Hardware Security
Week 4 -
Power Attacks

26T1
Hammond Pearce

Slide material acknowledgements to
Ramesh Karri, Benjamin Tan,
JV Rajendran, Jason Blocklove
Christian Pilato, Luca Collini

RECAP

1. What does DFT stand for?
2. Why is DFT typically used?
3. What are the security consequences of using DFT?
4. How do Mirror Key Registers address this challenge?
5. How can NOT gates be used to address this challenge?

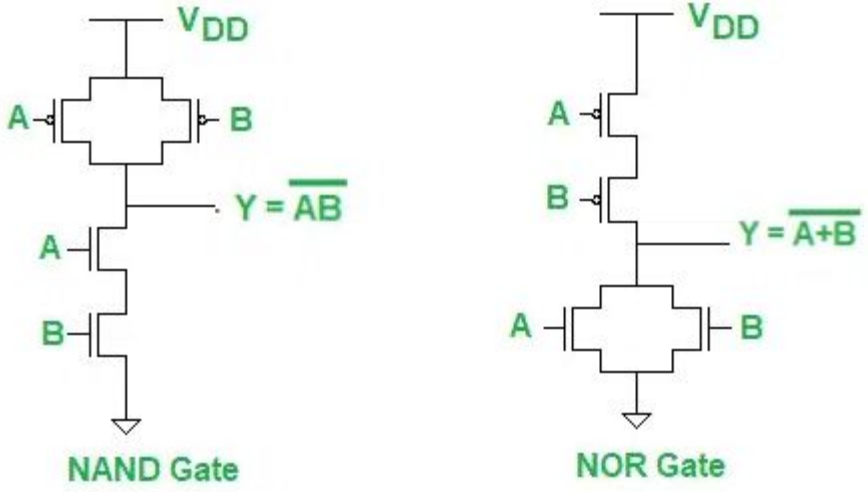
Let’s talk about Hardware

When we build an integrated circuit,

how does it work?

Logic gates are component abstractions

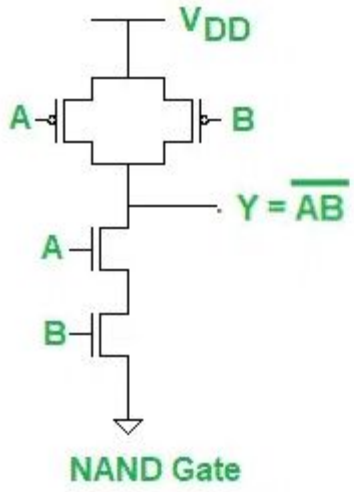
- Built mainly with FETs

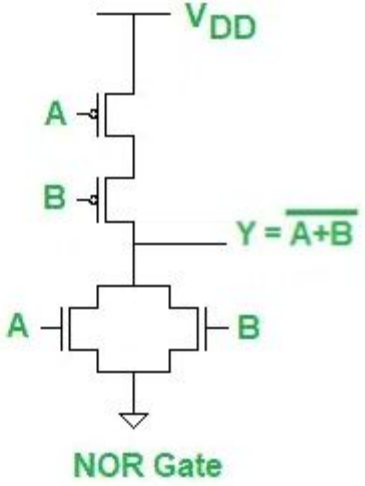


Logic gates are component abstractions

- Built mainly with FETs

We like to treat these as “ideal”... but are they?





Real circuits aren’t ideal

- Real execution takes time: propagation delays

And

- Real execution consumes power

Real circuits aren’t ideal

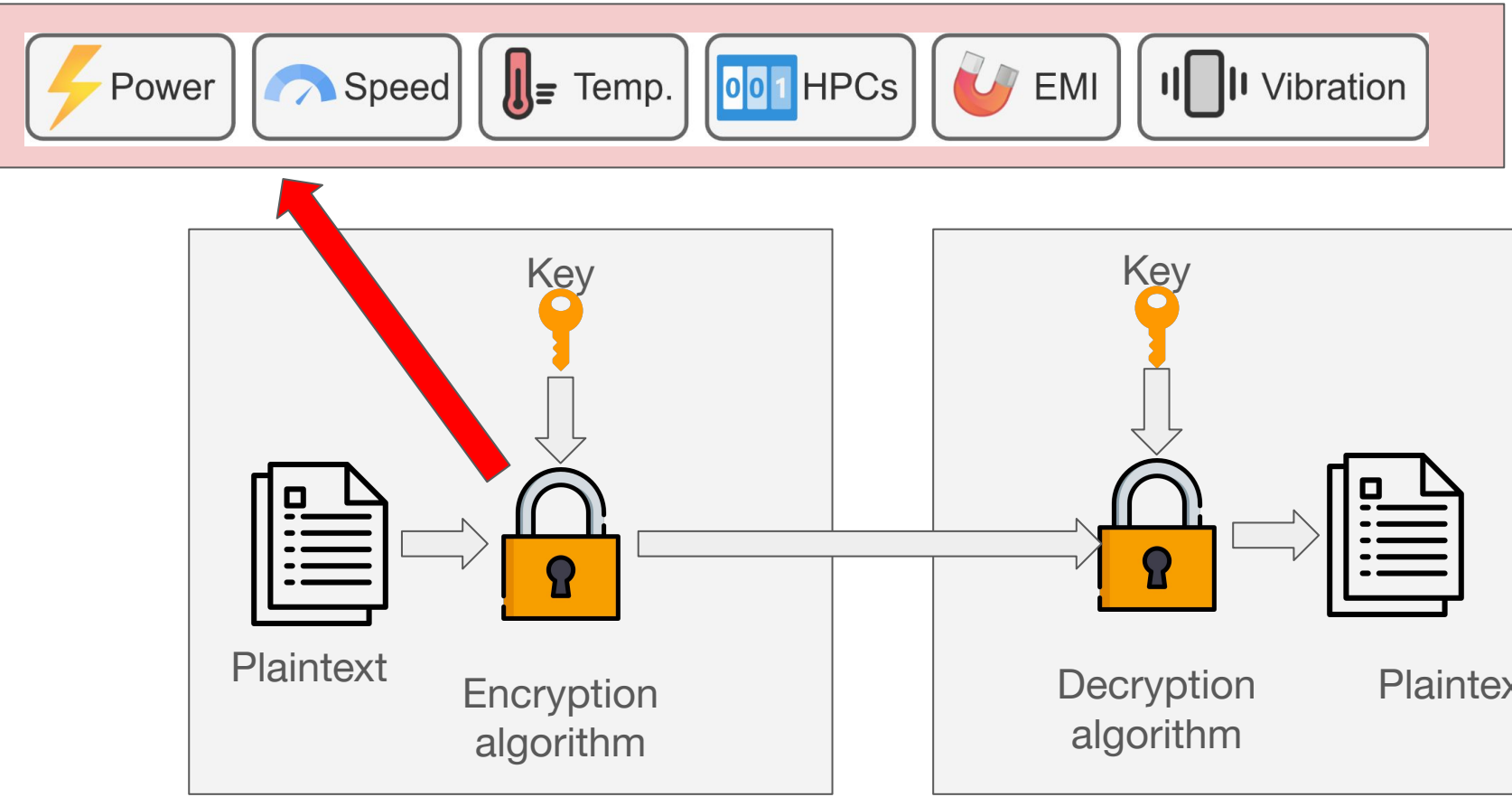
- Real execution takes time: propagation delays

And

- Real execution consumes power

What does this mean?

Side Channels



Side channel threat model

- Algorithms like AES are theoretically secure…
  - But we've already seen that real world implementations might be vulnerable!
- "Side channels" are *observable side effects* of an implementation
- They include:
  - Power consumption
  - Time durations
  - Electromagnetic emissions

- Can we use the side channels to tell us about the data??

A classic “software” side channel: timing

```python
def check_password(input_str):
    correct_password = "mysecret"

    for i in range(len(correct_password)):
        # If input_str is shorter, we'll hit an index error unless we check
        if i >= len(input_str) or input_str[i] != correct_password[i]:
            return False  # Return immediately on mismatch

    # If the user-supplied password is longer (or shorter) than correct_password,
    # it's not valid
    if len(input_str) != len(correct_password):
        return False

    # If we get here, every character matched and lengths are equal
    return True
```

A classic “software” side channel: timing

```
def check_password(input_str):
    correct_password = "mysecret"

    for i in range(len(correct_password)):
        # If input_str is shorter, we'll hit an index error unless we check
        if i >= len(input_str) or input_str[i] != correct_password[i]:
            return False  # Return immediately on mismatch

    # If the user-supplied password is longer (or shorter) than correct_password,
    # it's not valid
    if len(input_str)
        return False

    # If we get here,
    return True
```

If attacker can measure the time check_password takes,

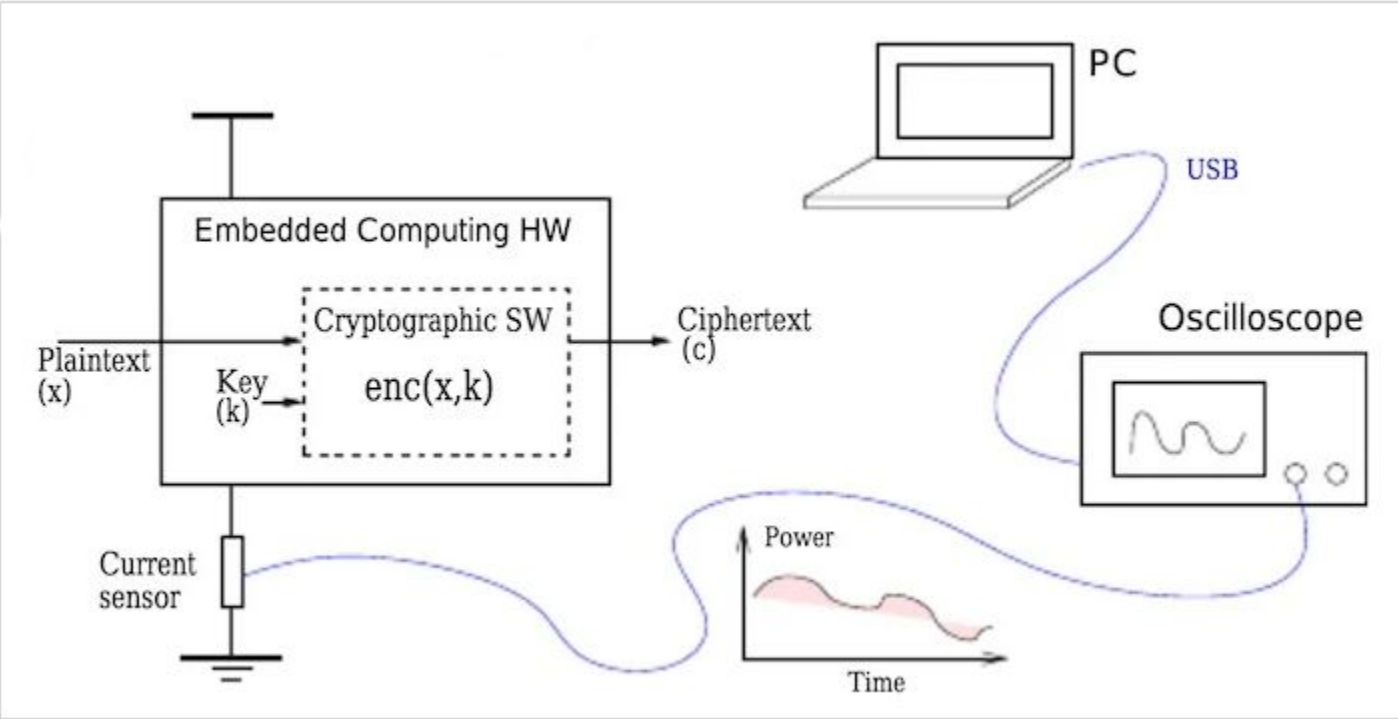
they can observe a different execution time depending on how many characters they guessed correctly!

