

rhme3 whitebox prequal

Hi I'm Jakob

- wanted to do a crypto challenge
- wanted to attack a whitebox for a while

- Narrator: “but it was a bad idea”

challenge

```
🐱 jakob@carbon(11:08) ❤️  
(~/CTF) ~> file whitebox_original | sed -e "s/, /\n/g"  
whitebox_original: ELF 64-bit LSB executable  
x86-64  
version 1 (SYSV)  
dynamically linked  
interpreter /lib64/ld-linux-x86-64.so.2  
for GNU/Linux 2.6.32  
BuildID[sha1]=417f6c8e54c3c7ebf8f76772faae13bdb78ddbce  
stripped  
🐱 jakob@carbon(11:08) ❤️  
(~/CTF) ~> ./whitebox_original  
usage:  
  method a) ./whitebox_original [16 bytes plaintext]  
  method b) ./whitebox_original --stdin
```

~DEMO~

(ノ◡ノ)ノ*:° ✨

[DEMO fallback]

(;~_~)

```
git clone git@github.com:SideChannelMarvels/Orka.git
cd Orka
docker build -t marvelsbase ./marvelsbase/
docker build --no-cache -t scamarvels ./marvelslatest
./docker_enter.sh
cd Deadpool/wbs_aes_rhme3_prequal/DFA/
./mount_tmp_and_attack.sh # gives last round key
~/Stark/aes_keyschedule 4E44EACD3F54F5B54A4FB15E0710B974 10
# K00: 61316C5F7434623133355F525F6F5235
# it's a11_t4b135_R_oR5
```

What in the sweet hell just happened

1. whitebox crypto
2. but that's just used in ctfs, right?
3. DFA crashcourse
4. lessons and practical steps



AES (very roughly)

- `SubBytes(state)`
- `ShiftRows(state)`
- `MixColumns(state)`
- `AddRoundKey(state,kr)`

AES

- SubBytes(state)
- ShiftRows(state)
- MixColumns(state)
- AddRoundKey(state,kr)
- ShiftRows(state)
- AddRoundKey(state,k'r-1)
- SubBytes(state)
- MixColumns(state)

AES

- SubBytes(state)
- ShiftRows(state)
- MixColumns(state)
- AddRoundKey(state,kr)

- ShiftRows(state)
- AddRoundKey(state,k'r-1)
- SubBytes(state)
- MixColumns(state)

Combine into Lookup table,
hardcode

AES

- SubBytes(state)
- ShiftRows(state)
- MixColumns(state)
- AddRoundKey(state,kr)

- ShiftRows
- TBoxesTyiTables
- XORTables

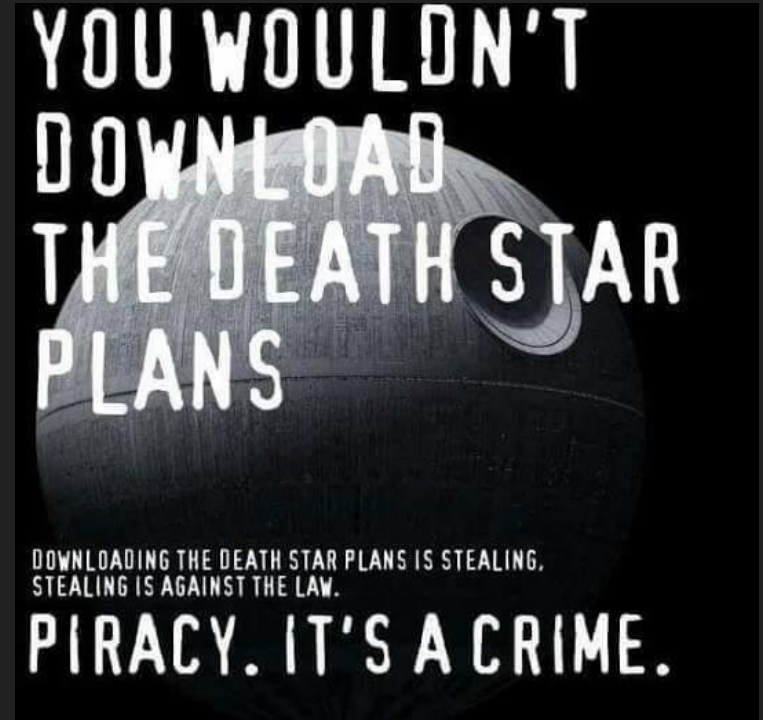
- ShiftRows(state)
- AddRoundKey(state,k'r-1)
- SubBytes(state)
- MixColumns(state)

Combine into Lookup table,
hardcode

It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.

