Grey-box attacks, four years later

Philippe Teuwen <pteuwen@quarkslab.com>

Quarkslab
www.quarkslab.com

Offline version

Offline notes

This version of the slides has been augmented with comments for a better offline experience.
In the original version, they were part of the oral presentation.
All such comments and additional references will be provided in such orange boxes.
Let me thank the WhibOx 2019 workshop organizers for the kind invitation.
Enjoy!

Table of Contents

Table of Contents

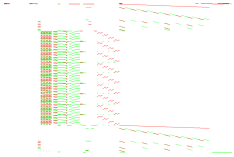
Context

"We'll design whitebox crypto"
"Your're the hacker, try to break it"

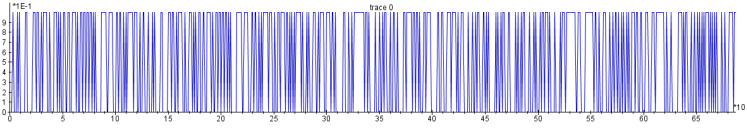
"Whitebox crypto? Full of math I don't understand..."
"Hey, wait, I know a bit side-channel attacks..."

Differential Computation Analysis

- Trace binaries, record memory/regs accesses



- Serialize bits



- Apply regular DPA

Offline notes

Four years ago we introduced the so-called Differential Computation Analysis.
The main trick was to serialize the bits of the traced data or addresses before applying a classical DPA.
See https://eprint.iacr.org/2015/753 for more details.

Requirements

- Access to non-encoded input or output
- Ability to trace e.g. memory accesses (data/addresses)
- Some leakage...
- *No* understanding of the design, *no* big reverse engineering

Offline notes

Some leakage = Something in the traced data correlates somehow with your leakage model, typically the Round1 SBOX output.

Disruptive?

- Key point: bringing back grey-box model in the white-box arena.
- Starting with simple DPA
- Empirical results on several DES & AES white-boxes, based on an intuition
- No real analysis on why it was working

Offline notes

The main contribution was to show to the white-box field that grey-box attack model must be taken into account as well and immunity against such class of attacks wasn't granted for free. Even the simplest side-channel attack demonstrated its potential.

Typical “hacker” point of view

- Attacking “real” code, not paper
- No *need* to understand *at first*, rather proof by PoC
- Not all academics are fine with that approach:
  There is no single formula in the text, [...]
  The claims can not be verified.

Offline notes

I fully understand an absence of mathematical foundations is not acceptable for some academic conferences, but...
claims can't be verified?
While we provided everything to reproduce the results? The data corpus, the tools, the script to automate them. This is quite at the opposite of my own feelings : I cannot verify claims by just looking at formulas. What if the paper lacks some implementation details? Can you detect it without actually trying? Can you detect for sure any error in formulas?
On the other side, with tools, I can reproduce the results by myself.

Table of Contents

Differential Fault Analysis

- Eloi, Cristofaro & Job using Piret & Quisquater 2003 (AES)
- Replay input, see non-encoded output
- Faults injected statically or dynamically before last MC
- 8 “good” faults (on AES-128 enc or dec)
- Blind injection feasible, choose your strategy
- Analysis time: a few seconds

Offline notes

Four years ago, simultaneously, *Unboxing the White-Box*
https://www.blackhat.com/docs/eu-15/materials/eu-15-Sanfelix-Unboxing-The-White-Box-Practical-Attacks-Against-Obfuscated-Ciphers-wp.pdf
(since then a joined paper on DCA & DFA is available at https://eprint.iacr.org/2017/355)

Table of Contents

PoC ⇒ Side Channel Marvels

https://github.com/SideChannelMarvels

Tracer
- TracerGrind
- TracerPIN
- TraceGraph

Deadpool
- White-boxes
- Attacks automation

Daredevil
- Side-channel analysis (CPA)

JeanGrey
- Fault analysis (DFA)

Table of Contents

Table of Contents

DFA in practice

- Fast (no tracing), reliable (no false positives)
- KryptoPlus^{TM} White-box challenge

  Kind of AES-1920
- WhibOx 2017

  Polling website, try auto DFA, try auto DCA, email key, captcha...
  Genuinely surprised so many were falling easily to script-kiddie attacks
  My chall was broken so fast, it can't be DFA?!