Symmetric key algorithms: constant time

- Unfortunately, a simple timing analysis attack won't work on AES or DES (why?)
- We need to look at a different side channel…
- What happens if we **record power consumption**?

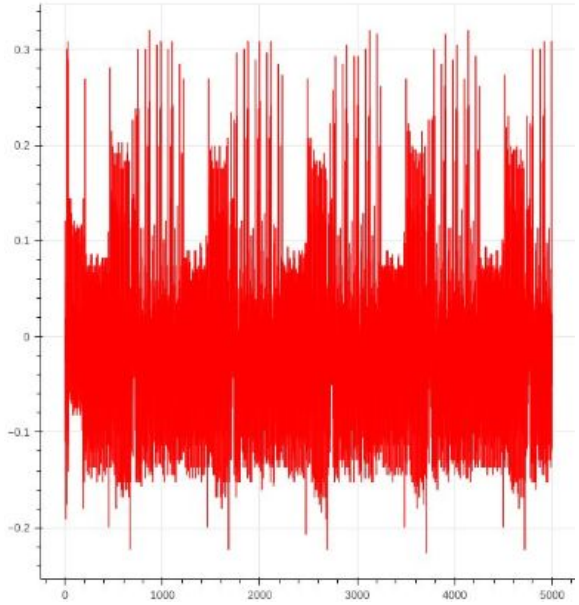
Part 1: Power Analysis Attack

Recording power consumption



Power side channels of circuits

- ICs have two types of power consumption:
- Static power (always there)
- Dynamic (Switching) power
  - Power changes depending on what the circuit is doing!!
  - I.e., Power = F(computation)!

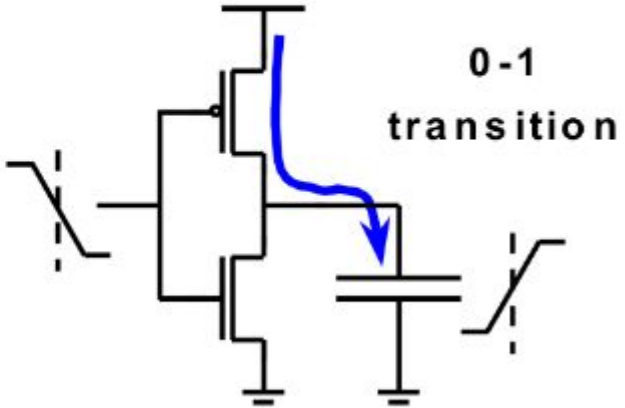


Power consumption reflects data - why?

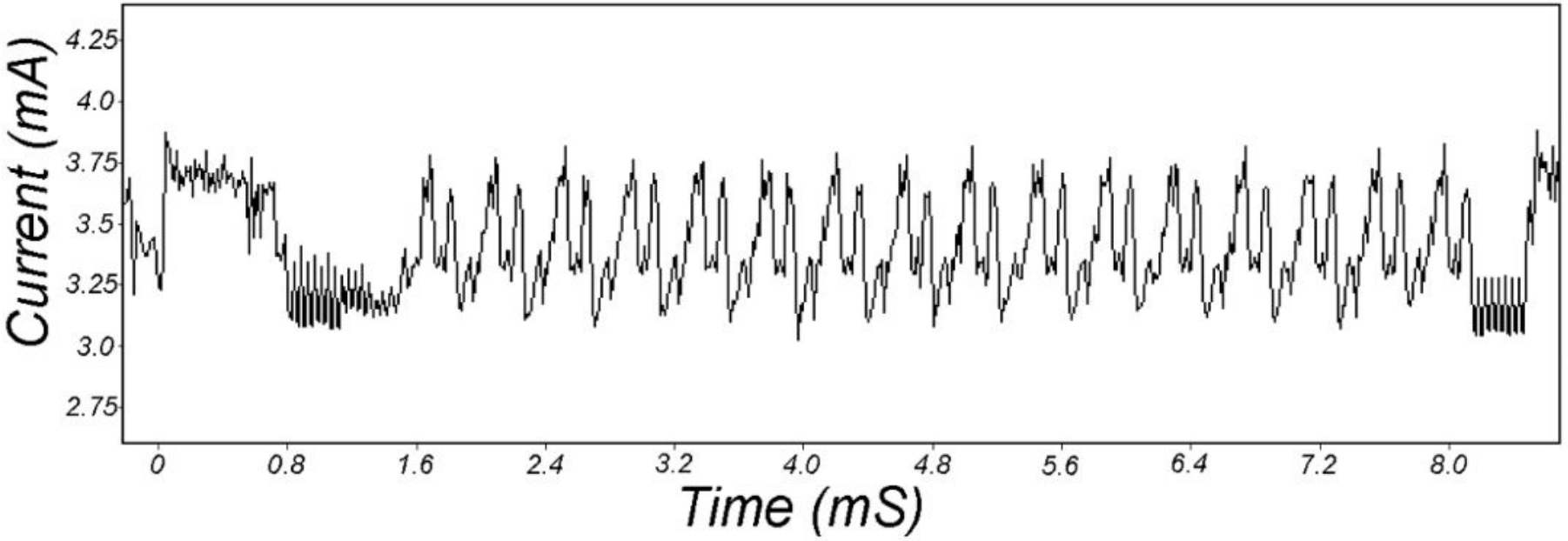
- Data transitions (switching) consume more power than idling:
  - CMOS inverter:

a=>a, b=>b Capacity

Input	Output	Current
0 → 0	1 → 1	Low
0 → 1	1 → 0	Discharge
1 → 0	0 → 1	Charge
1 → 1	0 → 0	Low

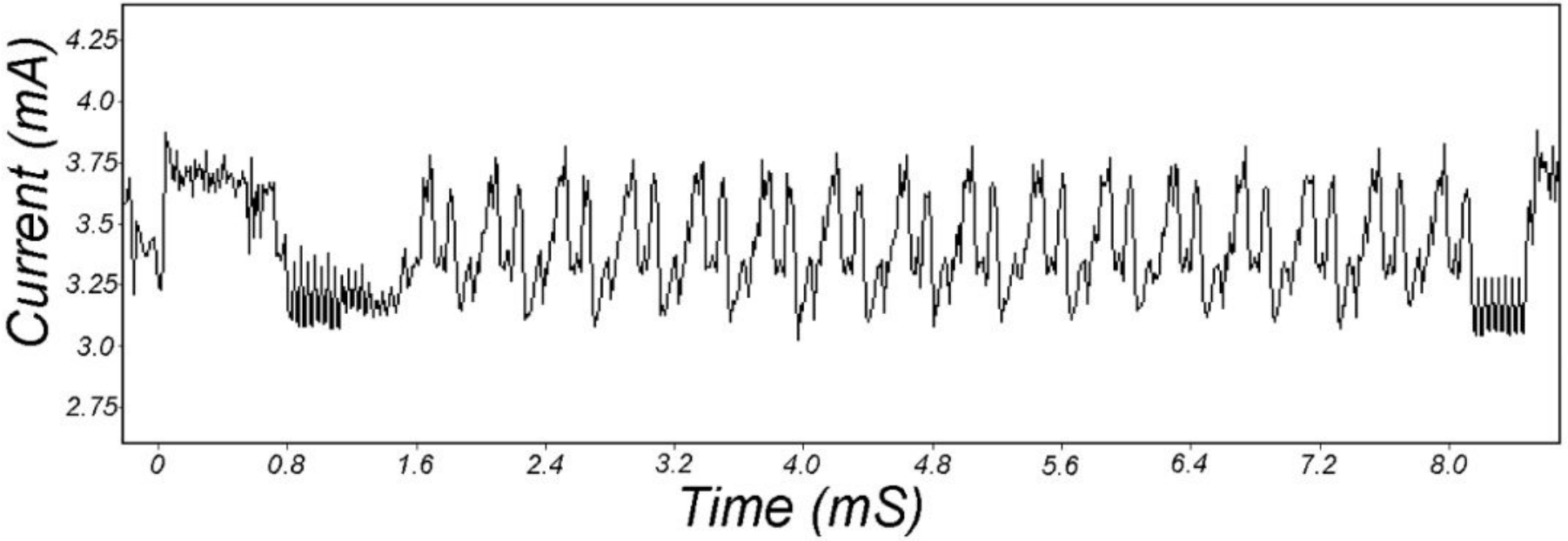


Consumption for DES



Paul Kocher, Joshua Jaffe and Benjamin Jun. "Differential Power Analysis". Crypto'99

Consumption for DES



Probably not dedicated hardware… why?