DRM & “Raubmordkopiererei”

- Short-lived keys
- Different levels for different Quality
- Widevine L3 uses whitebox crypto
- Challenge is lifting the code & automation



- 
- David Buchanan** @David3141593 · Jan 3
- Soooo, after a few evenings of work, I've 100% broken Widevine L3 DRM. Their Whitebox AES-128 implementation is vulnerable to the well-studied DFA attack, which can be used to recover the original key. Then you can decrypt the MPEG-CENC streams with plain old ffmpeg...
- 42 804 2.2K
- 
- David Buchanan** @David3141593 · Jan 3
- [#fulldisclosure](#)
- 2 3 84
- 
- David Buchanan** @David3141593 · Jan 3
- plz no lawsuit
- 1 17 268
- 
- David Buchanan** @David3141593 · Jan 3
- Huge thanks to [@doegox](#) and the Side-Channel Marvels project for making this attack scarily trivial to pull off.
- 1 5 98

See <https://twitter.com/David3141593/status/1080616099174141952>

WHO WOULD WIN?



NETFLIX

HBO

Disney

hulu

Jio

DIRECTV



prime video

SHOWTIME



slings TELEVISION

facebook

A FEW FLIPPY BOIS

Differential fault analysis(DFA)

After ShiftRow9	Fault injected '1E'	After Mixcolumn	K ₉
87 F2 4D 97	99 F2 4D 97	7B 40 43 4C	AC 19 28 57
6E 4C 90 EC	6E 4C 90 EC	29 D4 70 9F	77 FA D1 5C
46 E7 4A C3	46 E7 4A C3	8A E4 3A 42	66 DC 29 00
A6 8C D8 95	A6 8C D8 95	CF A5 A6 BC	F3 21 41 6E

After AddRoundKey9	After SubBytes10	After ShiftRows10	K ₁₀
07 59 8D 1B	0E CB 3D AF	0E CB 3D AF	D0 C9 E1 B6
5E 2E A1 C3	58 31 32 2E	31 32 2E 58	14 EE 3F 63
EC 38 13 42	CE 07 7D 2C	7D 2C CE 07	F9 25 DC DC
3C 84 E7 D2	EB 5F 94 B5	B5 EB 5F 94	A8 89 C8 A6

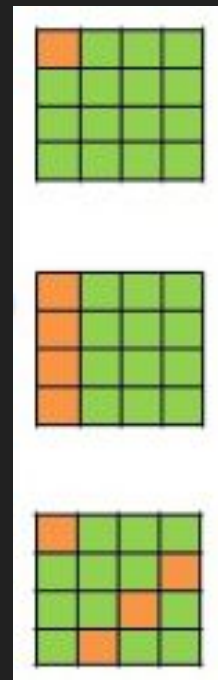
Output with faults	Output without fault	Error
0E 02 DC 19	39 02 DC 19	E7 00 00 00
25 DC 11 3B	25 DC 11 6A	00 00 00 51
84 09 C2 0B	84 09 B5 0B	00 00 47 00
1D 62 97 32	1D 69 97 32	00 99 00 00

