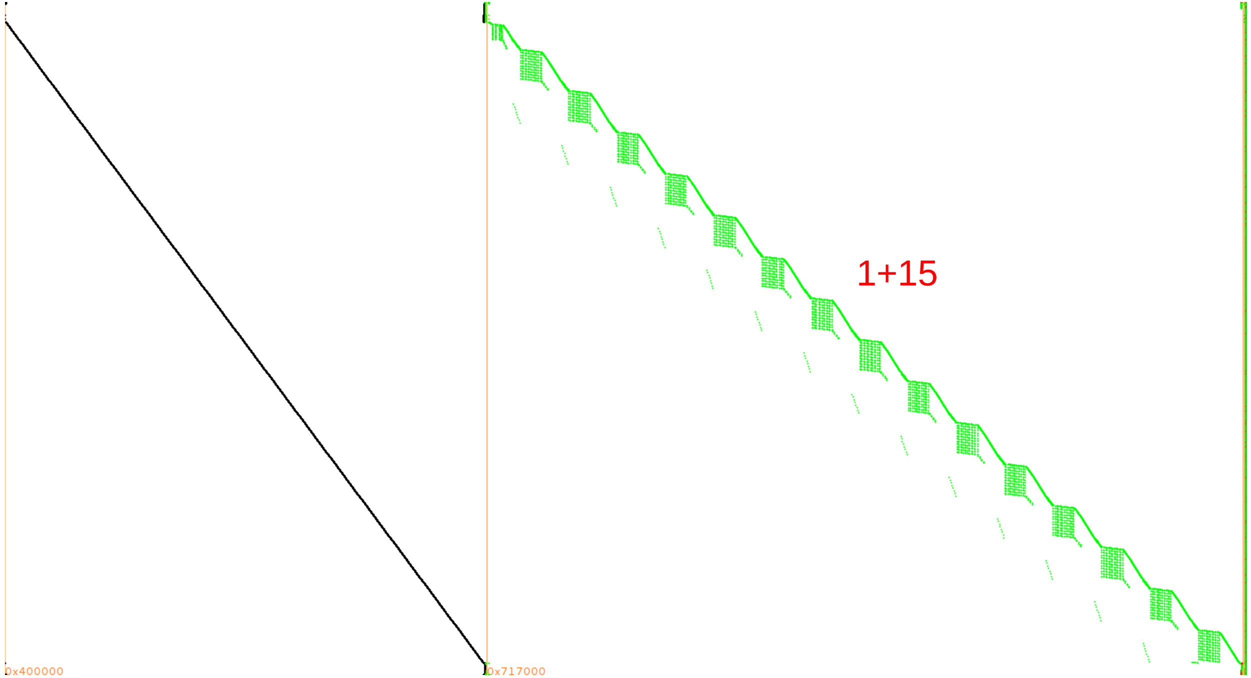
Visual crypto identification: code



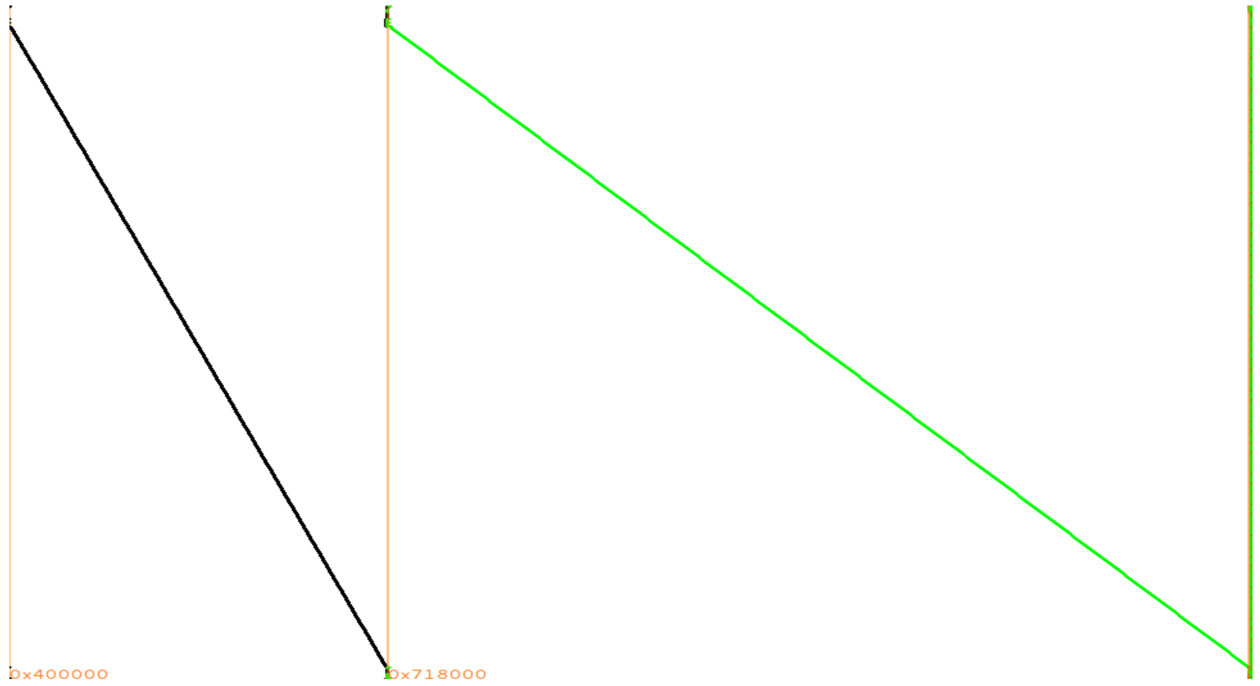
Visual crypto identification: code?