Paul Kocher, Joshua Jaffe and Benjamin Jun. "Differential Power Analysis". Crypto'99

The “simplest” power attack: SPA

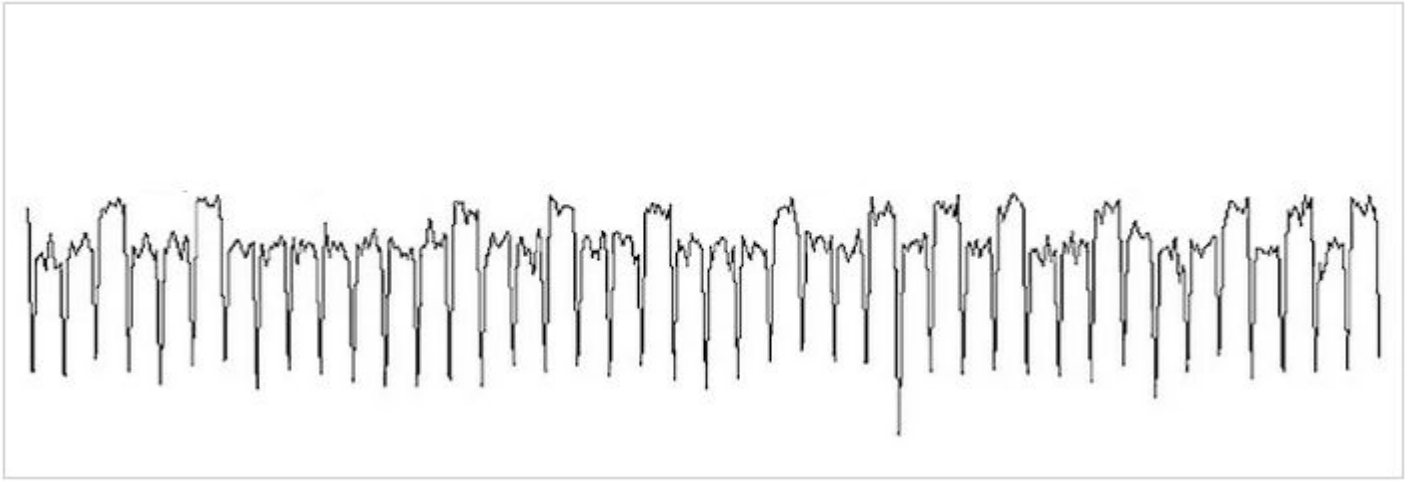
- Simple Power Analysis
- Interpret power consumption directly and visually

Could this work??

The “simplest” power attack: SPA

- Simple Power Analysis
- Interpret power consumption directly and visually

Could this work?? Here is power consumption of an RSA algo:



SPA on RSA

Algorithm 1 Multiply and Square Algorithm

```
1: procedure Mul − Squ(g,K)
2:     Convert K into binary representation k_0, k_1, ...k_n, where k_0 = 1
3:     if K == 0 then
4:         Result = 1
5:         return Result
6:     else
7:         Result = g
8:         for doi ← 1, n
9:             if k_i == 1 then
10:                Result = M(Result, Result)
11:                Result = M(Result, g)
12:            else
13:                Result = M(Result, Result)
14:            end if
15:        end for
16:        return Result
17:    end if
18: end procedure
```

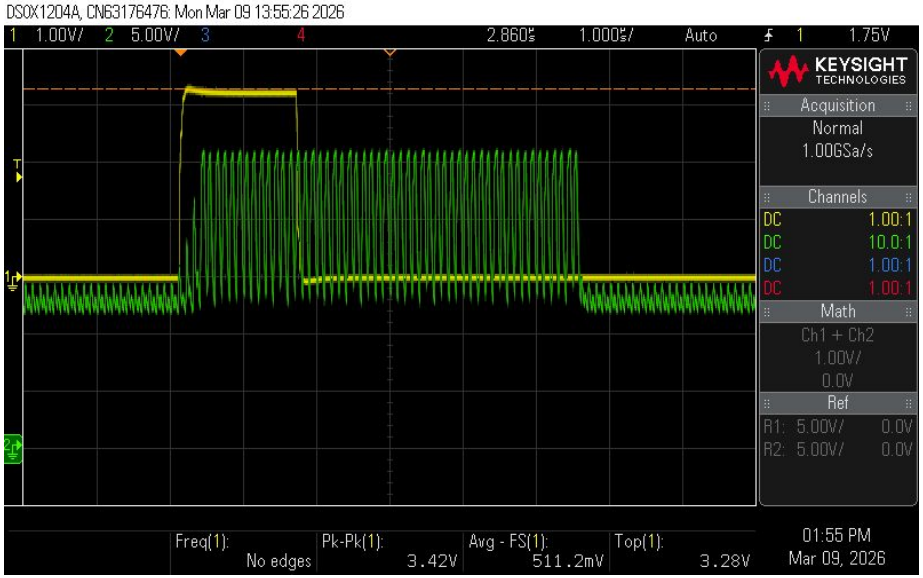
Multiplication operations consume more power than squaring operations, which results in a taller spike in the power trace

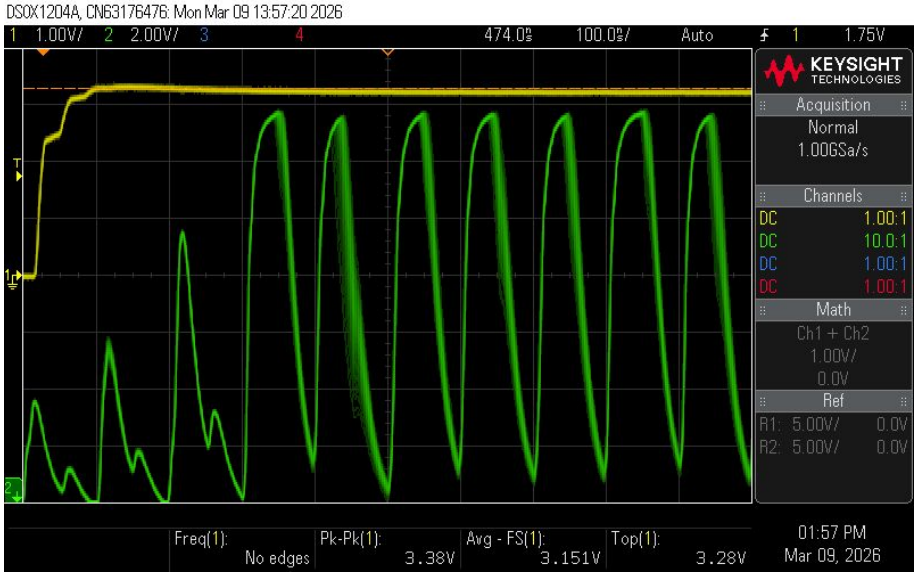
https://scl.engr.uconn.edu/courses/ece4451/Lec8aPower_Side_Channel_Analysis.pdf

Issues with SPA:

- Only works on simple electronic systems
  - Requires straightforward correlation between power & operation
- Depends on the “instruction dependency”
  - Easily defeated by adding dummy instructions/operations
- In more realistic/complex systems, there’s “too much else”
  - Lots of components being driven from same power rail
  - Signal becomes too noisy

SPA on AES?





- We can see each round…
- Not very easy to make out the 128-bit round keys!!!

Differential Power Analysis (DPA)

- A much more powerful technique
- DPA use statistics from large number of power traces
- Distinguishes a correct key from many wrong key guesses
- Exploits data dependency, not instruction dependencies
  - Much harder to stop!!

DPA: Steps

1. Measure power traces with corresponding plaintexts OR ciphertexts

2. Use the known data (plaintext OR ciphertext) with a guessed key to predict an intermediate value

3. Check if the intermediate value is reflected in the power trace:
   a. If yes, the guess is correct
   b. If no, the guess is wrong

DPA: Difference of Means

- Power differences are very small (almost impossible to detect)
  - Individual measurement errors will dominate
- So, gather large number of traces
  - Sort into two subsets (according to some guess…)
  - Calculate the average of each subset
  - If the averages are the same: guess was not distinguishing
  - If the averages were different: guess was distinguishing
    - The guess had a meaningful impact on the power consumption!

- How do we decide how to split the subsets?

Which intermediate value to select?

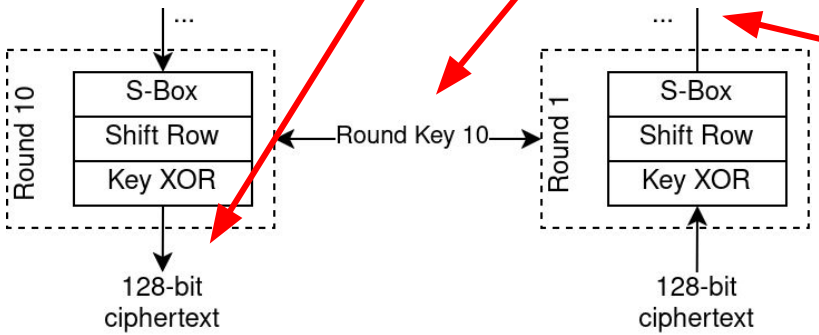
- Some value with an impact on power consumption
  - Actually everything has an impact on power
  - Dynamic power has largest swings (e.g. 0→1, 1→0)
  - It helps to know the start value (to differentiate between 0→1 and 1→0)
  - Some signals are more impactful/noisy than others
  - For block ciphers, S-Boxes!
    - S-Boxes tend to be large combinational circuits
    - Large combinational circuits have large power swings
- So, for AES/DES: Pick one bit of an output of an S-Box?

Which intermediate value to select?

- Extra challenge!
- We need to match power consumption to circuit location
- Difficult to do that for intermediate values mid-combination
- More easy to synchronise power consumption with registers

Best option? For 128-bit implementation

- Find locations which reveal info without too many unknowns
- For AES?



Can control Ciphertexts

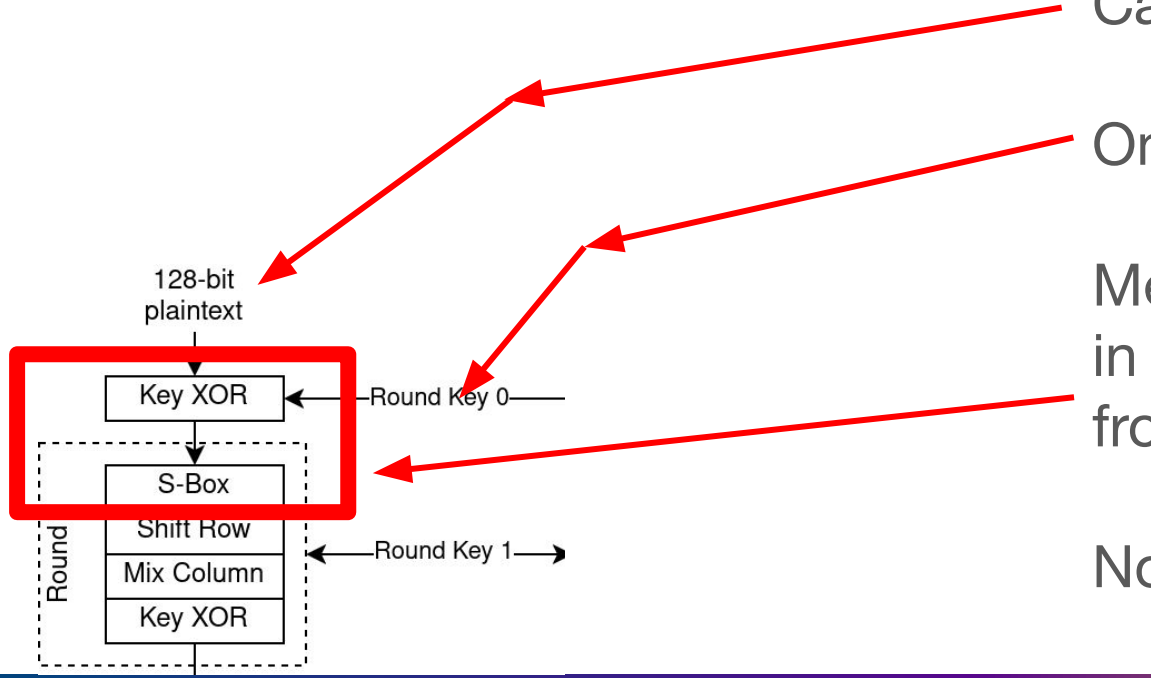
One round key (unknown)

Measure power for changes in **one bit** of R (reflects value from combinationals)

No mix columns!!