5FA 308001319A42E07072A
 51A28A8D2AA8B710880C4F3C
 021C0F80C118F979A0B 32

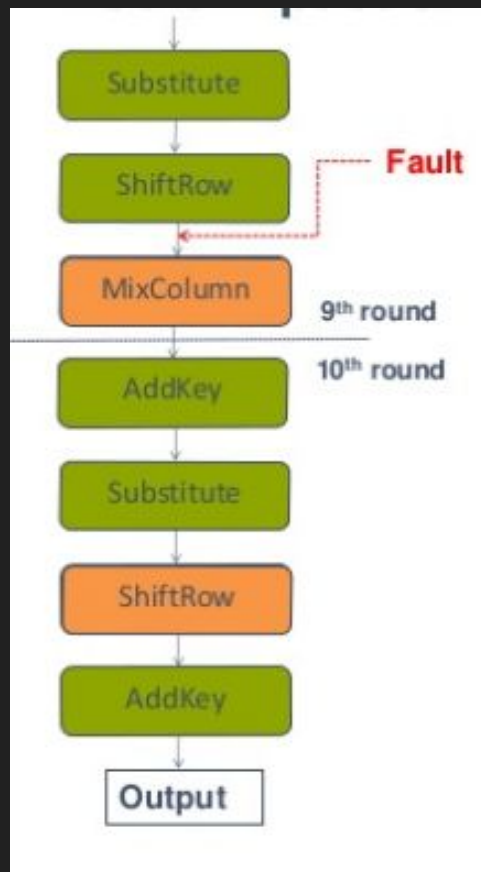
Differential Fault Analysis



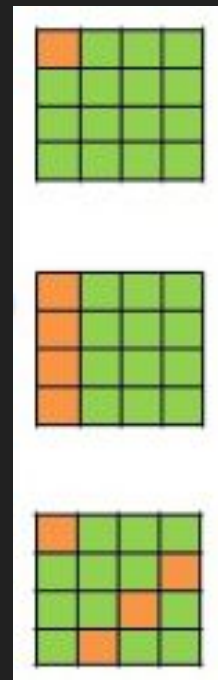
1. Inject fault before 9th Round MixColumn
2. MixColumn propagates fault in Column
3. AddKey changes Faults dependent on Value+Key
4. Substtute changes Faults depending on Value
5. ShiftRow propagates to 4 Cells
6. AddKey changes Faults depending on Value+Key



Differential Fault Analysis



1. Inject fault before 9th Round MixColumn
2. MixColumn propagates fault in Column
3. AddKey changes Faults dependent on Value+Key
4. Substitute changes Faults depending on Value
5. ShiftRow propagates to 4 Cells
6. AddKey changes Faults depending on Value+Key
7. MATH!
 - $ISB(x_1+K_1)+ISB(x_1+F_1+K_1)=$
 $2[ISB(x_2+K_2)+ISB(x_2+F_2+K_2)]$
 - $ISB(x_2+K_2)+ISB(x_2+F_2+K_2)=$
 $ISB(x_3+K_3)+ISB(x_3+F_3+K_3)$
 - $ISB(x_4+K_4)+ISB(x_4+F_4+K_4)=$
 $3[ISB(x_2+K_2)+ISB(x_2+F_2+K_2)]$











Differential Fault Analysis

- Cool, so it's a Hardware-Attacks in Software!
- Same challenges:
 - Needs a lot of specialized tools and even then doesn't always work

The WhibOx Contest Edition 2 - CHES 2019 CTF 👤

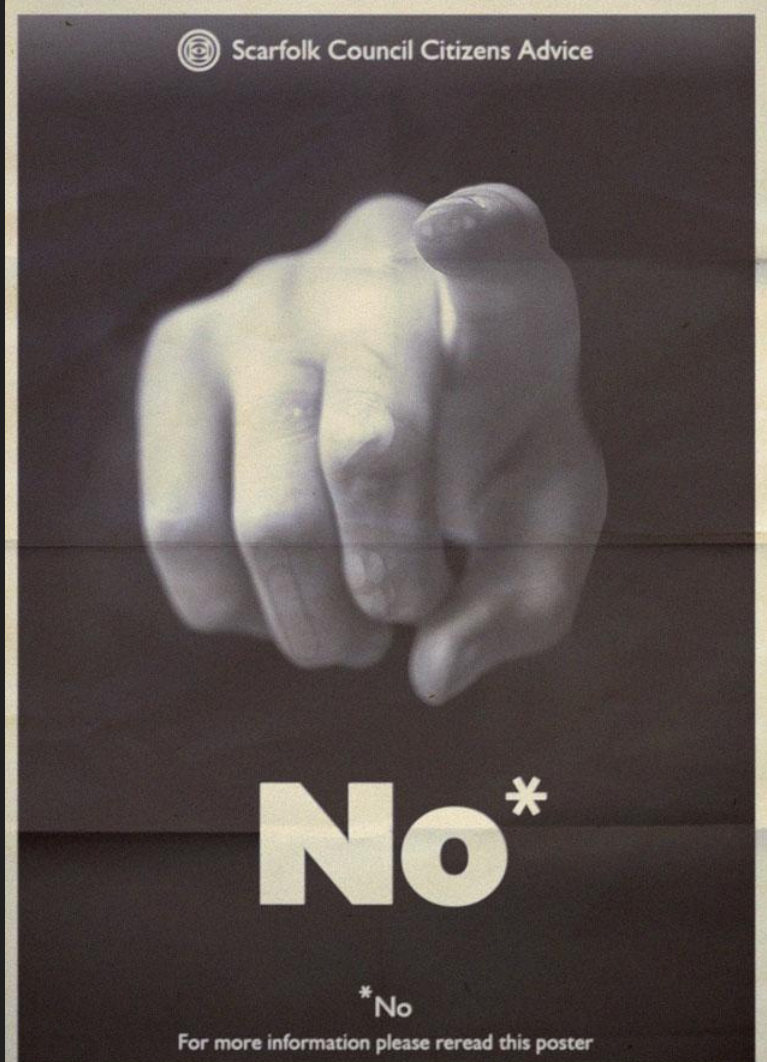
- Dashboard
- Your Dashboard
- Submit a Challenge
- Competition Rules
- Terms and Conditions
- Create an Account
- Sign in

