Finding the key in the haystack

A practical guide to Differential Power Analysis

hunz
Zn000h AT gmail.com

December 30, 2009

What's DPA?

- side channel attack
- introduced by Paul Kocher et al. 1998
- recover secret keys used for en/decryption
  algorithm needs to be known

- current consumption depends on data being processed
  $\Rightarrow$ current measurements give hints about internal data being processed

- key can't be found directly in the power consumption
  $\Rightarrow$ some sort of extraction/recovery method necessary
  $\Rightarrow$ DPA does this

DPA - The basic idea

- in this talk we'll attack a particular AES-128 encrypt implementation
- bruteforce: 2^{128} - one needs to get all bits right at the same time
- using DPA we'll know once we got a single byte of the key right
- we can recover the key byte-for-byte
  $\Rightarrow (2^8) * 16$ key guesses instead of 2^{128}
  $\Rightarrow$ 4k keys to try!
- we need to encrypt several (10^2 up to 10^6) plaintexts and measure power consumption

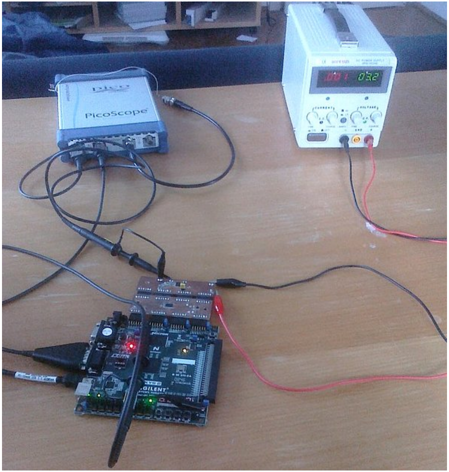
a few more notes...

- not a flaw of the AES crypto-algorithm!
- nearly every crypto-algorithm affected
- unless specific countermeasures realized in implementation

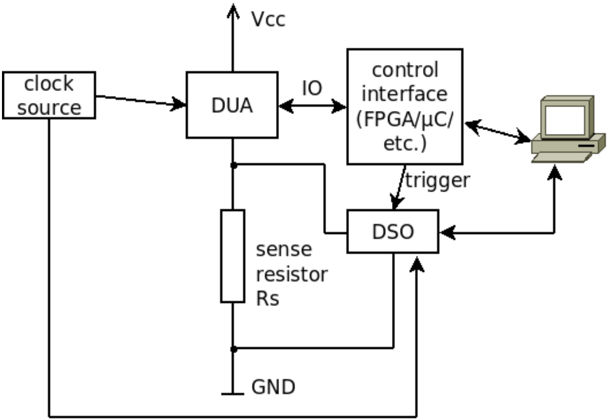
- no countermeasures in standard consumer hardware they're expensive
- because they're patented!

- that means: most consumer hardware vulnerable to PA attacks
- PA still not widespread in the hardware hacker community?

Measurement setup



Measurement setup

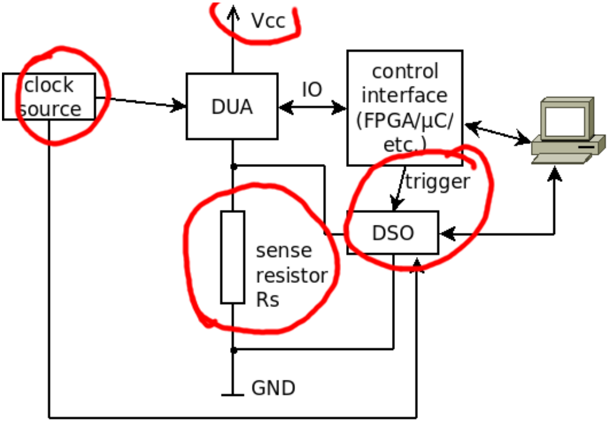


DUA: Device Under Attack
DSO: Digital Storage Oscilloscope

Measurement procedure

1. DUA: configure (prepare for en/decryption)
2. scope: setup trigger
3. DUA: start en/decryption
4. scope: read data
5. goto 1

Measurement setup - a closer look



Sensing resistor

- $I = \frac{\Delta U}{R}$
- measure voltage drop ΔU, R known
- 1..10 Ω usually fine

