



Advanced side-channel attacks: DPA & Countermeasures

Hardware security, Spring 2018

Lejla Batina

March 21, 2018

Institute for Computing and Information Sciences
Radboud University

Introduction

Differential Power Analysis (DPA)

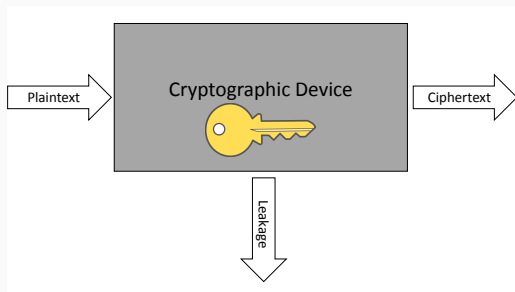
Correlation Power Analysis (CPA)

Countermeasures

- ▶ Lecture 1: Introduction to SCA, March 5
- ▶ Lecture 2: Advanced SCA and countermeasures, March 19
- ▶ Assignment 1: posted on March 19, April 2 deadline
- ▶ Assignment 2: posted on March 26, April 9 deadline
- ▶ Lecture 3: Tutorial on template attacks, March 26
- ▶ Assignment 3: posted on April 2, deadline April 16
- ▶ Lecture 4: Fault injection attacks, May 14
- ▶ Excursion to Riscure: tba
- ▶ DiS SCA lab tours: March 26, groups via Doodle

Introduction

- ▶ The cryptographic algorithm is **implemented on a real device** such as a processor, microcontroller, FPGA etc.
- ▶ We can observe certain *physical quantities* in the device's vicinity and use the additional information during cryptanalysis
- ▶ Can you derive the secret key by observing plaintext/ciphertext pairs **and a side-channel**?
- ▶ The *side-channel* is any unintentional signal that can offer us a blurry view of the algorithm's internal computations
- ▶ Execution time, power consumption, electromagnetic emission, sound and others



- ▶ We have limited access to the internal computations thus we work with a greybox scenario
- ▶ Algorithms that are secure under a blackbox scenario may be not secure under the greybox i.e. they may have implementations that are not secure
- ▶ **Side-channel attacks** are attacks on implementations of algorithms

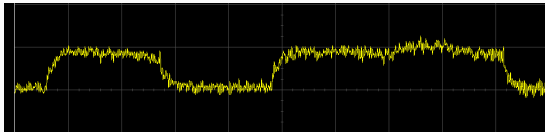
Active vs passive:

- ▶ Active i.e. tampering: the key is recovered by exploiting some abnormal behavior e.g. power glitches or laser pulses
- ▶ **Passive** i.e. eavesdropping: the device operates within its specification

Invasiveness:

- ▶ Invasive aka expensive: the strongest type e.g. bus probing
- ▶ Semi-invasive: the device is de-packaged but no direct contact with the chip e.g. optical attacks that read out memory cells (or faults/glitches by voltage, power supply, clock, EM, etc.)
- ▶ **Non-invasive** aka low-cost:
 - power/EM measurements
 - data remanence in memories – cooling down is increasing the retention time
 - Rowhammer

Power side-channel



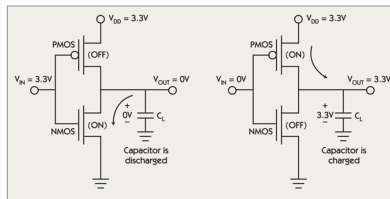
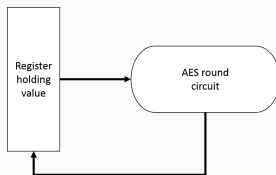


Fig. 1 Dynamic power in a CMOS inverter.