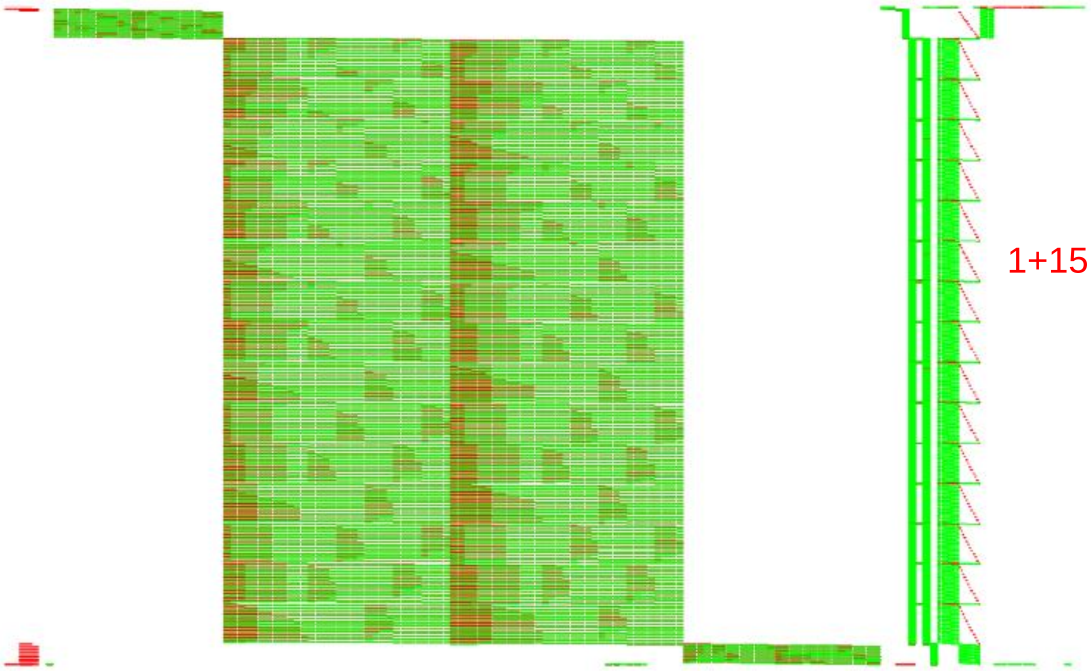
Visual crypto identification: code? data!



Visual crypto identification: code? data?



Visual crypto identification: stack!