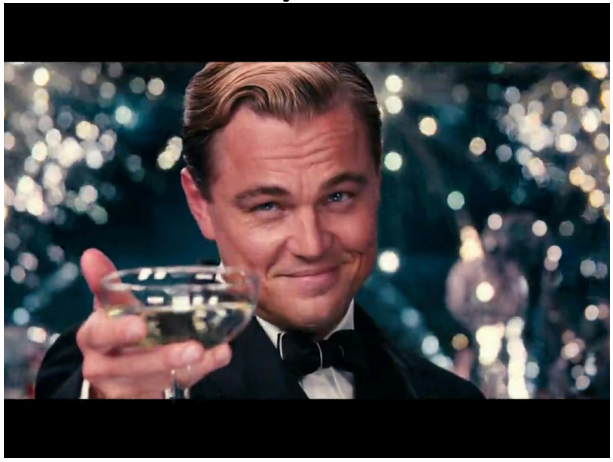
  Lessons learned in 2019?

Offline notes

I really love DFA and usually I try it first, before DCA.
KryptoPlus by Mehdi Sotoodeh was an interesting attempt for whiteboxing an AES256 by replacing the usual keyscheduled round keys by random round keys (so we get $128 * 15 = 1920$ independent key bits). DFA defeats it round by round starting from the last round (see https://github.com/SideChannelMarvels/Deadpool/tree/master/wbs_aes_kryptologik).
WhibOx 2017 (see https://whibox-contest.github.io/): DFA could have been prevented, even in some naive ways, still DFA had a shamefully high success rate on the encountered whiteboxes of the competition. And with the whole chain being automated (beside the Google captcha :/), some were broken in less than 5 min.
(see Alex Treff talk for more stats)

BTW, WhibOx initiatives:

Thank you folks!!



Offline notes

The WhibOx initiatives (workshops and competitions) are instrumental in bringing awareness on the whitebox state-of-the-art and how difficult it is to achieve something robust. The choice of a standardized API and source language make benchmarking and comparisons much easier and the competition helped generating new design attempts.
Why aren't there any WhibOx challenges in the Deadpool repository, you may ask?
I couldn't do it during the competition obviously and I must confess I was quite exhausted after months of competitions and moved to other topics, but anybody can contribute! This is not my *personal* repo, it's a repo for everybody willing to search on the subject, please contribute!

DFA in practice, it's also

News: Widevine L3 DRM broken
"scarily trivial to pull off" with the help
of the Side-Channel Marvels project
- David Buchanan
No blame there, what can you do with so limited size and cycles?

Offline notes

This is what happens when you provide tools: academic research getting used for practical purposes (David didn't publish his attack actually).
Honestly, I don't blame the vendor here. WhibOx competition has shown how difficult it is to build a robust AES whitebox even with very relaxed constraints (50 Mb source code, one whole second per AES block!) and DRM context requires handling HD video, xMb/s stream encrypted with some MPEG Common Encryption standards imposing AES, usage of dynamic keys and still, you need some CPU budget left to do some collateral tasks such as handling video codecs...
To quote a colleague, using SideChannelMarvels tools in this context was already like using a nuclear weapon to kill a fly.

DFA in SideChannelMarvels today

- Support for injection one round earlier
- 2 “good” faults in 8^{th} round (before last two MC)
- Analysis time down to 50 ms (thanks *kanarvc*)
- Blind injection scalable

Offline notes

So the DFA tool was extended to support injecting faults one round earlier.
An anonymous contributor fixed my Python code and gave it a speed boost, thanks to her/him!
Because analysis is now so fast, you can inject faults with less control and just try it out.