- ▶ CMOS is the most popular circuit style and exhibits several types of leakage
- ▶ The most relevant for side-channel attacks is the charge and discharge of the CMOS load capacitance a.k.a dynamic power consumption
- ▶ Dynamic power consumption (P_{dyn}) is produced by CMOS transitions from state 0 to 1 and from state 1 to 0
- ▶ Thus a power analysis attack explores the fact that the dynamic power consumption depends on the data and instructions being processed
- ▶ $P_{dyn} = CV_{DD}^2 P_{0 \rightarrow 1} f$,
where C the transistor capacitance, V_{DD} the power supply voltage, f the frequency and $P_{0 \rightarrow 1}$ the probability of a $0 \rightarrow 1$ transition

- ▶ Starting point: dynamic power consumption depends on bit transitions thus we use the number of transitions to model the leakage
- ▶ The Hamming distance model counts the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions, assuming that they are equivalent
- ▶ Example 1: Assume a hardware register R storing the result of an AES round. The register initially contains value v_0 and gets overwritten with value v_1

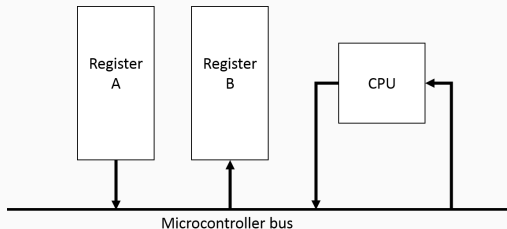


- ▶ The power consumption because of the register transition $v_0 \rightarrow v_1$ is related to the number of bit flips that occurred
- ▶ Thus it can be modeled as $\text{HammingDistance}(v_0, v_1) = \text{HammingWeight}(v_0 \oplus v_1)$
- ▶ It's common to see Hamming distances in hardware implementations (FPGA, ASIC)

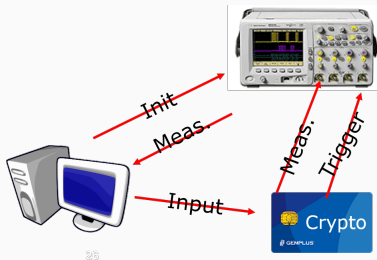
Power side-channel: Modeling the leakage

- Example 2: In a microcontroller, assume register A with value v_0 and an assembly instruction that moves the contents of register A to register B

```
mov rB, rA
```

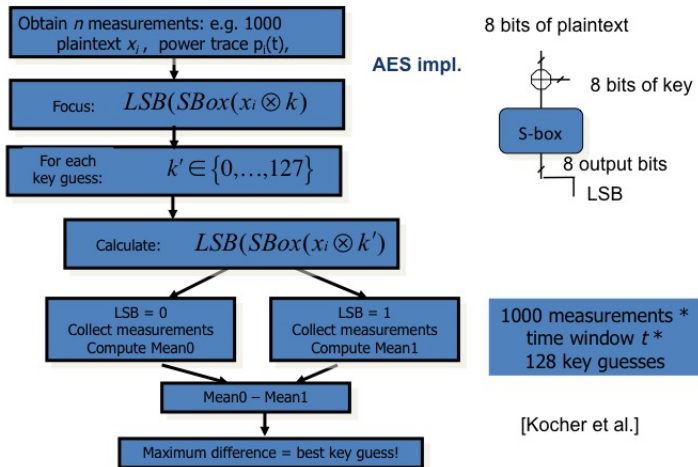


- In general-purpose processors the instruction will transfer value v_0 from register A to B via the CPU, using the bus
- In several cases the bus is very leaky component and it is also precharged at all bits being zeros or all being one (busInitialValue)
- The power consumption of the assembly instruction can be modeled as $\text{HammingDistance}(\text{busInitialValue}, v_0) = \text{HammingWeight}(v_0 \oplus 0) = \text{HW}(v_0)$
- It's common to see Hamming weight leakages in software implementations (AVR/ARM microcontrollers)



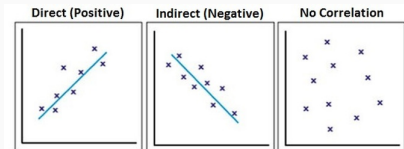
- Usually power measurements requires physical proximity to the device and customized measurement equipment (resistor, oscilloscope)