Best option? 32-bit implementation!

- Find locations which reveal info without too many unknowns
- For AES?



Can control Ciphertexts

One round key (unknown)

Measure power for changes in **one bit** of R (reflects value from combinationals)

No complex operations!!

AES/DES

- Record inputs + power traces
- Guess an S-Box input
- Partition power traces according to **one bit** of R (guess)
- See if there is a difference of means!
  - If no: guess was wrong.
  - If yes: guess was right!!

DPA Strategy

- If attack is guess and check; is this just brute force? **No.**
- Principle of attack is divide and conquer:
  - We use a statistical feature to distinguish correct/incorrect subkeys
- For DES:
  - Brute force is 2^{56}
  - DPA is 2^6 x 8 as we can guess 6-bit subkeys
- For AES:
  - Brute force is 2^{128}
  - DPA is ?

DPA Strategy

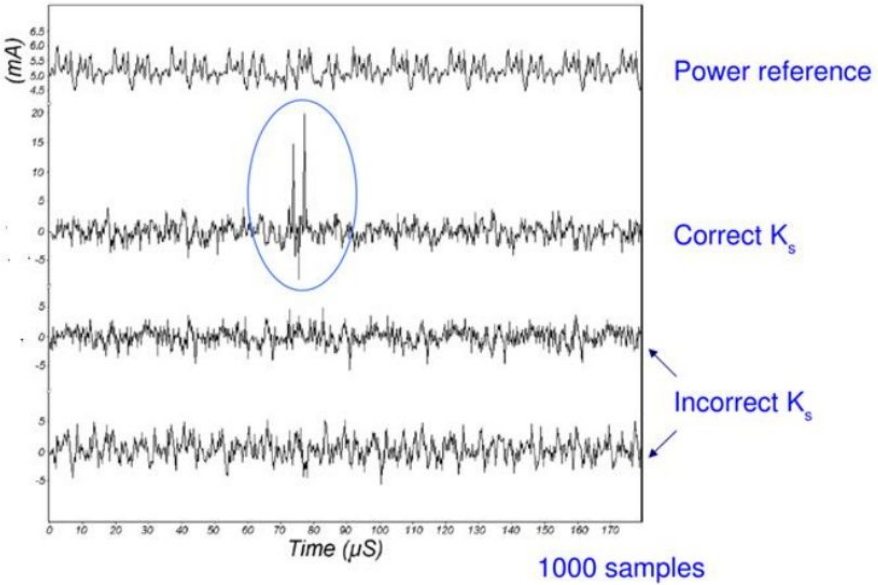
- If attack is guess and check; is this just brute force? **No.**
- Principle of attack is divide and conquer:
  - We use a statistical feature to distinguish correct/incorrect subkeys
- For DES:
  - Brute force is 2^{56}
  - DPA is 2^6 x 8 as we can guess 6-bit subkeys
- For AES:
  - Brute force is 2^{128}
  - DPA is 2^8 x 16 as we can guess 8-bit subkeys

DPA example

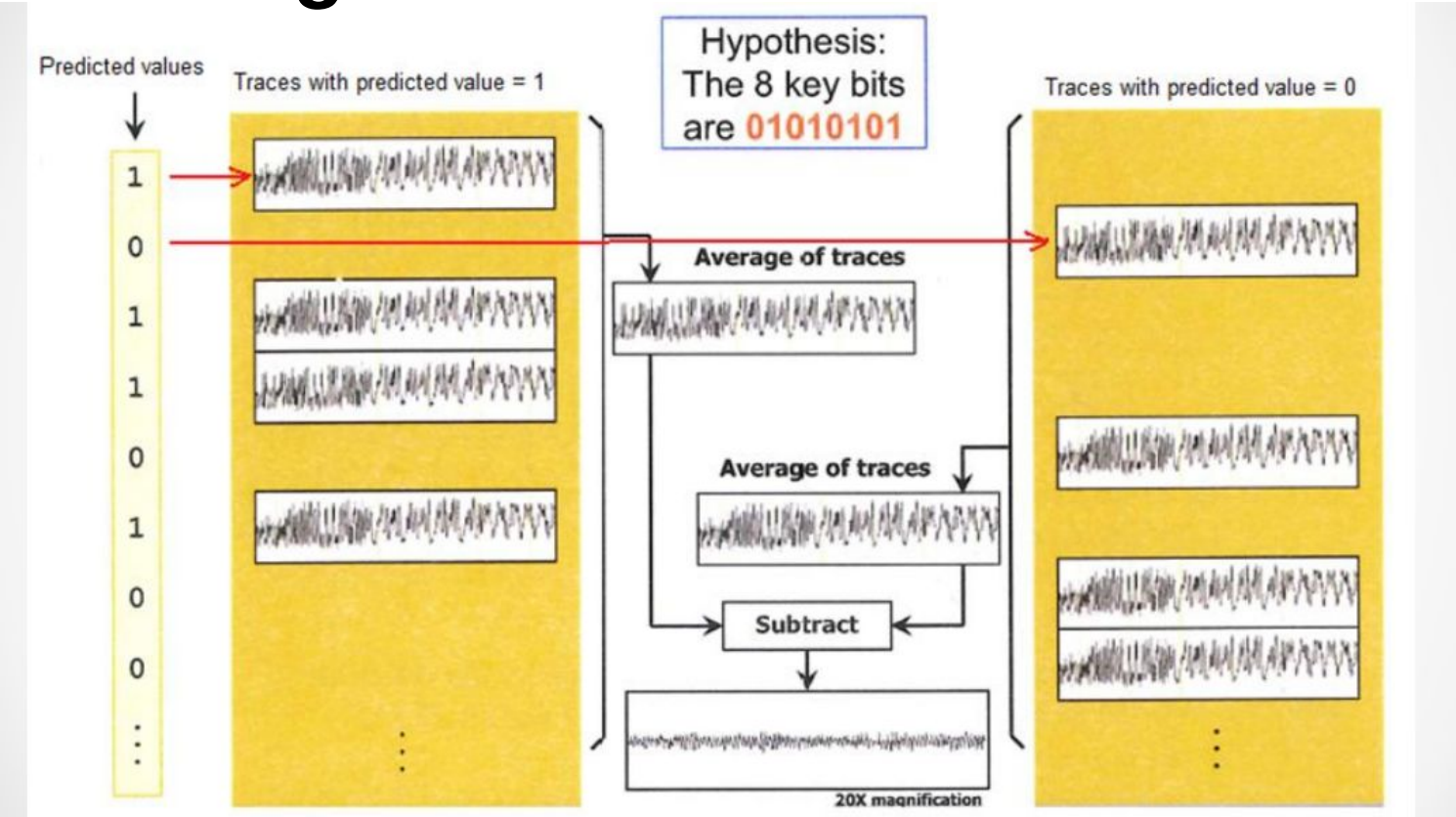
- Given traces Tr split into Tr_1, Tr_2
  - (according to your key guess)
- For each sample point j, compute:

$$\Delta_D[j] = \frac{\sum Tr_1[j]}{|Tr_1|} - \frac{\sum Tr_0[j]}{|Tr_0|}$$

- If key is wrong: delta close to zero
  - (There would be no correlation)
- If key is right: see a large spike
  - (Power consumption difference for an active switch 0→1 or 1→0)