1+15

Differential Power Analysis and friends

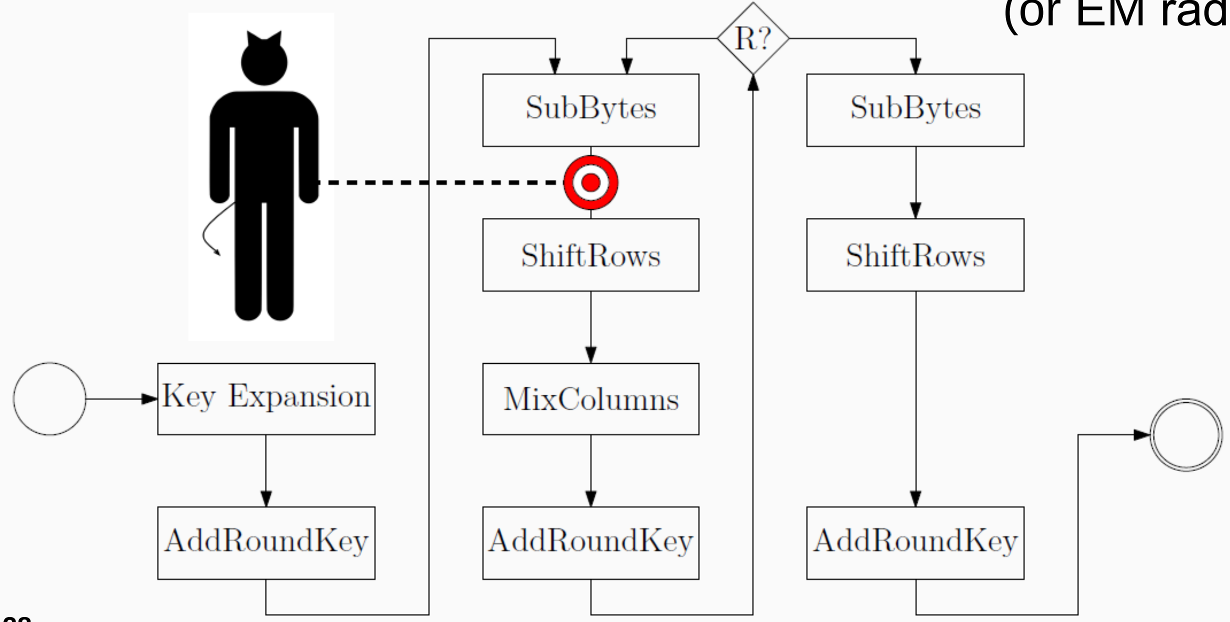
Very powerful grey box attack!

Requirements

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

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

For example in AES: $SubBytes(p \oplus \kappa)$



Differential Computation Analysis

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

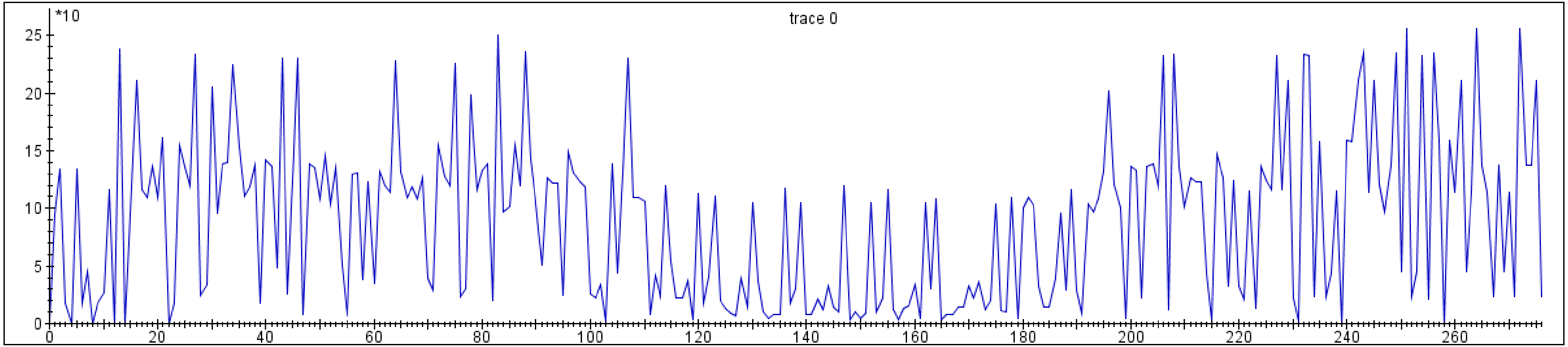
Differential Computation Analysis

~~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:

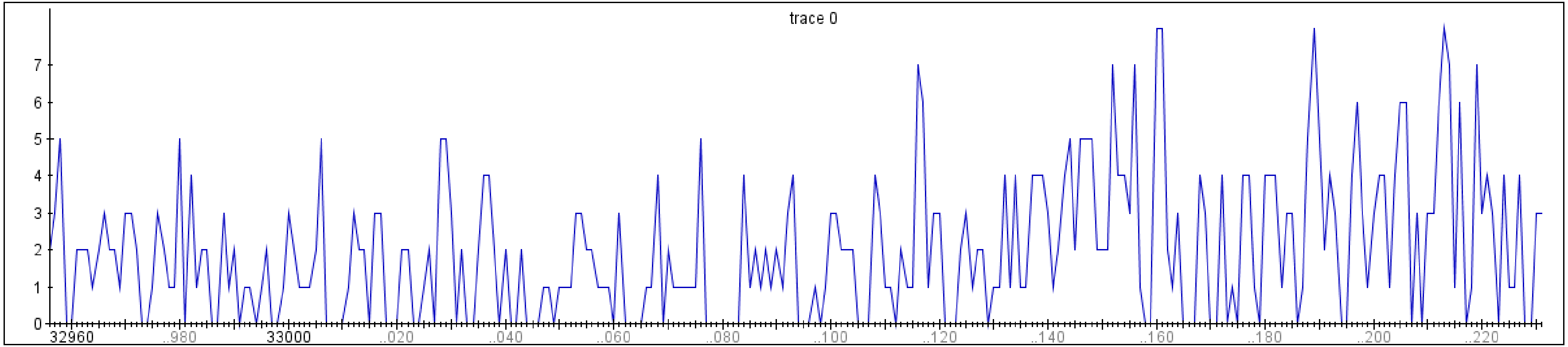


→ 256 possible discrete values

Differential Computation Analysis

256 possible discrete values but bit values dominated by the MSB

→ Build Hamming weight traces?