- smaller values are usually better $\rightarrow$ less drop
  but higher measurement precision necessary
  increase supply voltage if drop to high

- use resistors with low inductance
- GND or Vcc sides are both fine

Digital Storage Oscilloscope

- samplerate ≥ 250 MS/s
  doesn't depend directly on DUA-clock or max. DUA-clock
  but: "max clock" of "interesting part" of the IC
  but: internal capacitance of IC blocks hi frequency
- sample buffer should be rather Mpts than kpts
- example: 4kpts @250MS/s with DUA @4MHz:
  $\frac{4096}{250M/4M} = 66$ cycles
- splicing traces possible with precise triggering

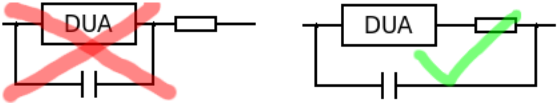
Digital Storage Oscilloscope (2)

- Picoscope 5203 1.8k€ : 32Mpts, 250MHz, 1GS/s, ±100mV
  with 10x probe (250MHz): ± 1000mV
  that's 2V / 256 ≈ 8mV precision
- try to use full range of ADC by adjusting
  - sensing resistor - larger R → larger ΔU
  - supply voltage - but be careful
- if you're lucky enough to have a differential probe use it
- we're trying to build our own low-cost diff-probe
  we'd totally appreciate your help!

Voltage source

- disconnect on-board supply, use your own
  in case of multiple supplies: smallest is usually the right one
- lab power supplies often got more ripple than one would think!
- no step-down, short thick cables, capacitors close to target
- rechargeable batteries + low noise linear regulators
  example: LP3878-ADJ
  use fixed adjustment resistors though!
- slightly higher supply voltage often won't hurt
  also, there's the drop across the sensing resistor

Capacitors



- remove on-board capacitors
  reduce clock if necessary for stability
- add ceramic fast-response capacitors with different capacities
- parallel to device AND reisitor
- seperate PCB if possible

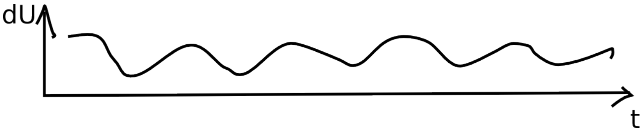
Clock

- sinus clock signal avoids ringing (series resistor)
- use external clock source and sync with scope if possible
  otherwise there's jitter and drifting
  workaround: stretch cycles to fixed raster using software
  (align edges of current-peaks)

- higher clock $\longrightarrow$ better use of sample buffer
- slower clock $\longrightarrow$ more stability but wasting sample buffer

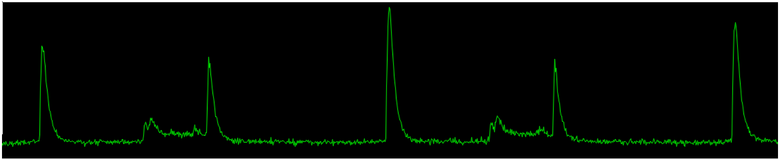
Examples

bad:



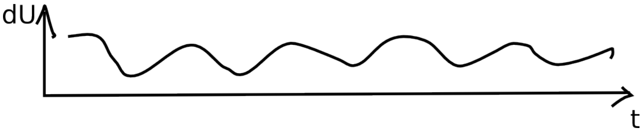
(slower clock might help here)

good:



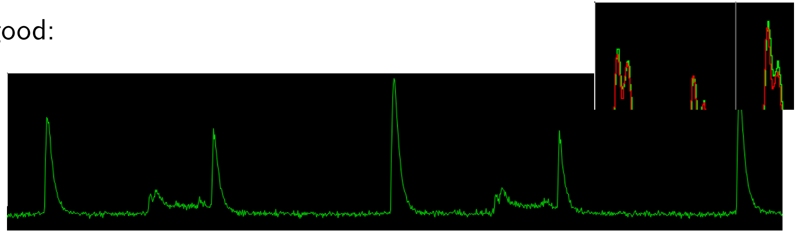
Examples

bad:



(slower clock might help here)

good:

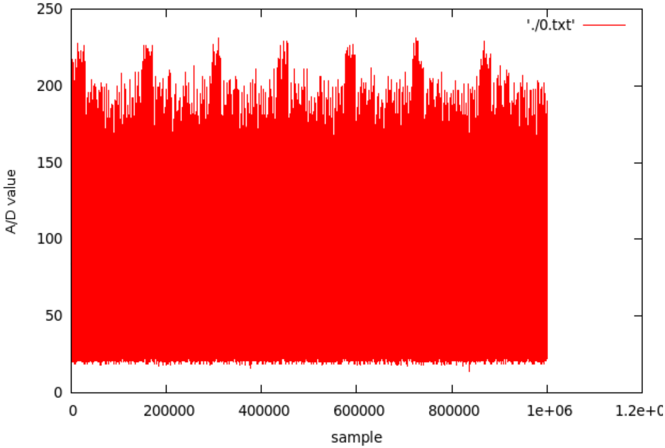


Trigger & alignment

- proper alignment/syncronization of the power traces is crucial
- every instruction needs to be at constant sample offset

- precise triggering based on IO of the DUA
- pattern-matching to align the traces after recording (majority of dynamic current is not data- but instruction-dependent)
- least squares is a simple method → Wikipedia: Sum of squares (sum of squared differences between two traces)

look for 10 AES rounds (7 shown here)



How does DPA work?

- constant values (key) can't be recovered directly without profiling the device → template based power analysis
- DPA: recover unknown const data (key) by analyzing its influence on known, variable data (plain- or ciphertext)

- Original method introduced by Kocher: Difference of means
- here: Analysis using Pearson Correlation
  will spare you the formula here but wikipedia is your friend:
  Wikipedia: Pearson product-moment correlation coefficient

Analysis: short version

1. guess part of the key
2. use it to evaluate the en/decryption function to get suitable intermediate values
3. use power consumption measurements to verify the correctness of the intermediate values
4. if correct done else goto 1

- 1st question: What's a suitable intermediate value?

Intermediate values

look for intermediate values during en/decryption that

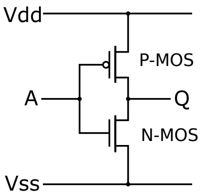
- depend on both key and plaintext
- depend only on small portions of the key
  (exhaustive search necessary on these portions)
- exhibit strong variation even for little input variation
  (S-Boxes!)
  otherwise wrong key guesses with few wrong bits seem to be correct as well

Example: AES Encryption

- round 1:
  - AddRoundKey: ival[0..15] := key[0..15] $\oplus$ plain[0..15]
    - depends on key and plain $\surd$
    - depends on small portion of key $\surd$ (8 bit)
    - but: no strong variation for little input variation :-((1 bit)
  - SubBytes: ival'[0..15] := SubByte(ival[0..15])
    - strong variations: $\surd$:-) (due to sbox-properties)
- done.

- Next Question: How to verify ival' using the power consumption?

Estimating data-dependent current consumption



C-MOS Inverter
(source: wikipedia)

- usually Complementary (N- & P-) MOS logic
- capacity at Q
- switching causes (dis)charge- and short-circuit current
  $\Rightarrow$ current increases with $0 \leftrightarrow 1$ changes
- only approximations possible

Hamming Distance model

- HD(a, b) := number of bits changed from a to b
  example: HD(b101, b011) = 2
- fine for registers & hardware crypto units
- problem: previous value needs to be known
  $\longrightarrow$ not always the case, implementation specific

Example: Hamming Distance model

8-Bit AVR AES implementation:
(source: Rijndaelfurious from
http://point-at-infinity.org/avraes/)

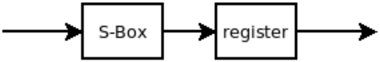
```
encrypt:
        ldi I, 10
encryp1:rcall addroundkey          ; AddRoundKey
        ldi ZH, high(sbox<<1)      ; S-Box LUT addr. hi byte
        mov ZL, ST11               ; S-Box LUT addr. lo byte
        lpm ST11, Z                ; ST11 := sbox[ST11]
```

- AddRoundKey: ival[0..15] := key[0..15] ⊕ plain[0..15]
- SubBytes: ival'[0..15] := SubByte(ival[0..15])

- ⇒ HD(ival[i], sbox(ival[i]))

2nd example: Hamming Distance model

hardware AES implementation:



- dedicated register at output of S-Box
- holds last S-Box output

- $\Rightarrow$ HD(sbox(ival[i]), sbox(ival[j]))
  for some i, j with i $\neq$ j

- but: you have to guess 2 bytes of the key at a time

Hamming Weight model

- HW(a) := number of '1'-bits
  example: HW(b101) = 2
- often helps if previous value of register isn't known
- works as long as previous value is constant
- fine for software crypto implementations
  (data busses being charged to '1')

Putting the pieces together

estimated current for keybyte[0] guess x00 (values in hex):

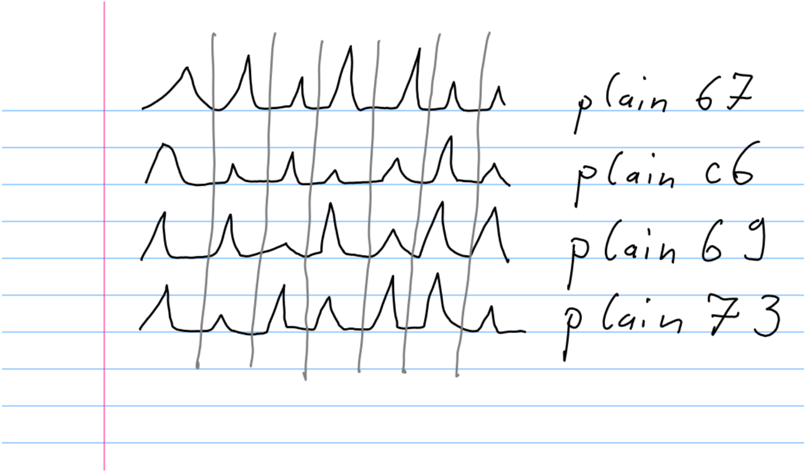
```
plaintext[0]=67 -> ival[0]=67
             -> sbox(ival[0])=85 -> HD(23,42) = 4

plaintext[0]=c6 -> ival[0]=c6
             -> sbox(ival[0])=b4 -> HD(23,42) = 4

plaintext[0]=69 -> ival[0]=69
             -> sbox(ival[0])=f9 -> HD(23,42) = 2

plaintext[0]=73 -> ival[0]=73
             -> sbox(ival[0])=8f -> HD(23,42) = 6
```

Putting the pieces together

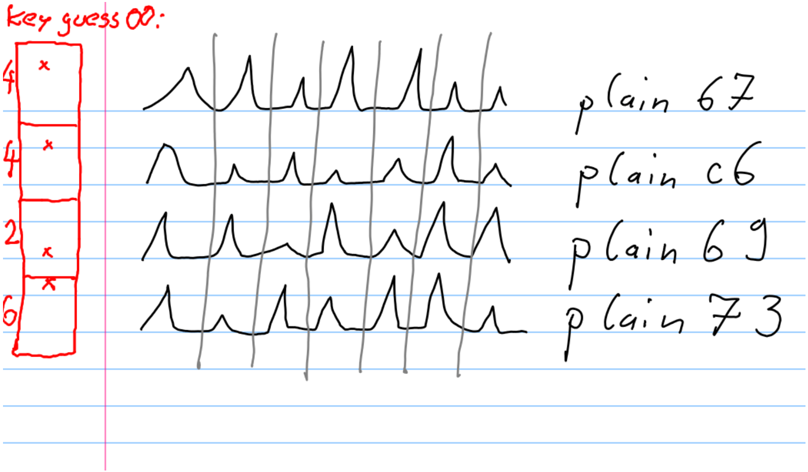
estimated current for keybyte[0] guess x00 (values in hex):

```
plaintext[0]=67 -> ival[0]=67
            -> sbox(ival[0])=85 -> HD(23,42) = 4

plaintext[0]=c6 -> ival[0]=c6
            -> sbox(ival[0])=b4 -> HD(23,42) = 4

plaintext[0]=69 -> ival[0]=69
            -> sbox(ival[0])=f9 -> HD(23,42) = 2

plaintext[0]=73 -> ival[0]=73
            -> sbox(ival[0])=8f -> HD(23,42) = 6
```