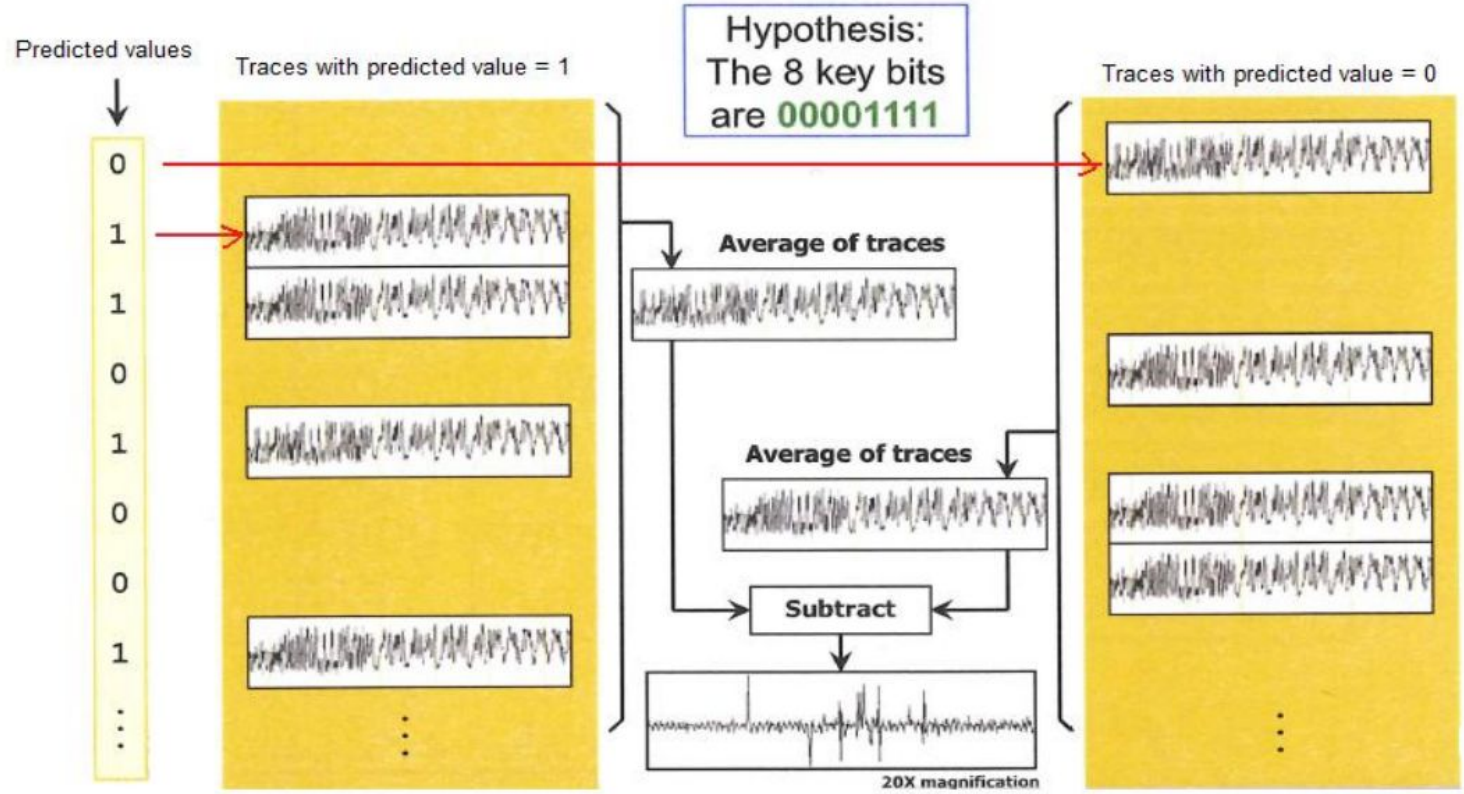


DPA - bad guess



https://pdfs.semanticscholar.org/e92d/50e613fc38849cf39940eaeea914869c6c51.pdf

DPA - good guess



https://pdfs.semanticscholar.org/e92d/50e613fc38849cf39940eaeea914869c6c51.pdf

DPA on AES

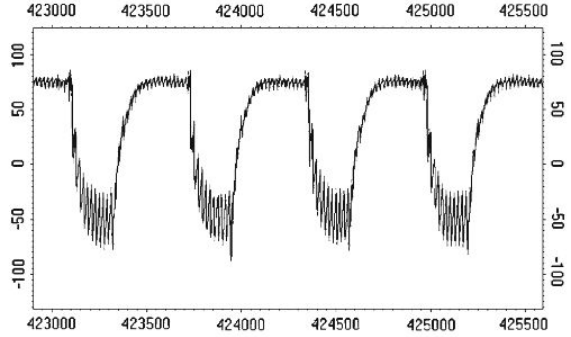


Fig. 2 Part of a power trace from an FPGA performing AES-128 CBC mode encryptions. In power side measurement, down corresponds to more power

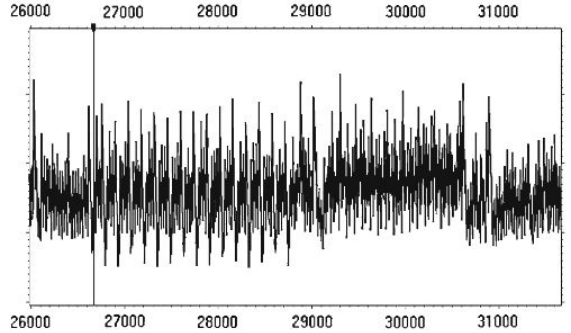


Fig. 3 Power trace segment showing the first round of AES-128 encryption on a smart card. A vertical line marks the location of first S-box lookup

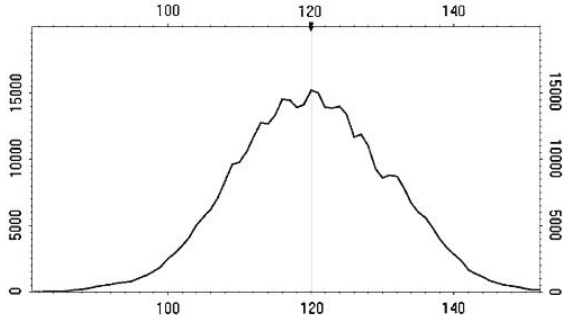


Fig. 4 Distribution of power consumption at first S-box output computation

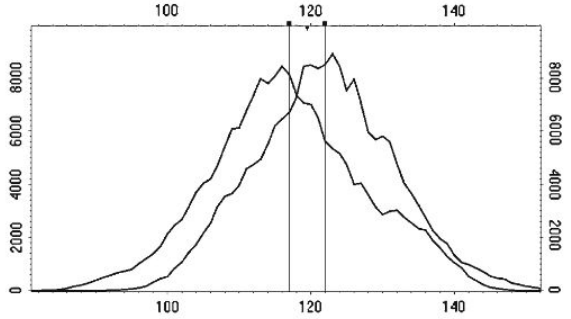


Fig. 5 Distributions of power consumption measurements for traces with the LSB of the output of the first S-box being 1 (left) and 0 (right)

https://link.springer.com/article/10.1007/s13389-011-0006-y

Correlation Power Analysis (CPA)

- Actually, DPA over 1 bit is quite hard - not much signal
- Can we analyse more than 1 bit at once?

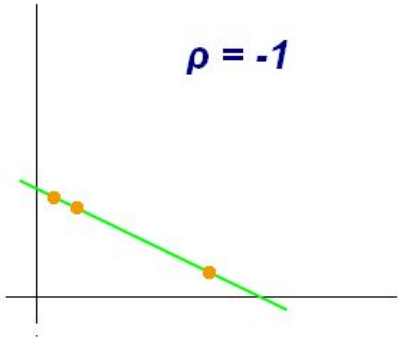
Correlation Power Analysis (CPA)

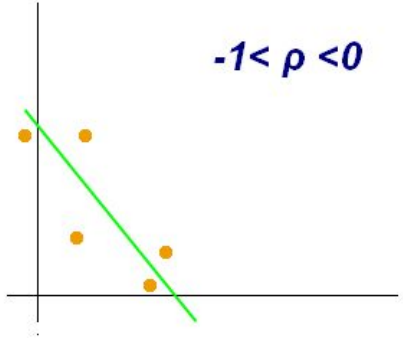
- Actually, DPA over 1 bit is quite hard - not much signal
- Can we analyse more than 1 bit at once? **YES**
- CPA uses a Pearson correlation as a metric to distinguish a correct key guess from the wrong key guesses
- CPA uses leakage of multiple bits - more signal, more efficiency
- In order to do this, we need to correlate with a *power model*

Brief aside: What is correlation?

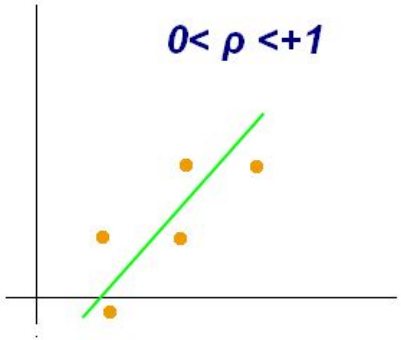
https://www.youtube.com/watch?v=dsyTQNUvqH0

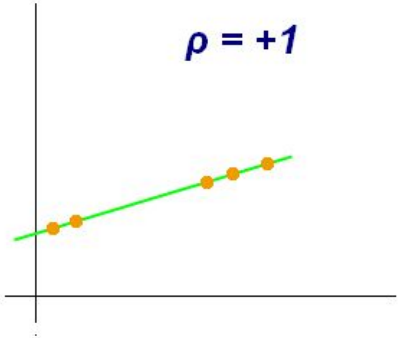
Pearson Correlation Coefficients

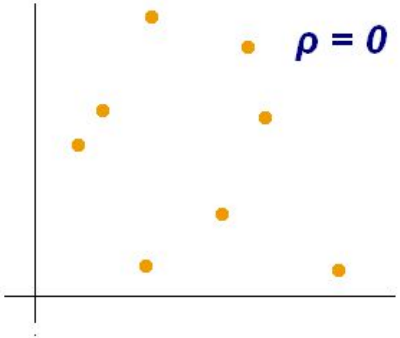




$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$







https://en.wikipedia.org/wiki/Pearson_correlation_coefficient#/media/File:Correlation_coefficient.png

How do we correlate encryption with power consumption?

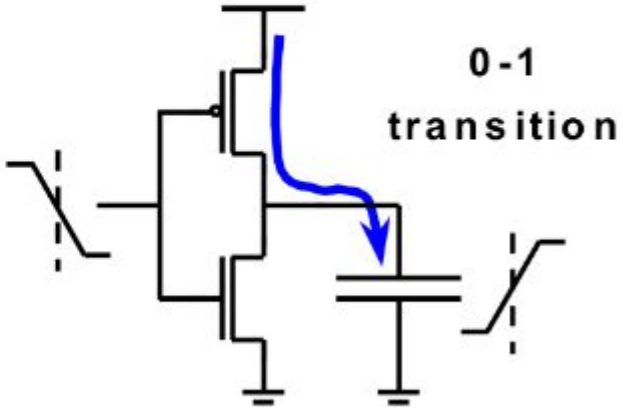
We need a power model

Recap: Power consumption reflects data

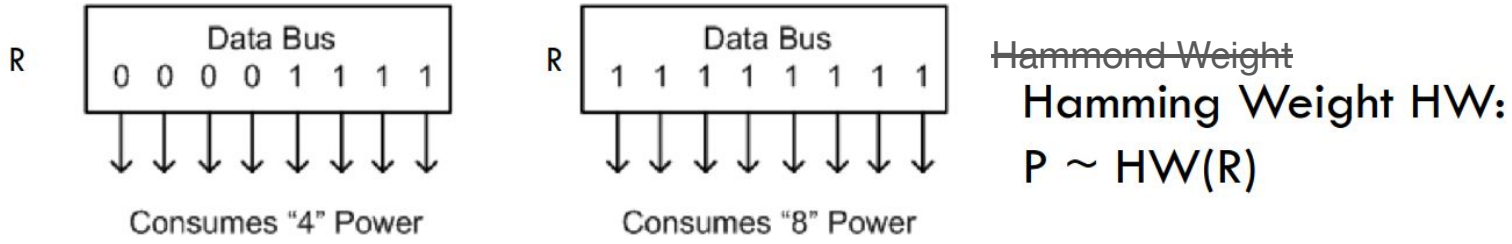
- Data transitions (switching) consume more power than idling:
  - CMOS inverter:

a=>a, b=>b Capacity

Input	Output	Current
0 → 0	1 → 1	Low
0 → 1	1 → 0	Discharge
1 → 0	0 → 1	Charge
1 → 1	0 → 0	Low

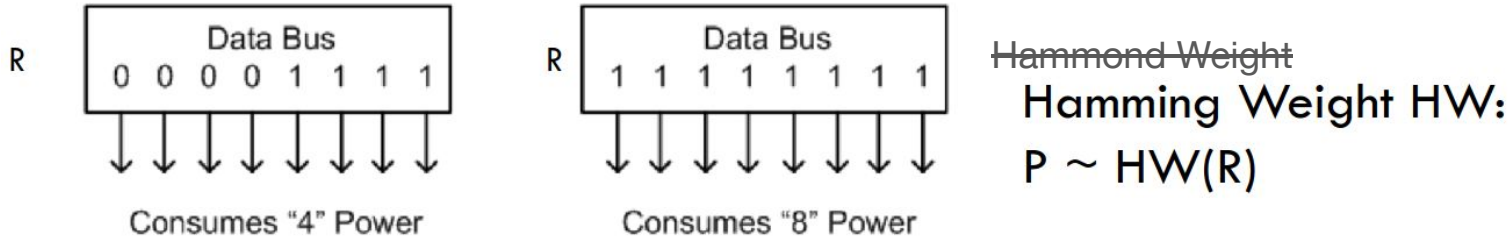


Power models: HW and HD



Example showing Hamming Weight (HW). HW is ~ #1s ~ power.