→ 8 possible discrete values

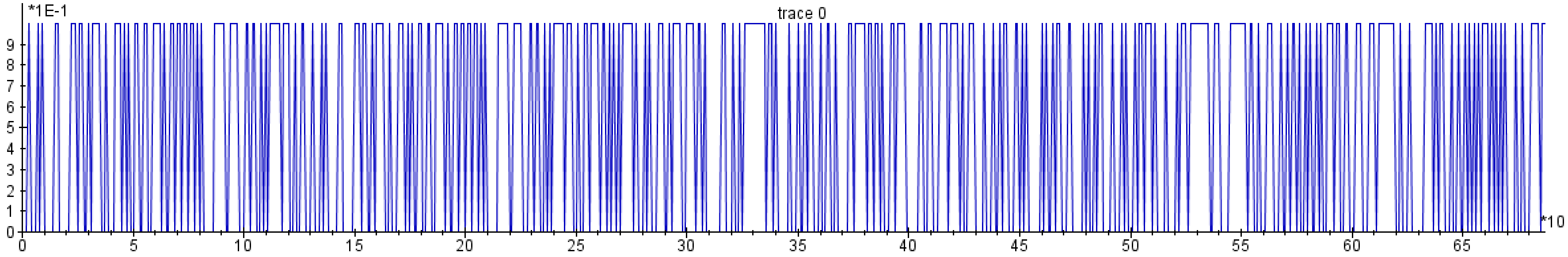
That works but we can do better…

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***

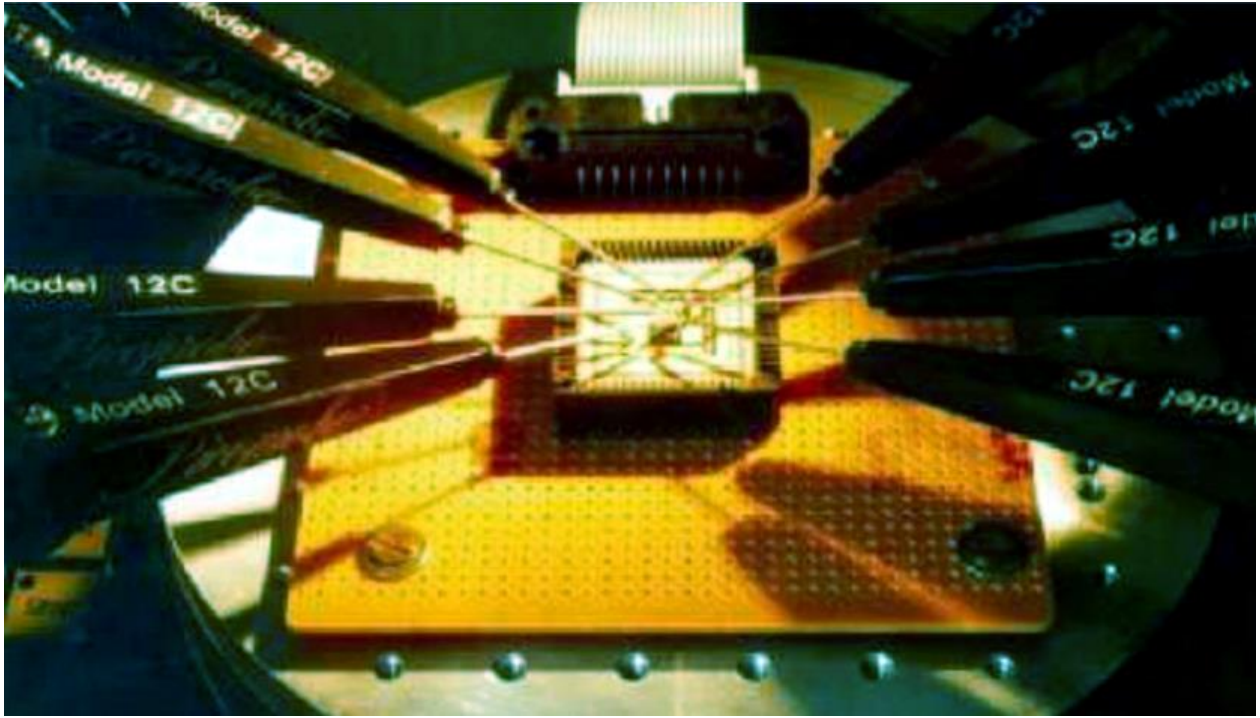


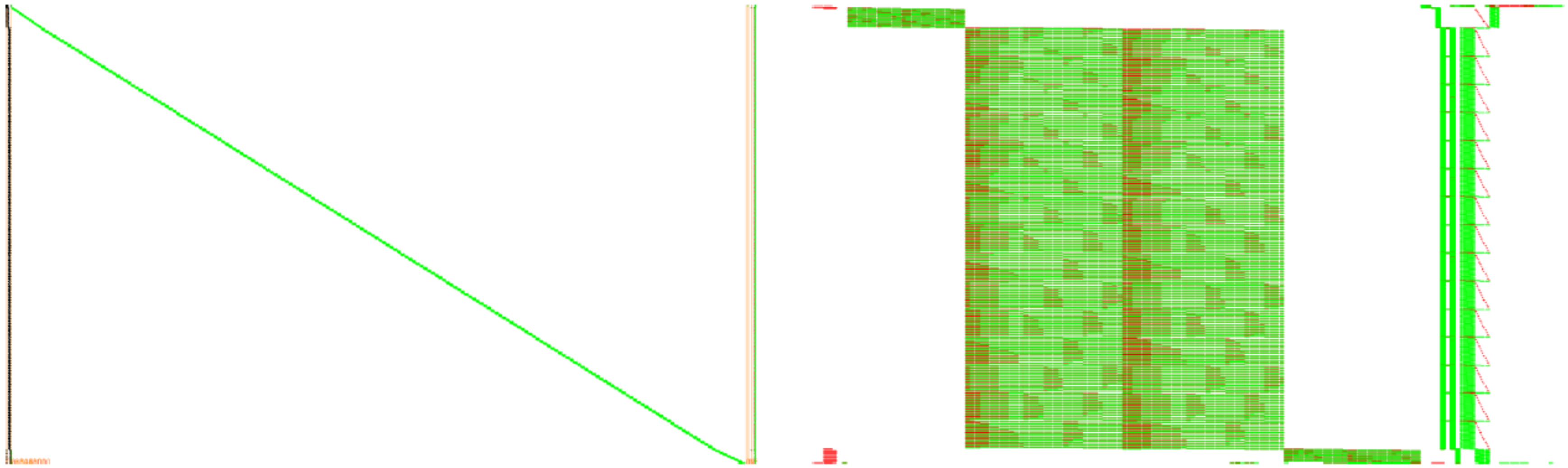
Image source: Brightsight

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)