Dashboard

 65 Users!	 27 Challenges!	 3 / 3 Unbroken! / Uninverted!	 124 / 23 Breaks! / Inversions!
Scroll to Details 	Scroll to Details 	Scroll to Details 	Scroll to Details 

Practical tips

1. Don't unless you're into that (no judgement)
 - No learning curve
 - Biggest challenge: get tools to work



Practical tips

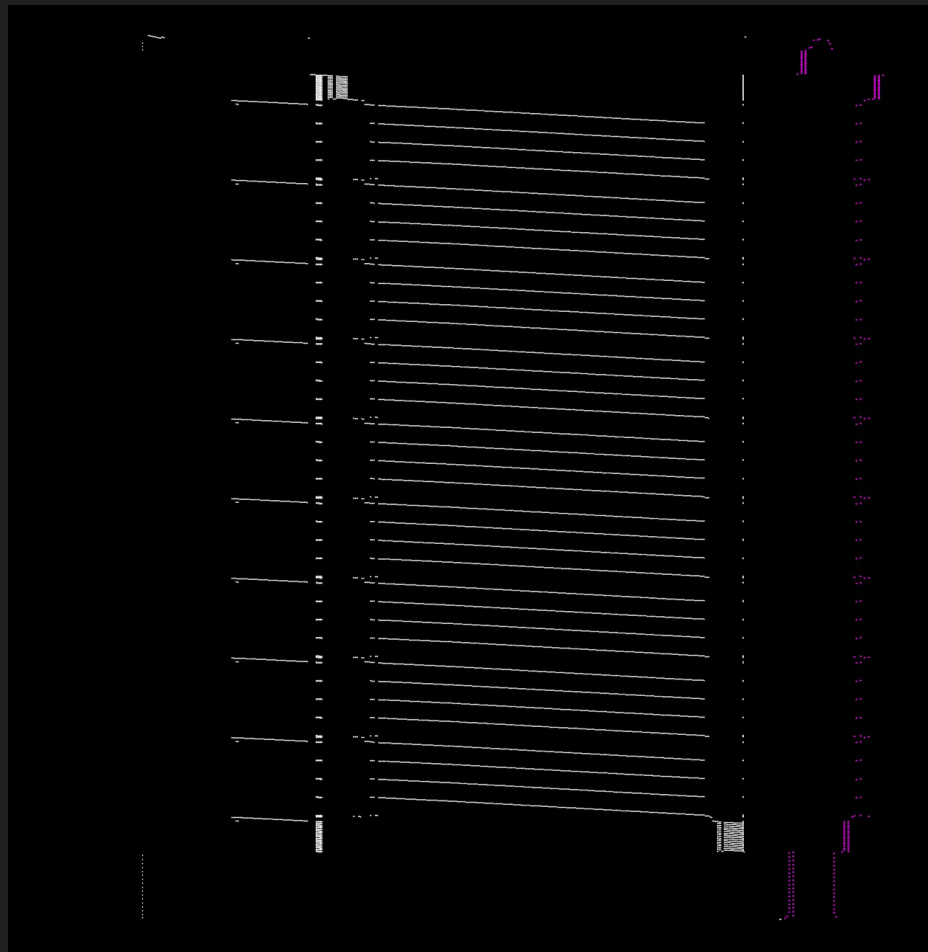
1. Don't unless you're into that (no judgement)
 - No learning curve
 - Biggest challenge: get tools to work
2. ID whitebox
 - Needs lookup tables
 - In our case: lots of movs
 - Looks like a hash function

```
mov    dword [0x00668840], 0xc839e3cf
mov    dword [0x00668844], 0xc938e1cc
mov    dword [0x00668848], 0xff0e8d96
mov    dword [0x0066884c], 0x9f87a97
mov    dword [0x00668850], 0x5dacd26b
mov    dword [0x00668854], 0x36c704d6
mov    dword [0x00668858], 0x9f6e4d36
mov    dword [0x0066885c], 0xaa5b2769
mov    dword [0x00668860], 0x84757b1b
mov    dword [0x00668864], 0xf6079f8d
mov    dword [0x00668868], 0xe415bbbb
mov    dword [0x0066886c], 0x5eafd46e
mov    dword [0x00668870], 0x6c9db038
```

```
🐱 jakob@carbon(14:09) ❤️
(~/.CTF) ~-> ./01_showinstructions.sh
objdump -d whitebox_original | grep -P "^\\s+\\d+:" | cut -c 33-
| cut -d ' ' -f 1 | sed '/^$/d' | sort | uniq -c | sort -nr |
head
5884 movl
434 movb
171 mov
47 add
28 movslq
```