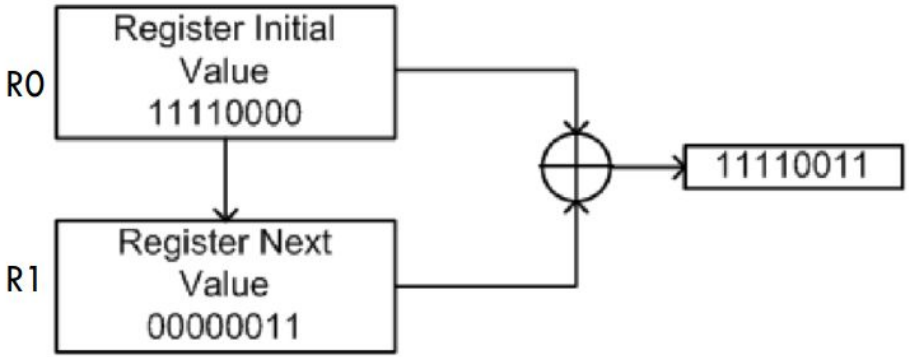
Power models: HW and HD



Example showing Hamming Weight (HW). HW is ~ #1s ~ power.

But this is idle power - i.e., not very much.

Power models: HW and HD

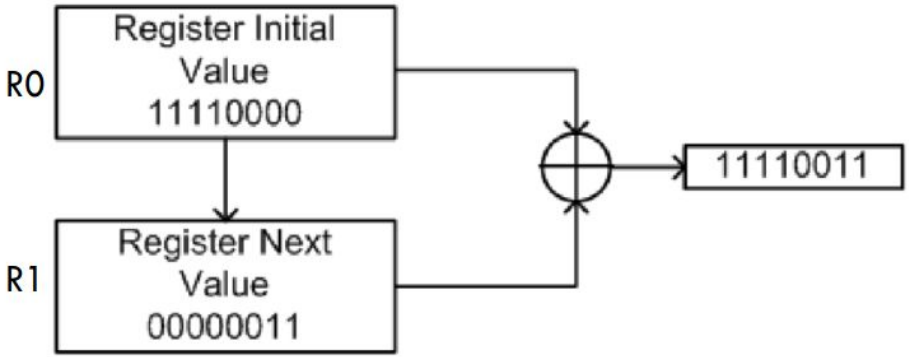


Hamming Distance HD:

P ~ HD(R0,R1)=HW(R0 XOR R1)

Example showing Hamming Distance (HD).

Power models: HW and HD



Hamming Distance HD:

P ~ HD(R0,R1)=HW(R0 XOR R1)

Example showing Hamming Distance (HD).

This is switching power - a higher power value.

Hamming Distance

- “Every time a bit is changed from a 0 to a 1 (or vice versa), some current is required to (dis)charge the data lines.”
- Hamming Distance: # of different bits (number of changes)
- HammingDistance(00110010, 00100011) = 2

If we can find a point in the encryption algorithm where the victim changes a variable from x to y, then we can estimate that the power consumption is proportional to Hamming Distance(x, y)

Hamming Distance

- “Every time a bit is changed from a 0 to a 1 (or vice versa), some current is required to (dis)charge the data lines.”
- Hamming Distance: # of different bits (number of changes)
- HammingDistance(00110010, 00100011) = 2

NOTE: the HD is adimensional!
It does not tell us anything about a single measurement,
but we can use it to compare different measurements!

What do we need to compute HD?

- We need to know what a target's ORIGINAL value is
- We need to know what a target's NEXT value is

What do we need to compute HD?

- We need to know what a target's ORIGINAL value is
- We need to know what a target's NEXT value is

E.g.:

We have a 32-bit register which is reset to all 0's

It loads the value 0xC0DE

What's the HD?

What do we need to compute HD?

- We need to know what a target's ORIGINAL value is
- We need to know what a target's NEXT value is

E.g.:

We have a 32-bit register which is reset to all 0's

It loads the value 0xC0DE

What's the HD? - all 0 becomes 1100 0000 1101 1110

- there are eight 1s → HD = 8

How do we correlate with HD?

- We can model the HD in each clock cycle
  - (e.g. by guessing the result of a computation)
- We then measure the real power consumption observed
- Can we check if the model matches the observation?
  - Remember that the HD is **adimensional** !!
  - a single reading can’t inform us!

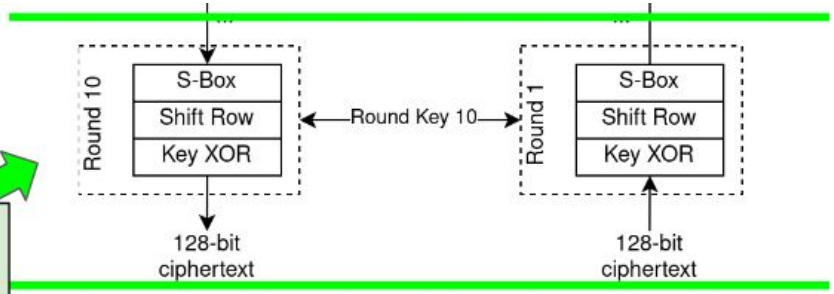
A better strategy

- Let's make *many* guesses
  - E.g. let's work out what the power consumption would be:
  - For many clock cycles
  - For many different options
- Then, we correlate against each guess
- The best guess will have the best correlation!

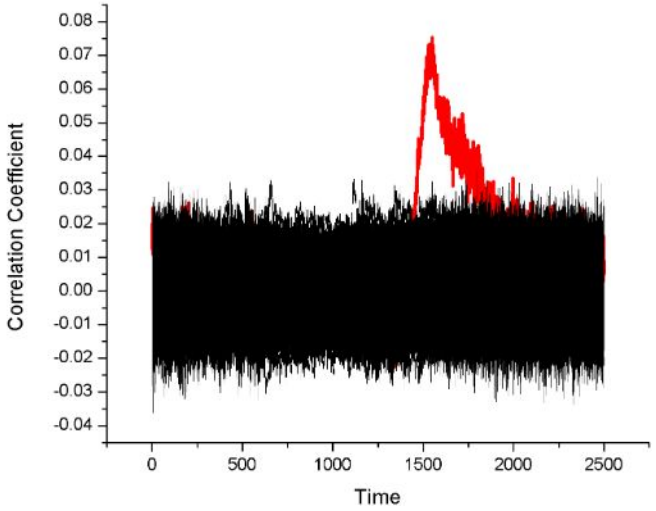
Applying this strategy to encryption

- Let's guess a byte of the Key
  - Compute the theoretical HD against a byte of input
  - Then do it again for another byte of input… and another… and another…
  - Then, guess a different key byte, and repeat…
- You will end up with traces of theoretical power consumptions
  - One for each possible key byte
- Then, see which trace correlates ***best*** with the observed power!

CPA: AES



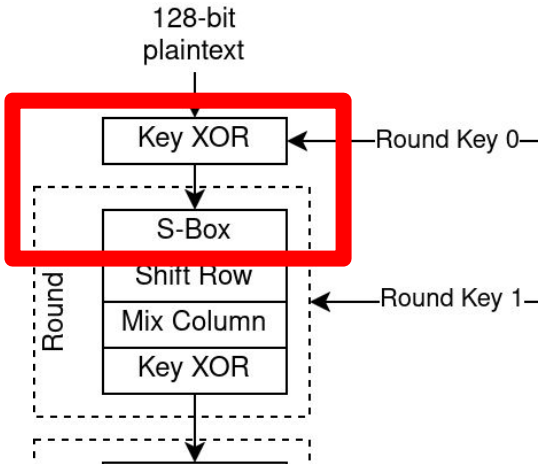
Here has 1 round key (and no mix column!)

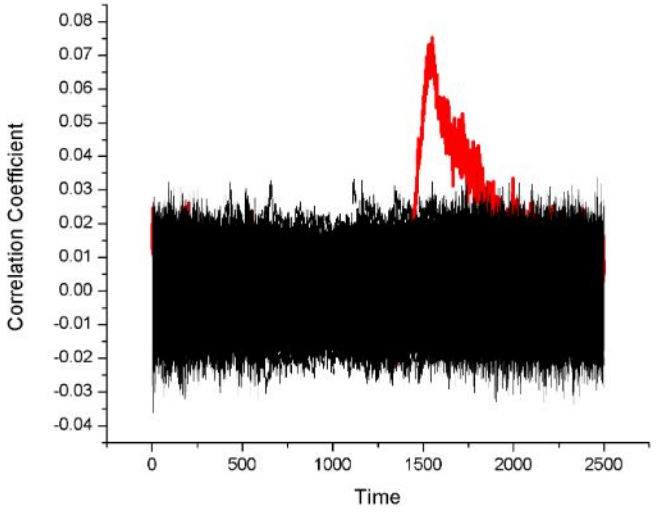


- Same as DPA; we could target last round and do byte-by-byte
- Same math as with DES (but over different ops)
- Example: Extracting 1 byte of RK

CPA: AES (32-bit)

If AES HW implementation breaks up each round, we have an easier option!