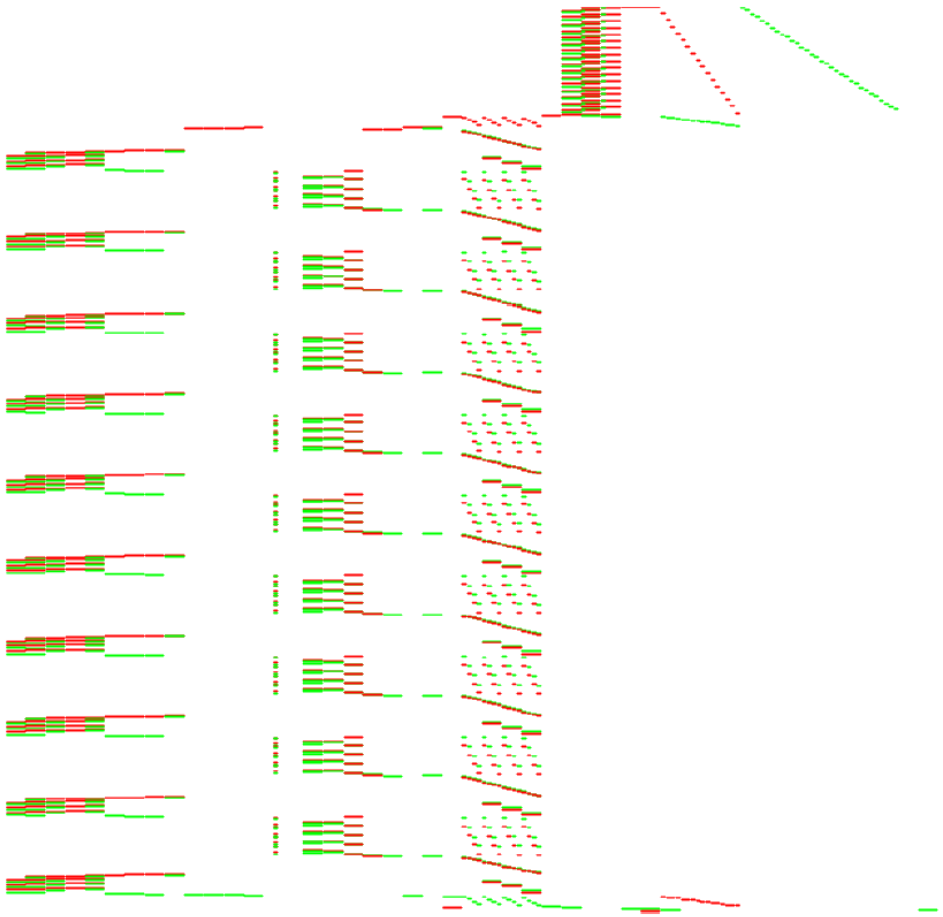
Results

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!