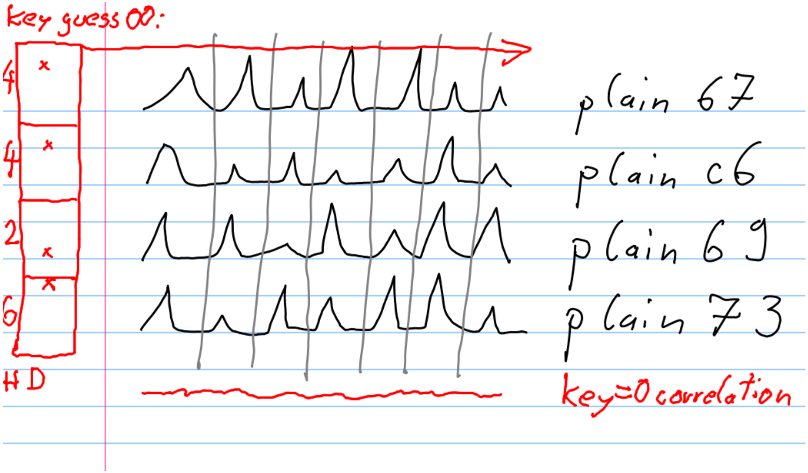
correlation key[0] guess x00



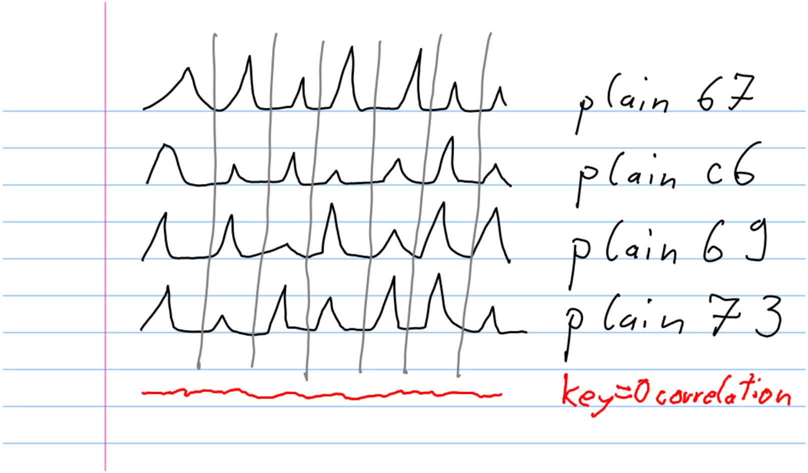
correlation key[0] guess x00



correlation key[0] guess x00



correlation key[0] guess x00



let's try key[0]=1

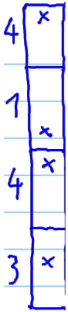
estimated current for keybyte[0] guess x01 (values in hex):

```
plaintext[0]=67 -> ival[0]=66
            -> sbox(ival[0])=e8 -> HD(23,42) = 4

plaintext[0]=c6 -> ival[0]=c7
            -> sbox(ival[0])=2c -> HD(23,42) = 1

plaintext[0]=69 -> ival[0]=68
            -> sbox(ival[0])=59 -> HD(23,42) = 4

plaintext[0]=73 -> ival[0]=72
            -> sbox(ival[0])=cb -> HD(23,42) = 3
```

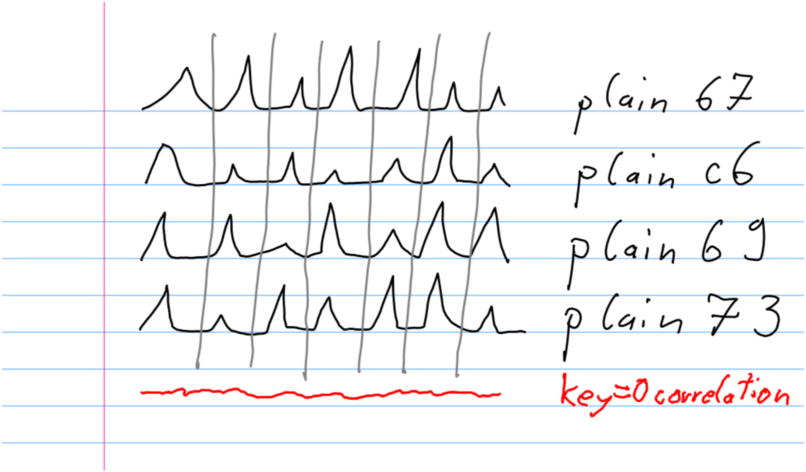
let's try key[0]=1

estimated current for keybyte[0] guess x01 (values in hex):