- We target “after the S-Box”
  - in a 32-bit implementation
- Then do byte-by-byte analysis:
  - Correlate actual power consumption vs. the hamming model
- Best match: 1 byte of RK

CPA: AES (32-bit)

$$X_i = HD(\ S\text{-}Box\ (RK0 \otimes PTi)\ )$$

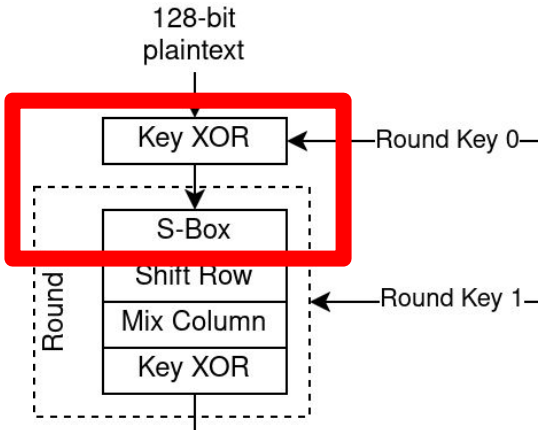
Ti - The power trace for *PTi*
Ti[j] - The jth element of the trace *PTi*

$$Y_{i,j} = T_i[j], j = 1, \ldots, k$$

$$\bar{X} = \sum_{i=1}^{m} X_i/m$$

$$\bar{Y_j} = \sum_{i=1}^{m} Y_{i,j}/m$$

$$r_j = \frac{\sum_{i=1}^{m}(X_i - \bar{X}) \times (Y_{i,j} - \bar{Y_j})}{(\sqrt{\sum_{i=1}^{m}(X_i - \bar{X})^2}) \times (\sqrt{\sum_{i=1}^{m}(Y_{i,j} - \bar{Y_j})^2})}$$



1. Collect power traces T over Plaintexts PT
   a. 2D array of *PTi* / *Ti[j]*, i.e.
   b. Each plaintext *PTi* has j power cap. in *Ti*
   c. Let $Y_{i,j}$ be the power trace for *PTi*
2. Guess a Round Key byte *RK0*
3. For each plaintext *PTi* / *RK0*, compute HD *Xi*
   a. 2D array of *PTi* / HD for the guess *RK0*
4. Correlate each *Xi*, with observed power $Y_{i,j}$
5. Which is the best guess? The best correlation!
6. Repeat 2-5 for each byte of Key RK0!

Break: 5 minutes

Break activity: Talk to your neighbour -

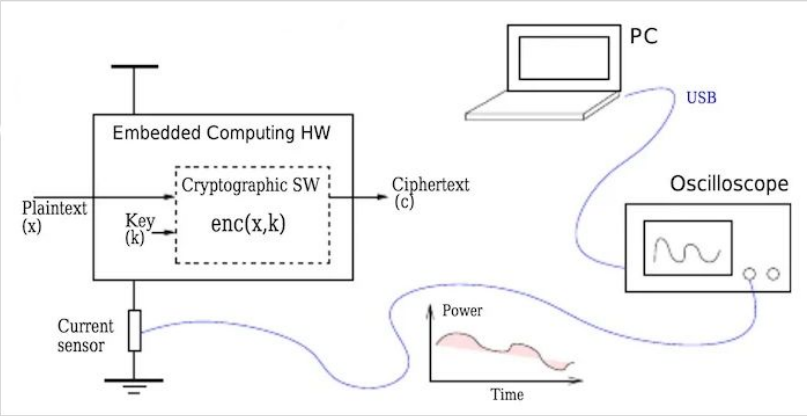
hi
So, that power analysis, right?

I guess you could say…
it's watt's cracking in cryptography these days

Part 2: Power Defenses

Threat model

- Assume a passive, non-invasive attacker measuring power
- Distinguish between:
  - horizontal (single-trace SPA) and
  - vertical (multi-trace DPA/CPA)
- Attacker can use profiling (training on identical devices)
  - Or non-profiling! (e.g. HW/HD)



Countermeasures (broadly)

1. Detection
    a. Can we detect a measurement before it occurs and flush secrets?
2. Hiding
    a. Conceal data-dependent power consumption
3. Masking
    a. Randomizes intermediate values
4. Key management
    a. Frequent key changes to prevent effective modeling

Detecting power analysis?

- Power/voltage sensors
  - Analog monitors in power delivery network - flag anomalies
  - Can even use an ML model on-chip!
    - E.g. F. Kenarangi and I. Partin-Vaisband, "Security Network On-Chip for Mitigating Side-Channel Attacks," 2019 ACM/IEEE SLIP, 2019
    - https://ieeexplore.ieee.org/abstract/document/8771328

Other sensors possible…

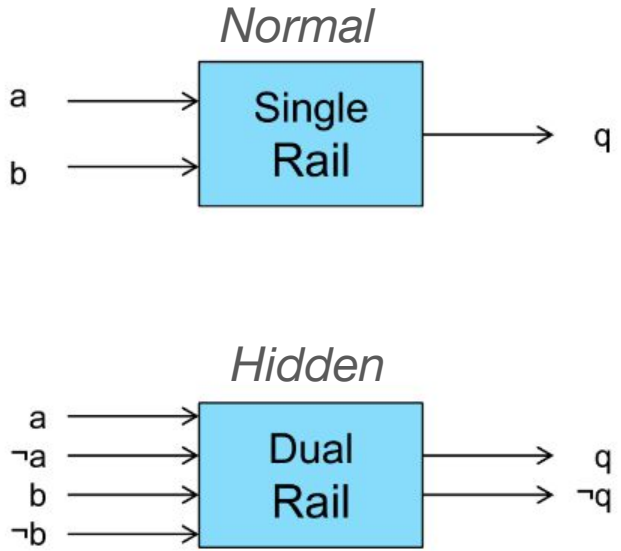
- Impedance Monitoring
  - An attached probe or leakage path changes observed impedance
  - Detect measurement before enough traces are captured!
    - E.g. R. Munny and J. Hu, "Power Side-Channel Attack Detection through Battery Impedance Monitoring," ISCAS 2021
    - https://ieeexplore.ieee.org/document/9401542

Detection - downsides

- Huge overhead in area/power
- Complex design
- Stochastic (not guaranteed to work)

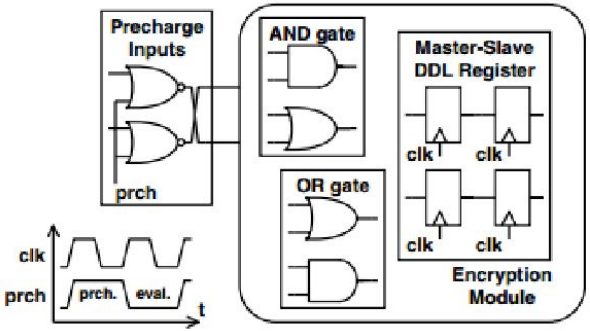
Hiding via Power-Equalization (Balancing)

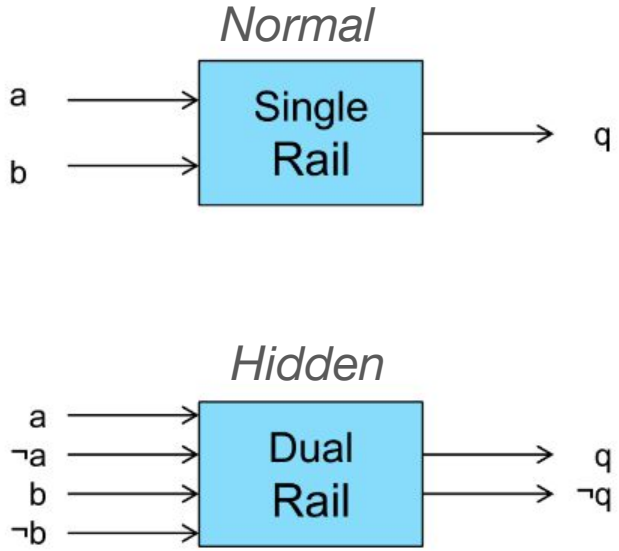
- Balancing power consumption by including *op* and NOT(*op*)
- Two wires carry one bit of information
  - Dual-rail pre-charging



Hiding via Power-Equalization (Balancing)

- Balancing power consumption by including *op* and NOT(*op*)
- Two wires carry one bit of information
  - Dual-rail pre-charging
- Clock divided into two-phases
  - Precharging - q and not(q) into same value
  - Evaluation - q and not(q) go to output





Balancing - Downsides

- Larger area (often 2x !)
- Higher power consumption (could be 2x !)
- More complex circuit routing
  - Need to preserve balancing, minimise glitching

Balancing - Some works

- Tiri, Kris, and Ingrid Verbauwhede. "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation." Proceedings Design, Automation and Test in Europe Conference and Exhibition. Vol. 1. IEEE, 2004.
    - https://ieeexplore.ieee.org/abstract/document/1268856
- Choi, Byong-Deok, et al. "Symmetric adiabatic logic circuits against differential power analysis." ETRI journal 32.1 (2010): 166-168.
    - https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.10.0209.0247
- Avital, Moshe, et al. "Randomized multitopology logic against differential power analysis." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 23.4 (2014): 702-711.
    - https://ieeexplore.ieee.org/abstract/document/6825857

Hiding via Electrickery

- Adjustable current sources or shunt-feedback loops can compensate for instantaneous current draws

- Passive filters (capacitors/inductors) can also suppress spikes to minimise variations heading to the sensor

Might be expensive or impossible to do on-chip!

Might be possible to overcome if on-PCB!

Masking - Noise Injection

- Random noise generator on-chip (e.g. hash engine, R-Os)
  - Run continuously to superimpose over cryptographic ops
  - Decorrelates actual signal
    - E.g. Zhu, Nianhao, Yujie Zhou, and Hongming Liu. "Counteracting leakage power analysis attack using random ring oscillators." ICSNSTPC, 2015
    - https://ieeexplore.ieee.org/abstract/document/6553838

- Correlated noise
  - Counters / PRNGs tied to crypto ops to inject data-independent toggles
    - E.g. Alipour, Amir, et al. "On the performance of non-profiled differential deep learning attacks against an AES encryption algorithm protected using a correlated noise generation based hiding countermeasure." DATE, 2020.
    - https://past.date-conference.com/proceedings-archive/2020/pdf/0472.pdf

Masking - Clock / Voltage Randomization

- Varying / gating system clock → different timings → dif. Power
  - E.g. randomly skip clock pulses, use multiple-phase clocks
- DVFS
  - Use a feedback regulator to randomize frequency and voltage
- Break trace alignments

E.g. K. H. Boey, Y. Lu, M. O'Neill, and R. Woods, "Random clock against differential power analysis," in Asia Pacific Conference on Circuits and Systems, 2010, pp. 756–759.

