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◕ワ◕)ノ*:･ﾟ✧

[DEMO fallback]

(;ᴗ_ᴗ)

```
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 a1l_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)

See https://eprint.iacr.org/2013/104.pdf

AES

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

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

See https://eprint.iacr.org/2013/104.pdf

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

See https://eprint.iacr.org/2013/104.pdf

AES

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

- ShiftRows
- TBoxesTyiTables
- XORTables

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

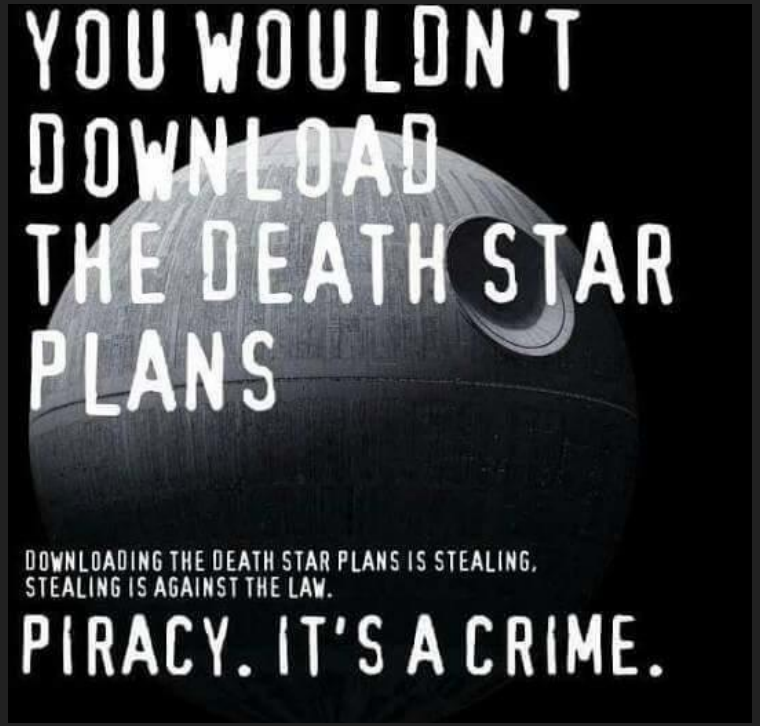
Combine into Lookup table, hardcode

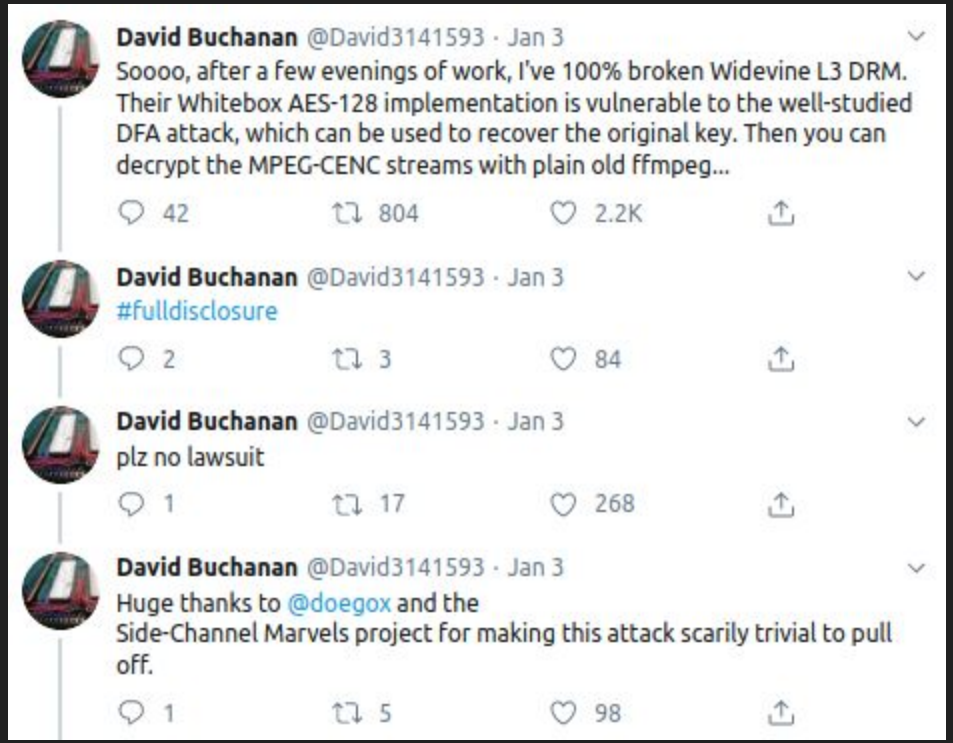
See https://eprint.iacr.org/2013/104.pdf

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

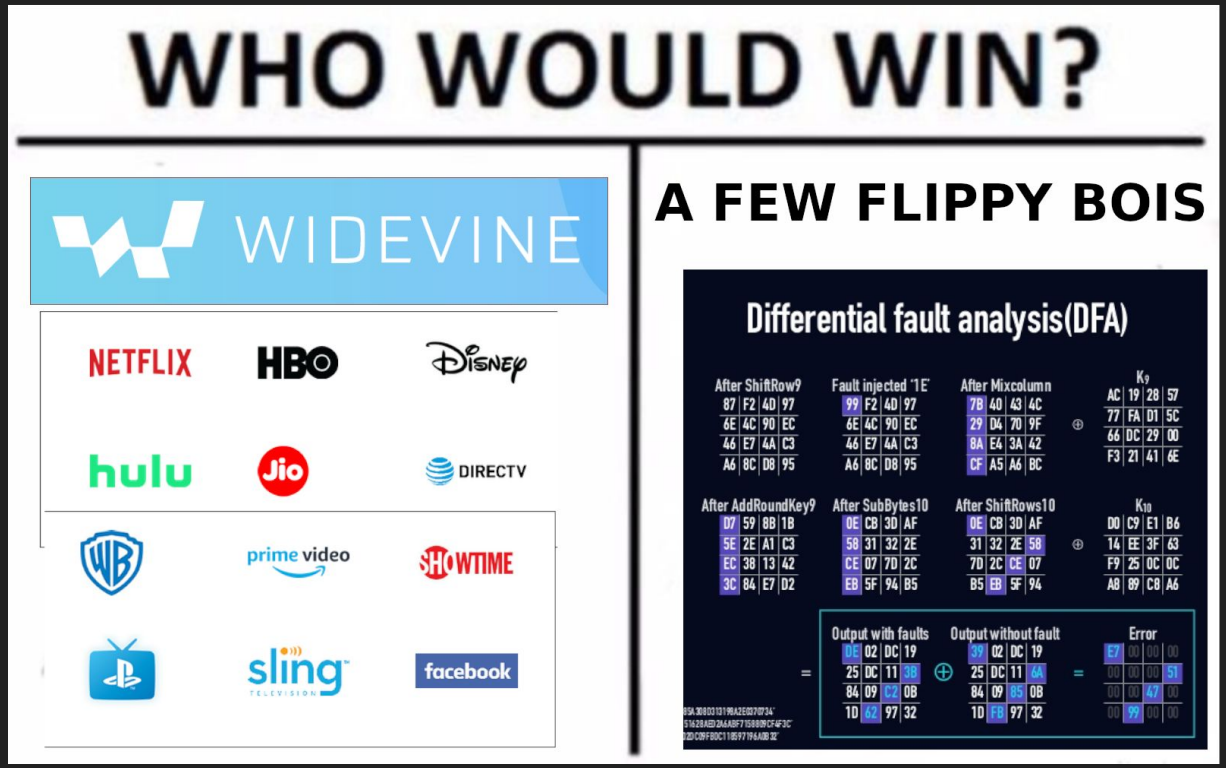
DRM & “Raubmordkopiererei”

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