Table of Contents

Yifan Lu on PlayStation Vita

- Broke the 30 hw master keys of PSVita including the 0x208 key ...with ChipWhisperer and SideChannelMarvels DFA
- Why? Was it the best tool?
- No, just guided by lack of DFA tools:
  Jovanovic's implementation of Tunstall's single fault (2009, $\mathcal{O}(2^{32})$)
  SideChannelMarvels implementation (based on Piret, 2003)
- Tweaked practical attack to relax 2-fault hypothesis
- Published back his tools as opensource

Offline notes

Yifan broke the PS Vita keys end of last year. (see https://yifan.lu/2019/02/22/attacking-hardware-aes-with-dfa/) What strikes me is why he used SideChannelMarvels. The explanation is very representative of the situation today: Despite the numerous papers on DFA, when he looked for *decent* tools, he found only Philipp Jovanovic's implementation of Michael Tunstall and Debdeep Mukhopadhyay (single fault DFA, a very interesting attack but requiring a bruteforce phase of $\mathcal{O}(2^{32})$) and... the SideChannelMarvels implementation. He made practical adaptations, explained everything in a paper (https://arxiv.org/abs/1902.08693) and in return published his modified version of our DFA (https://github.com/TeamMolecule/f00dsimpleserial/tree/master/scripts/dfa_crack) together with all pieces needed to *reproduce* his results. Very nice!

Publishing tools is IMHO essential

- for the security/hackers community and for the industry
- even if just a PoC

Why publishing tools?

- Much more likely to be used out of academia than only a paper
- Easier to convince non-academics
- Reproducibility, a basis for sciences, remember?
- Better something not perfect than nothing

$\Rightarrow$ Helps bridging the gap between academia & rest of the world

$\Rightarrow$ Paper and tool complementary, beware of lack of paper as well...

Offline notes

Ok maybe it's not a digression but the main point of my talk ;)

- Not everybody is ready to spend a week implementing a tool from a paper he's reading just to see if it would be adequate for the needs of his current work and it's a pity if it becomes a burden. Moreover, tools help raising awareness out of academia: there is no such thing as a cracking demo, with its *wow effect*.
- There are people, like me, who understand better things by looking at source code rather than paper formulas.
- You can hide implementation details in a paper... not in an implementation. Reproducibility, a basis for sciences: I know we lost a bit that habit in a field of attacks of impractical order...
- A PoC can at least serve as a test reference for developing better versions, it will always be better than having to start from scratch. I blame much more an absence of tools than tools of bad quality.

$\Rightarrow$ Publishing tools helps bringing practical considerations and concerns

$\Rightarrow$ Obviously we still need paper too: there are a lot of scientific advances in security which are only available through some tools & blogposts, which is much less persistent and referable than an academic article.

Table of Contents

DCA in academia

Many very good research papers!

- Bringing math foundations to the attack (I trust you...)
- Generalizing the attack (beyond sbox output high corr.)
- Accelerating the attack (e.g. trace compression)
- ...

Offline notes

I didn't take the risk of forgetting some by enumerating the recent papers (see https://github.com/SideChannelMarvels/Deadpool/wiki/DCA-related-literature) but you get a very good overview in Andrey Bogdanov's talk and the most recent ones are presenting during this WhibOx edition (see https://www.cryptoexperts.com/whibox2019/)

Tools!

- Hulk: bruteforcing missing bytes, with AES-NI
- conditional-reduction: sample reduction for DCA
- Jlsca: DPA in Julia
- qscat: Qt SCA tool
- White-box Algebraic Security:
  PoC for Attacks and Countermeasures for WB designs
- On Recovering Affine Encodings in WB Implementations
- DATA - Differential Address Trace Analysis
- Lascar - Ledger's Advanced Side Channel Analysis Repository
- Rainbow - It makes unicorn traces

Offline notes

- Hulk: https://github.com/pgarba/Hulk
- conditional-reduction: https://github.com/ikizhvatov/conditional-reduction
- Jlsca: https://github.com/Riscure/Jlsca
- qscat: https://github.com/FdLSifu/qscat.git
- White-box Algebraic Security: https://github.com/cryptolu/whitebox
- On Recovering Affine Encodings: http://wbcheon.gforge.inria.fr/
- DATA: https://github.com/Fraunhofer-AISEC/DATA
- Lascar: https://github.com/Ledger-Donjon/lascar
- Rainbow: https://github.com/Ledger-Donjon/rainbow