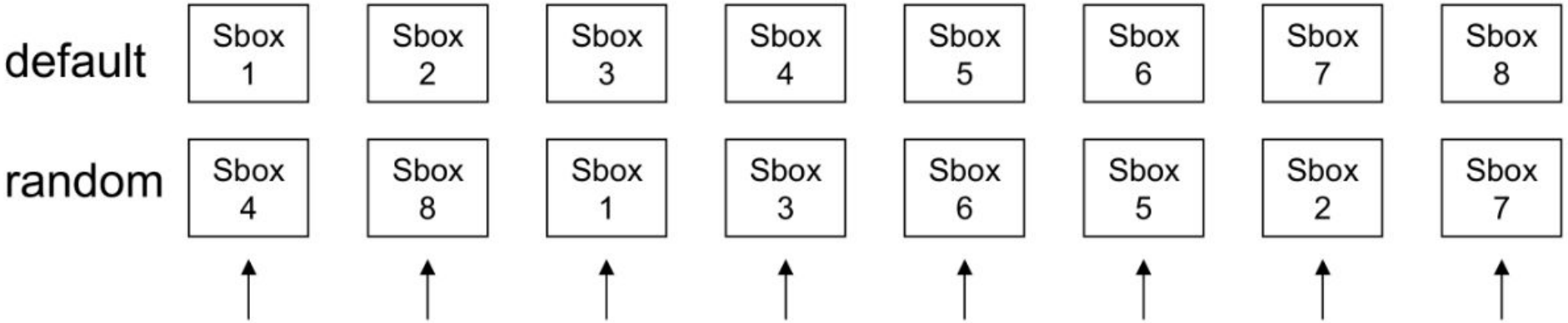
Masking - Random “shuffling”

- Randomly permute the order of independent operations (e.g. S-box encryptions) each encryption.
- Similarly, we can randomly add dummy instructions / no-ops

- This “time hiding” → misaligned traces

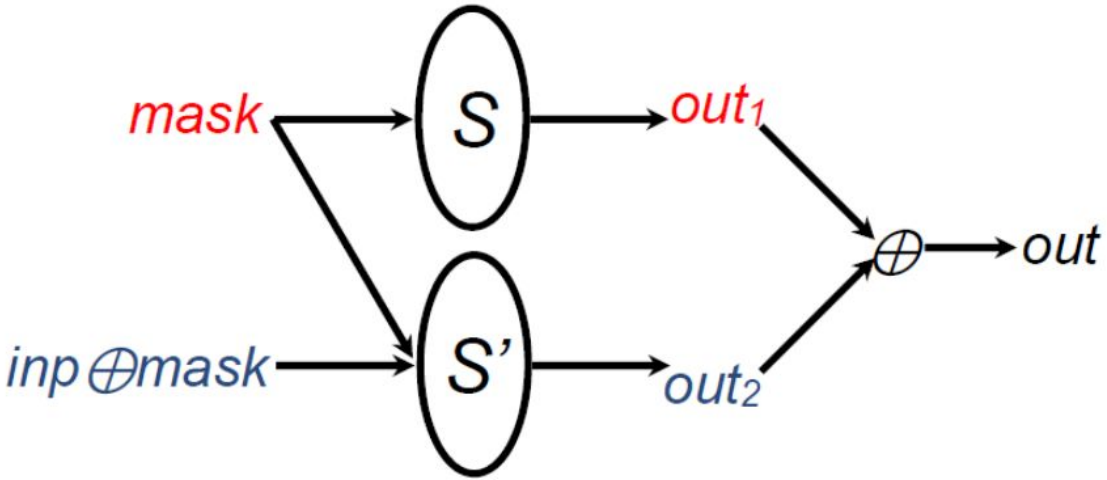
Shuffling example

- Reorder data-independent operations
- Works better for some architectures over others
  - (E.g. nothing in our DES or AES is data-independent)



Masking - random data manipulation

- Only input and outputs are in the implementation
- All intermediate values are masked (XORed) with random #s



Masking

- Different techniques have different area/power tradeoffs
- All come with a cost
  - Shuffling can also add a *performance* cost - those no-ops take time!

- For high-value secrets, might be worth it!
  - Combining defenses common
  - Even adding power sensors!

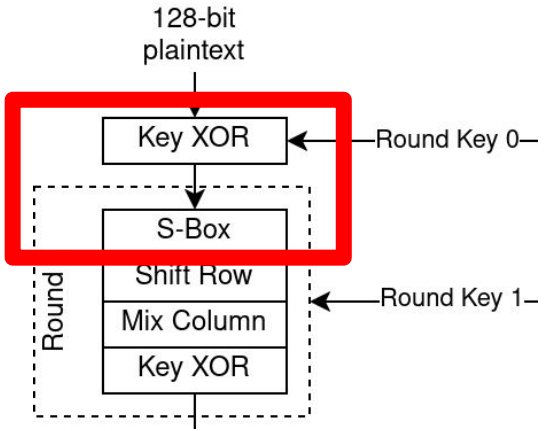
Good defenses can “prevent” power analysis!

Assessments: Week 4 (Power Attack)

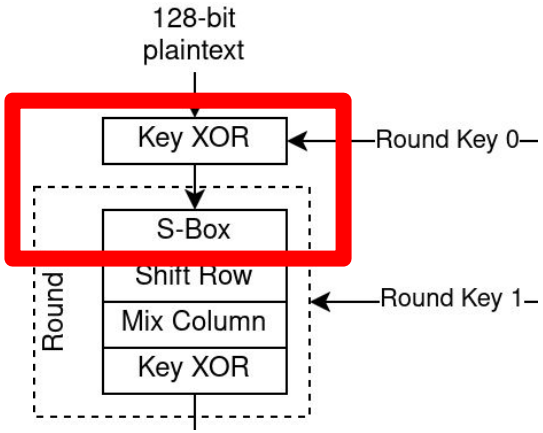
Lab 2

Lab 2: CPA on AES

- A 32-bit implementation of AES is provided
- It needs to save intermediate values of encryption…
- Normally this is safe!
- But when it comes to power analysis, it leaves an opening for analysis!



Lab 2: CPA on AES (32-bit)



$$X_i = HD(\ S\text{-}Box\ (RK0 \otimes PTi)\ )$$

Ti - The power trace for *PTi*
Ti[j] - The jth element of the trace *PTi*

$$Y_{i,j} = T_i[j], j = 1, \ldots, k$$

$$\bar{X} = \sum_{i=1}^{m} X_i/m$$

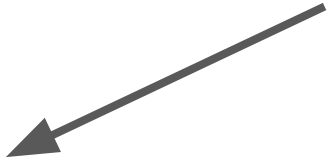
$$\bar{Y_j} = \sum_{i=1}^{m} Y_{i,j}/m$$

$$r_j = \frac{\sum_{i=1}^{m}(X_i - \bar{X}) \times (Y_{i,j} - \bar{Y_j})}{(\sqrt{\sum_{i=1}^{m}(X_i - \bar{X})^2}) \times (\sqrt{\sum_{i=1}^{m}(Y_{i,j} - \bar{Y_j})^2})}$$

1. Collect power traces T over Plaintexts PT
   a. 2D array of PTi / $Ti[j]$, i.e.
   b. Each plaintext PTi has j power cap. in Ti
   c. Let Yi,j be the power trace for PTi
2. Guess a Round Key byte $RK0$
3. For each plaintext PTi / $RK0$, compute HD Xi
   a. 2D array of PTi / HD for the guess $RK0$
4. Correlate each Xi, with observed power Yi,j
5. Which is the best guess? The best correlation!
6. Repeat 2-5 for each byte of Key RK0!

Lab 2: CPA on AES (32-bit)

- Two generic implementations provided:
  - /examples/spi_aes
  - /examples/power_aes_no_io
- Two student-specific files:
  - /[your zID]_lab2_spi_aes.bin
  - /[your zID]_lab2_power_aes_no_io.bin



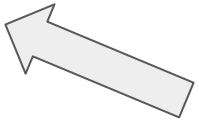
These have the same unique key

The power one takes has no IO and can be used to extremely quickly generate thousands of power traces (inputs are fed by LFSR)

Lab 2: CPA on AES

1. Understand the two generic implementations
2. Download and obtain your implementations
3. Collect the power traces (lab docs have further info. as well)
4. Use CPA to crack the internal key
5. Validate your answer using the SPI version (optional)

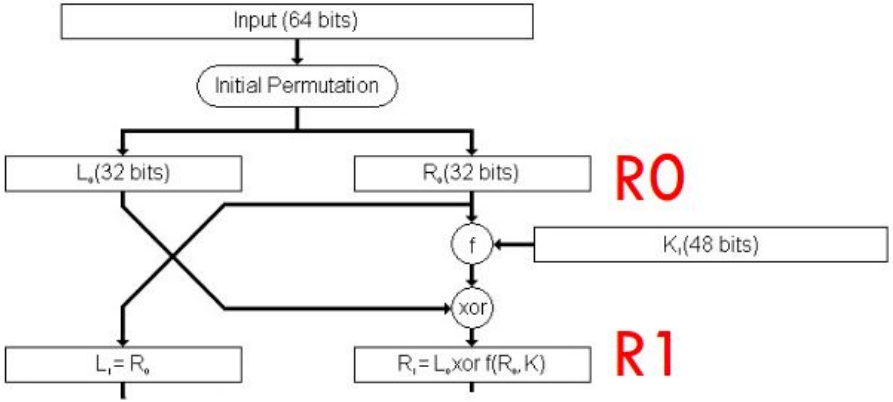
```
CPA completed  : [00] [11] [22] [33] [44] [55] [66] [77] [88] [99] [AA] [BB] [CC] [DD] [EE] [FF]
```

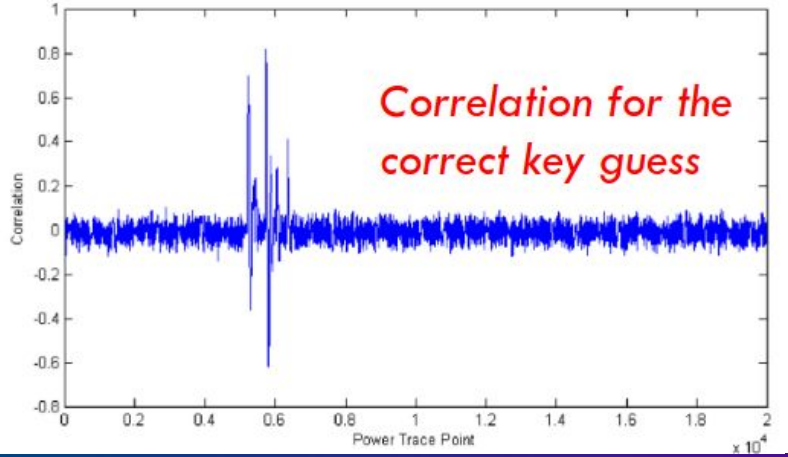


So you believe me that it's possible, my script when cracking /examples/power_aes_no_io

Appendix: CPA on DES

CPA: HD for DES





$$X_i = HD(R0, R1) = HW(R_0 \oplus f(R_0 \oplus K) \oplus L_0)$$

We guess K (we know R0, L0 representing the plaintext corresponding to the i-th power trace)

$$Y_{i,j} = T_i[j], j = 1, \ldots, k$$

$$\bar{X} = \sum_{i=1}^{m} X_i / m$$

$$\bar{Y_j} = \sum_{i=1}^{m} Y_{i,j} / m$$

$$r_j = \frac{\sum_{i=1}^{m}(X_i - \bar{X}) \times (Y_{i,j} - \bar{Y_j})}{(\sqrt{\sum_{i=1}^{m}(X_i - \bar{X})^2}) \times (\sqrt{\sum_{i=1}^{m}(Y_{i,j} - \bar{Y_j})^2})}$$