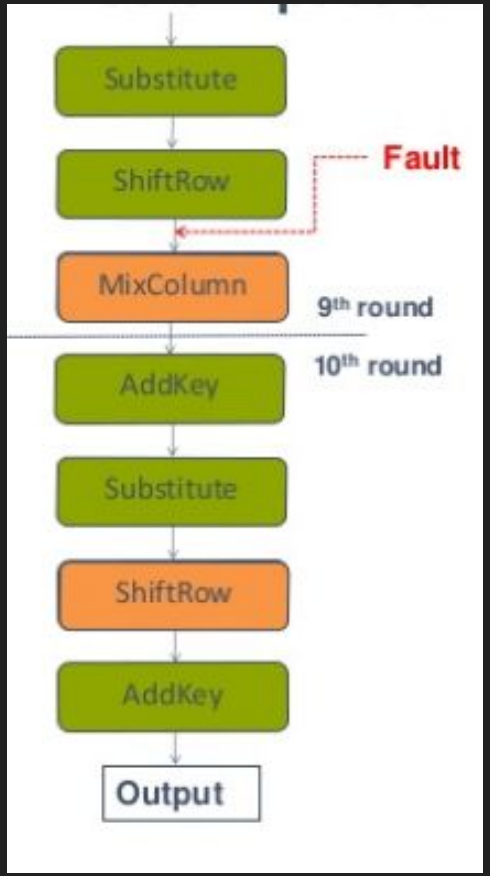




See https://twitter.com/David3141593/status/1080616099174141952

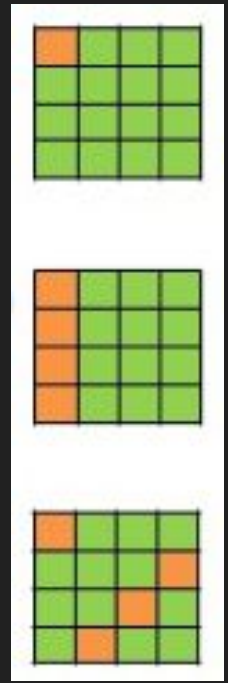


See https://twitter.com/David3141593/status/1080616099174141952

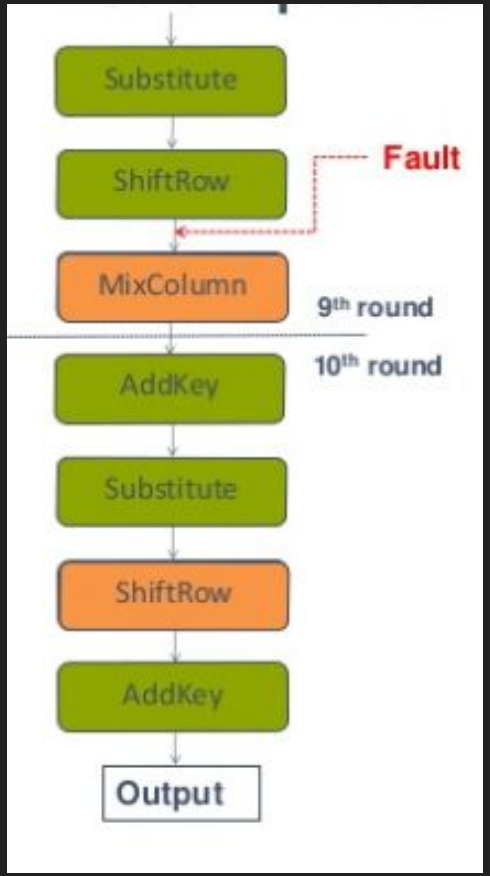


Differential Fault Analysis

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



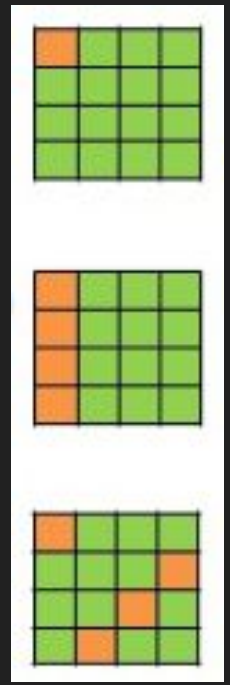
See https://www.slideshare.net/riscure/practical-differential-fault-attack-on-aes



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!