Table of Contents

Table of Contents

DCA can fail, yeah

.	key byte															
target bit	**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
key recovery	ok	ok	ok	ok	ok	ok	fail	ok	ok	ok	ok	ok	ok	ok	ok	ok

Offline notes

This table is from the initial paper, showing that on Chow-like implementations with proper encodings, some target bits might not leak, like here the key byte 6.
Since then, we've even seen commercial white-boxes carefully choosing their encodings such that the simple DPA on SBOX output doesn't leak at all! But using other target such as the multiplicative inverse gave the usual partial results.

AES SBOX: multiplicative inverse, then

- $c(x) = \texttt{0x1F}(x) \cdot b(x) + \texttt{0x63}\ (mod\ x^8 + 1)$
- What if other affine transformations are added?
- Skip fix term, skip rotations
- We end up with 35 possible targets for classical DCA
- Alternatively, just look at all $2^8 - 1$ bit combinations

Jakub Klemsa thesis

Offline notes

Besides the SBOX affine transformation, any other one can lead to interesting results. All affine transformations can be reduced to 35, the other ones leading to redundant results. Actually, looking at individual output bit, it's equivalent to looking at all the 255 possible linear combinations of the multiplicative inverse step.

Generalizing DCA

White-Box Cryptography in the Gray Box: a boolean function is called balanced m-th order correlation immune if and only if its Walsh transform values satisfy $W_f(\omega) = 0$ *for* $0 \leq HW(\omega) \leq m$

- Try all the possible combinations of sbox output bits
- Enumerate plaintexts (one byte while fixing the other ones)

Generalized DCA $\Leftrightarrow W_{f_k[j]}(\omega)\ \forall \omega : 0 < HW(\omega) \leq 8$
for each sample j and each key candidate k

Offline notes

White-Box Cryptography in the Gray Box was one of the first papers trying to explain the DCA. BTW they had an interesting approach of implementing a white-box in a FPGA and looking for *physical* side-channel leakages. This sounds maybe a bit strange but it was a question the industry was looking at. Indeed, if a white-box is perfect, it's also grey-box-proof by definition, isn't it?
They provided that condition for being leakage-resistant.
Actually, if we use the generalized DCA (over all 255 ω) and enumerate plaintexts rather than using random ones, it can be demonstrated that the attack is equivalent to performing all the Walsh transforms from first order to eighth (maximal) order in one shot, for each sample and each key candidate.

Generalizing DCA

$$\Delta_{k,\omega}[j] = \frac{\sum_{i=1}^{m}(x_i \cdot \omega)\mathbf{T}_i[j]}{\sum_{i=1}^{m} x_i \cdot \omega} - \frac{\sum_{i=1}^{m}(1-x_i \cdot \omega)\mathbf{T}_i[j]}{\sum_{i=1}^{m} 1-x_i \cdot \omega}$$

$$\text{with } x_i = S(p_i \oplus k)$$

$$= \frac{1}{256} \sum_{x \in \{0,1\}^8} (-1)^{f_k(x)[j] \oplus x \cdot \omega}$$

$$\text{with } f_k(x)[j] = \mathbf{T}_{S^{-1}(x) \oplus k}[j]$$

$$= \frac{1}{256} W_{f_k[j]}(\omega)$$

Generalizing DCA

E.g. for a Chow whitebox without non-linear 4-bit encoding: the leaks have the highest possible correlation $max\left(|\Delta_{k,\omega}[j]|\right) = 1$.