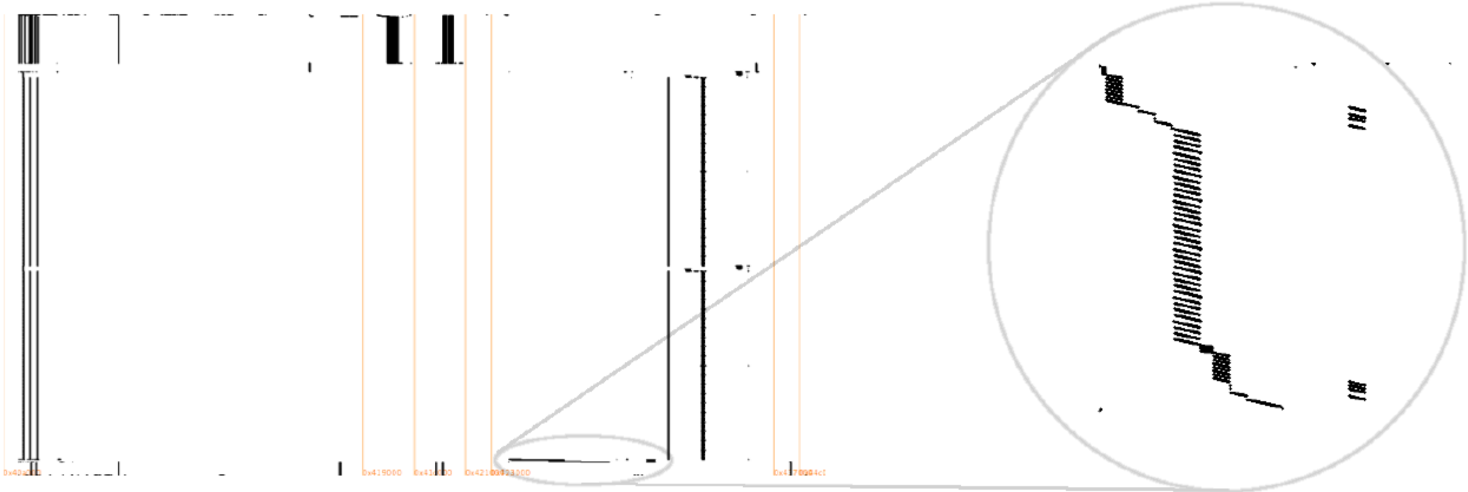
WB implementation	Algorithm	#traces
Wyseur challenge, 2007	DES (Chow+)	65
Hack.lu challenge, 2009	AES (Chow)	16 (no encodings)
SSTIC challenge, 2012		
Klinec implementation, 2013		

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	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

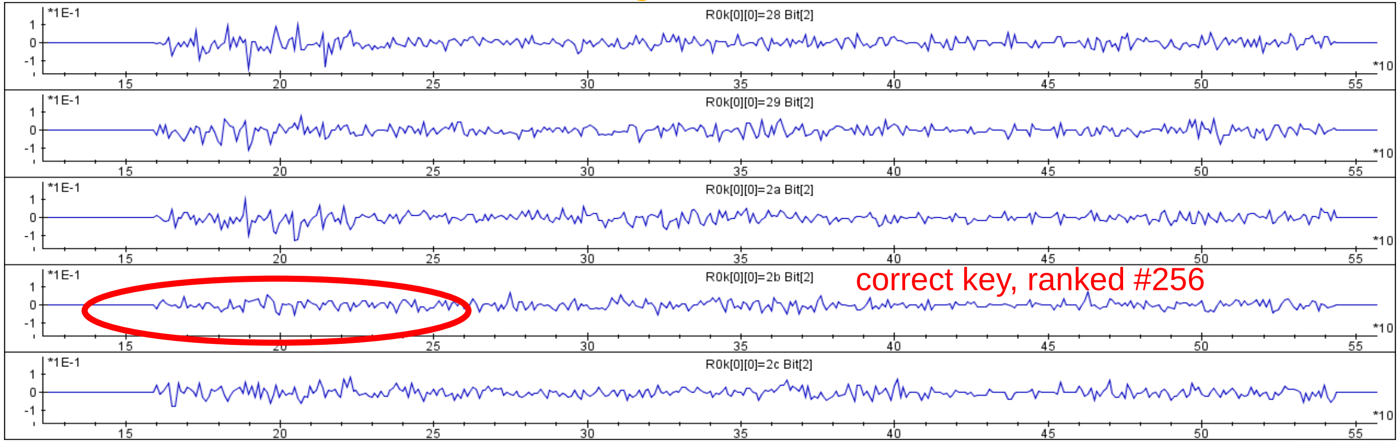


Table 1. DCA ranking for a Karroumi white-box implementation when targeting the output of the *SubBytes* step in the first round based on the least significant address byte on memory reads.

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 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.

target bit	key byte 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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, code-obfuscation, integrity checks, platform binding, detect DBI frameworks, etc

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?

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.
- Eloi Sanfelix Gonzalez, Cristofaro Mune, Job de Haas: *Unboxing the White-Box: Practical Attacks Against Obfuscated Ciphers*. Black Hat Europe 2015.

- Plan to release our DCA tools soon!

SECURE CONNECTIONS
FOR A SMARTER WORLD