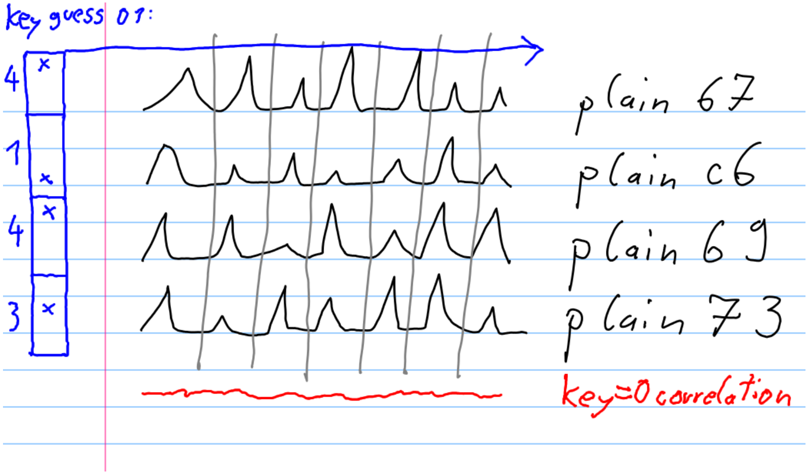
```
plaintext[0]=67 -> ival[0]=66
            -> sbox(ival[0])=e8 -> HD(23,42) = 4

plaintext[0]=c6 -> ival[0]=c7
            -> sbox(ival[0])=2c -> HD(23,42) = 1

plaintext[0]=69 -> ival[0]=68
            -> sbox(ival[0])=59 -> HD(23,42) = 4

plaintext[0]=73 -> ival[0]=72
            -> sbox(ival[0])=cb -> HD(23,42) = 3
```