Relaxing hypotheses in practice

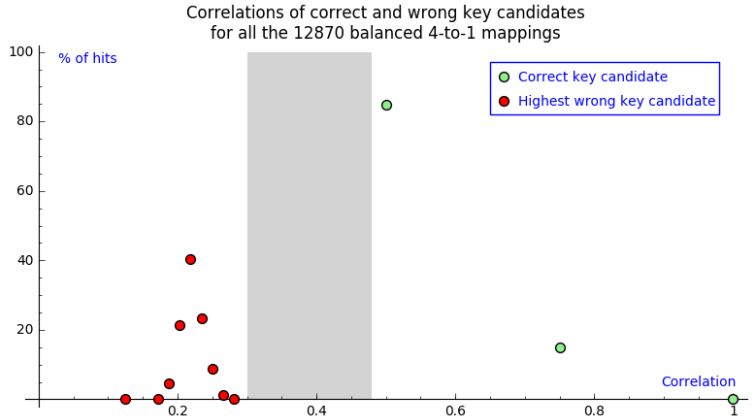
- Enumerating 256 plaintexts $\rightarrow$ random choice of about 30
- Remember, simple DCA:
  8 ω, 32 output bits: $P(X > 0) \approx 0.64$
- generalized DCA:
  255 ω, 32 output bits: $P(X > 0) = 1$
- 34 ω, 32 output bits: $P(X > 0) \approx 0.99$
- 21 ω, 32 output bits: $P(X > 0) \approx \frac{15}{16}$

255×256 (full Walsh) $\rightarrow 34 \times 30$ in practice $\rightarrow 64\times$ faster

Offline notes

$P(X > 0)$: probability of at least one leak among these 32 output bits
So we can reduce the number of ω, choosing randomly e.g. 34 of them to still achieve a success rate of 99%.

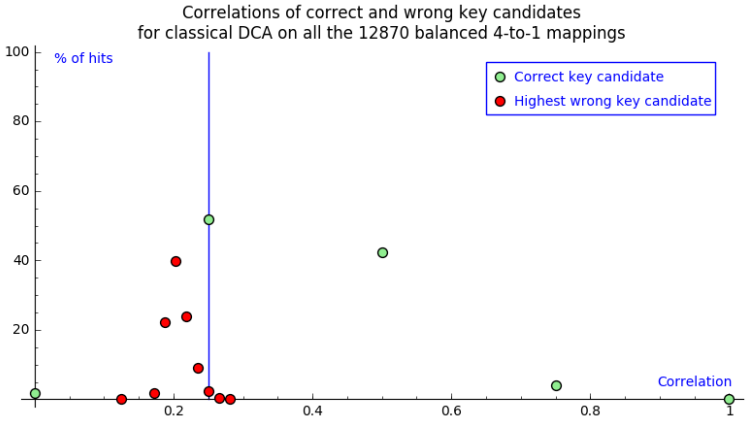
Effect of non-linear encodings



Offline notes

Looking at the local 4-bit non-linear encodings. In total there are 16! (= a lot) possible look-up tables but if we look at the $f_i(x)$ individually, there are 12 870 ways to map 4 bits to 1 bit with balance (i.e. the Hamming weight of the mapping of the $2^4 = 16$ possible values being equal to 8). It they weren't balanced, they could not be combined in a 4-bit to 4-bit permutation encodings. And we can exhaustively check these 12 870 4-to-1 mappings. We see that with the generalized DCA they all stand clearly above wrong key candidates, with peaks exactly equal to 0.5, 0.75 or 1. Therefore the attack can be performed on any single output bit of the non-linear encodings.

If using classical DCA



Offline notes

If, instead of applying the generalized DCA, we limit ourselves to the classical 8 ω values of a DPA, some of the correlations will be equal to 0.25 or even 0. In *On the Ineffectiveness of Internal Encodings - Revisiting the DCA Attack on White-Box Cryptography*, the authors show that in such case, it's possible to use the discrete values of the correlation to recover the key, even if it's not the highest amongst candidates. But things can be more complex as some wrong candidates can also have a correlation of 0.25, we'll come to that later...

Take-away from these 4 years

- Very happy to witness this grey-box in white-box maturity
- Very impressed by latest papers & tools
- Very thankful to WhibOx initiatives (workshops & competitions)
- Publish papers *and tools*, contribute to tools, we need them!

Offline notes

Your take-away:
Publish your tools!

Questions?

Quarkslab

SECURING EVERY BIT OF YOUR DATA