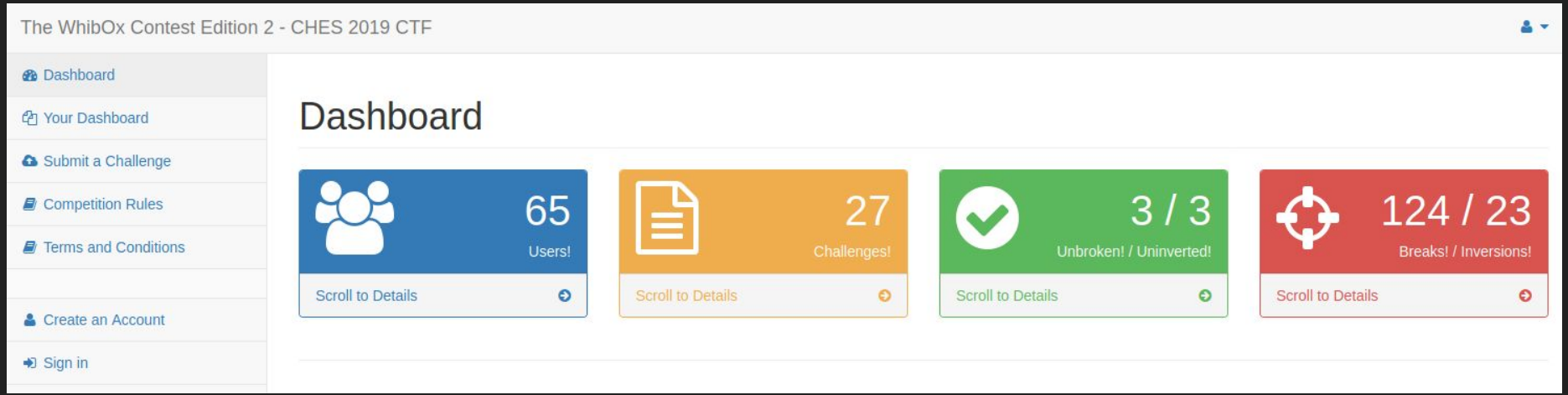
- `ISB(x1+K1)+ISB(x1+F1+K1)=`
- `2[ISB(x2+K2)+ISB(x2+F2+K2)]`
- `ISB(x2+K2)+ISB(x2+F2+K2)=`
- `ISB(x3+K3)+ISB(x3+F3+K3)`
- `ISB(x4+K4)+ISB(x4+F4+K4)=`
- `3[ISB(x2+K2)+ISB(x2+F2+K2)]`



See https://www.cryptoexperts.com/ches2015/slides/tutorials/tutorial1.pptx