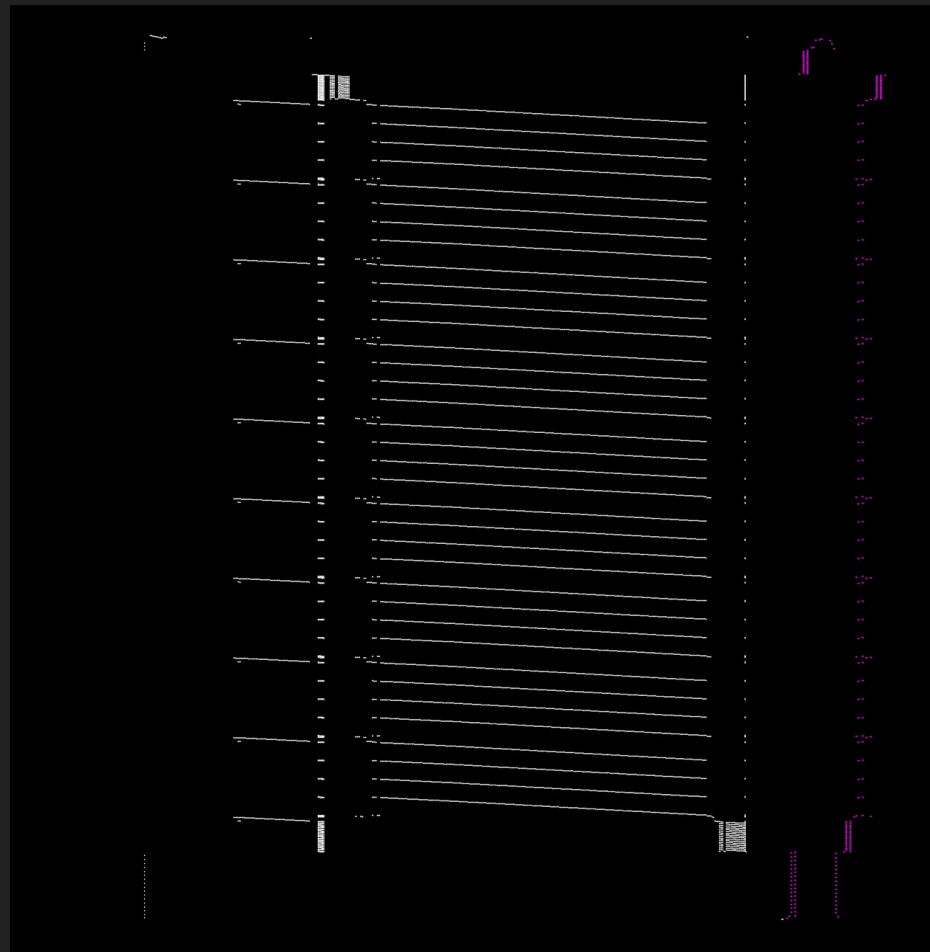
Practical tips

1. Don't unless you're into that (no judgement)
 - No learning curve
 - Biggest challenge: get tools to work
2. ID whitebox
 - Needs lookup tables
 - In our case: lots of movs
 - Looks like a hash function
3. ID algorithm



Practical tips

1. Don't unless you're into that (no judgement)
 - No learning curve
 - Biggest challenge: get tools to work
2. ID whitebox
 - Needs lookup tables
 - In our case: lots of movs
 - Looks like a hash function
3. ID algorithm
 - 10/12/14 rounds? -> AES
 - 16 rounds? -> DES
 - No asymmetric WB :)



Links

- How to whitebox:
 - <https://eprint.iacr.org/2013/104.pdf>
- How to hack:
 - <http://phrack.org/issues/68/8.html#article>
- Tools:
 - <https://github.com/SideChannelMarvels>
- Background:
 - <https://www.slideshare.net/riscure/practical-differential-fault-attack-on-aes>
- Math:
 - <https://www.cryptoexperts.com/ches2015/slides/tutorials/tutorial1.pptx>
- Open Challenges:
 - <https://whibox.cyber-crypt.com/>