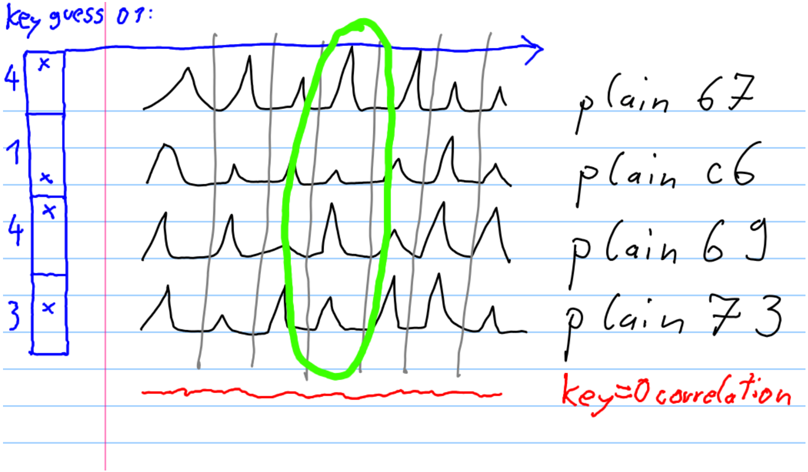
correlation key[0] guess x01



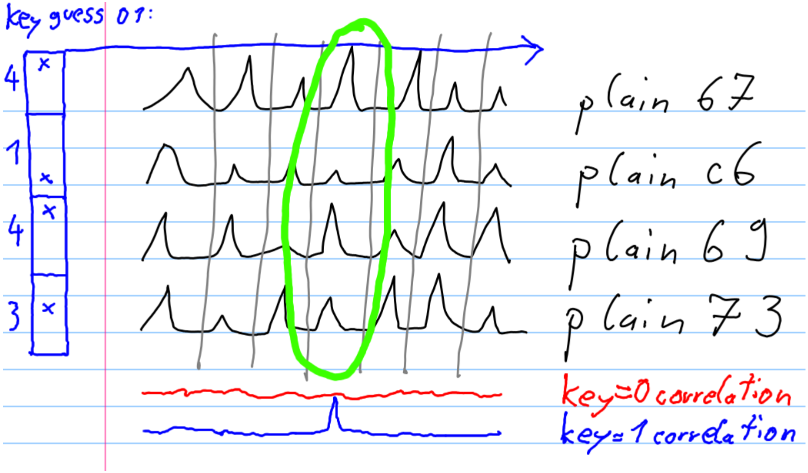
correlation key[0] guess x01



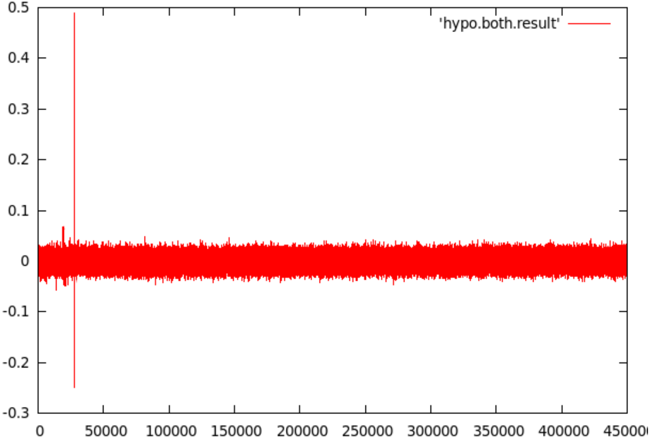
correlation key[0] guess x01



correlation key[0] guess x01

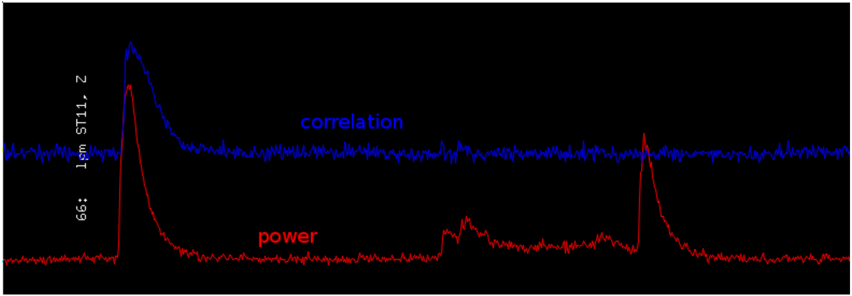


HD correlation example (correct key)



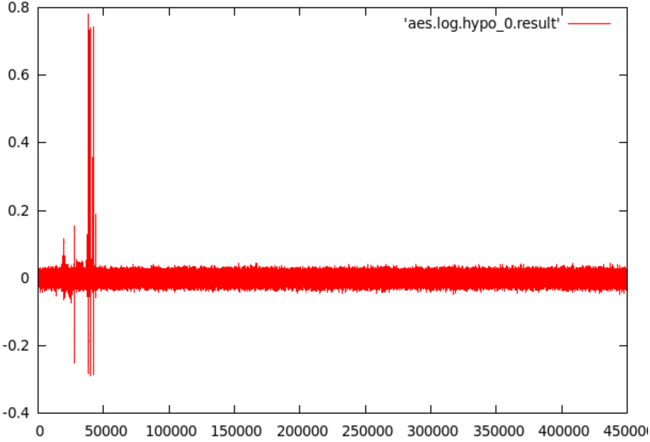
(x: sample, y: correlation)

HD correlation example zoomed



(x: sample, y: correlation/power)

HW correlation example (correct key)



(x: sample, y: correlation)

Short note on decryption

- cipher- instead of plaintext
- inverse round-order

- actual AES-key is roundkey of last round now
- can't be recovered directly
- requires knowledge of prior roundkeys

- recovery of each roundkey necessary

conclusion

- nearly all crypto implementations in consumer products vulnerable to PA attacks
- can be done at home, analysis is no rocket science

- adequate DSOs are expensive but should be affordable for hackerspaces
- be patient, play with the measurement setup
- write down your attempts and observations

- attack your own device before doing blackboxes

References

- Power Analysis Attacks: Revealing the Secrets of Smart Cards
  ISBN 0-387-30857-1
- M. Aigner, E. Oswald: Power Analysis Tutorial
- P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis"
- RijndaelFurious AVR AES Implementation
- Wikipedia: Advanced Encryption Standard
- Wikipedia: Sum of squares
- Wikipedia: Pearson product-moment correlation coefficient
- Sample code (Google code project)
- "DPA talk @26c3" at Google Wave