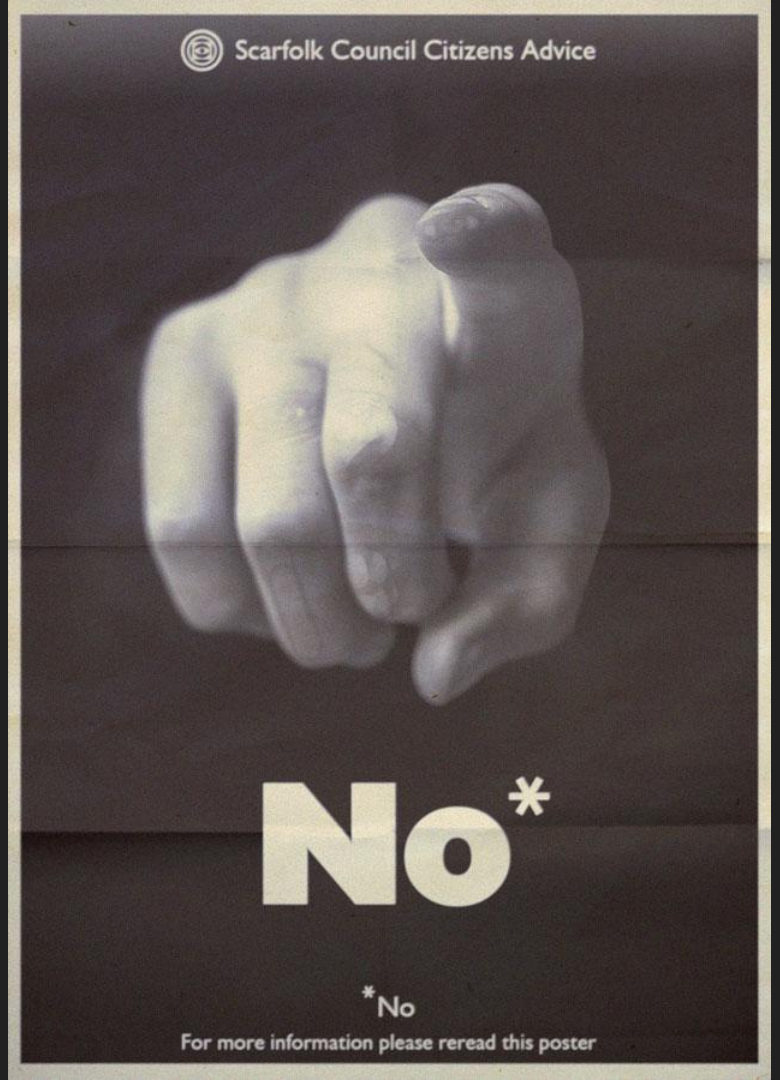
Differential Fault Analysis

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



Practical tips

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



Practical tips

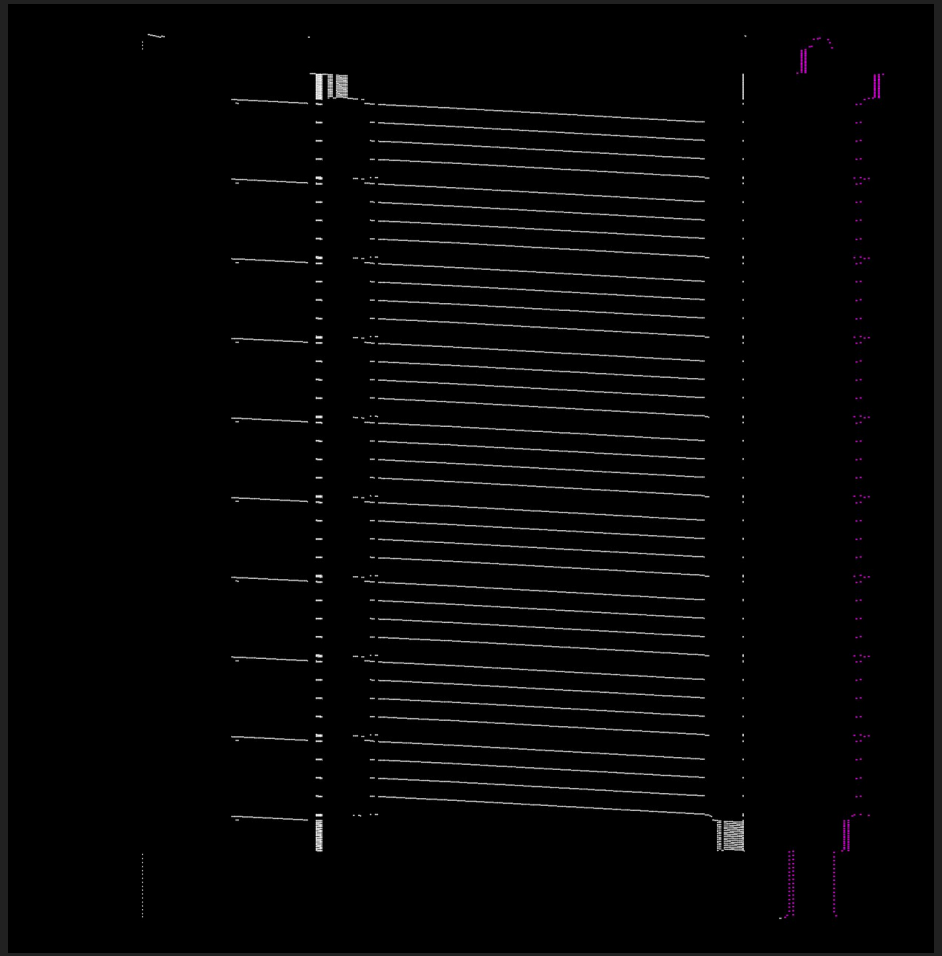
1. Don’t unless you’re into that (no judgement)
   - No learning curve
   - Biggest challenge: get tools to work
2. ID whitebox
   - Needs lookup tables
     - In our case: lots of movs
   - Looks like a hash function

```
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
```

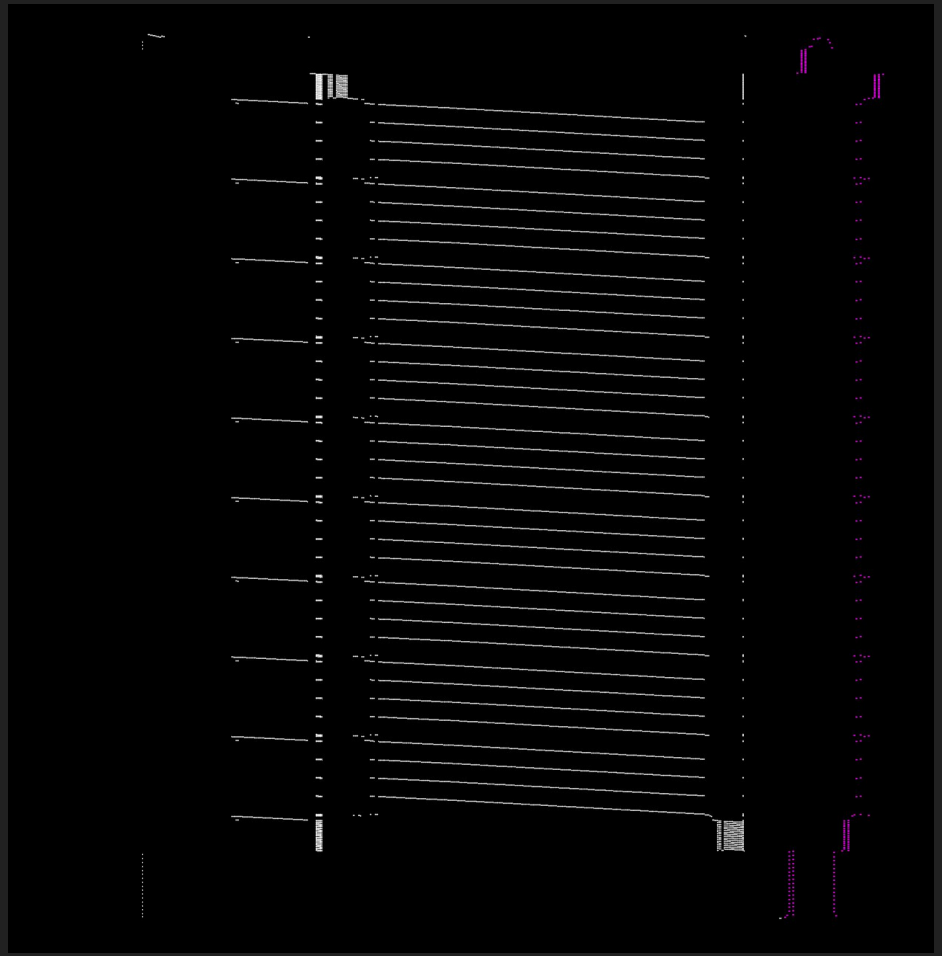
Practical tips

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/