Differential Power Analysis (DPA)



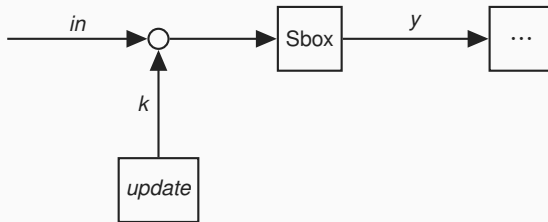
Correlation Power Analysis (CPA)

Correlation Power Analysis (CPA)



- ▶ The most popular side-channel attack, still used often everywhere i.e. universities and evaluation labs
- ▶ Aims at recovering the secret key by using a large number of power measurements (a.k.a. traces)
- ▶ A fairly generic procedure

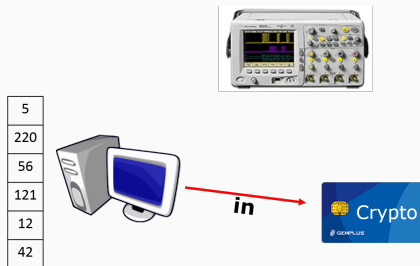
CPA Step 1: Choose intermediate value



- ▶ Assume a software AES implementation (as shown above)
- ▶ The implementation is built for AVR microcontrollers which have a word size of 8 bits, i.e. the operation above is executed one byte at a time
- ▶ Step 1: Choose an intermediate value v of the AES cipher to attack
- ▶ The value v must be a function of the input and the key, i.e. $v = f(in, k)$
- ▶ A common choice for intermediate value is the Sbox output, i.e.
 $v = y = Sbox(in \oplus k)$
- ▶ The targeted value v must be fairly small. Our AES implementation works with 8-bit words which is manageable
- ▶ Throughout our attack the key k must remain constant
- ▶ Throughout our attack the input in is random but known

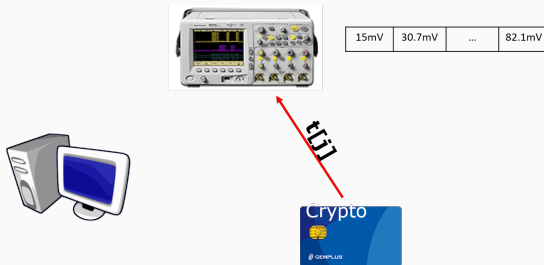
CPA Step 2: Measure the power consumption

- ▶ Step 2 measures the power consumption for multiple random inputs
- ▶ In your measurement setup generate randomly n 8-bit inputs. Typically n is large (thousands to millions!)
- ▶ Store the inputs in vector $\mathbf{in} = [in_1 in_2 in_3 \dots in_n]^T$

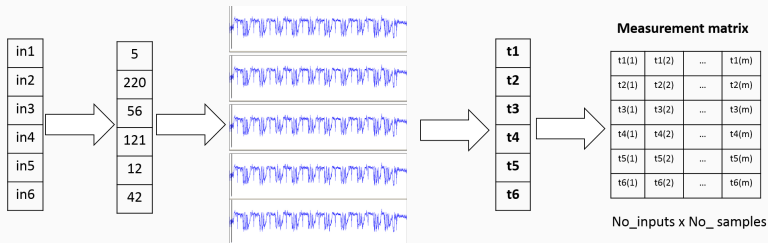


CPA Step 2: Measure the power consumption

- ▶ For every generated 8-bit input we measure the power consumption of the AES implementation over time
- ▶ Thus for every input $in_j, j = 1, \dots, n$ we capture a digitized trace over time
- ▶ We denote the trace w.r.t. input in_j as $\mathbf{t}_j = [t_j^1 t_j^2 \dots t_j^m]^\top$. It contains m points in time (a.k.a. samples)



- ▶ E.g. capturing 6 power traces with m time points (samples) results in the following measurement matrix



- ▶ Note that the power traces originate from the device, i.e. they are related to the secret key stored inside the device
- ▶ We will refer to the unknown key that is stored in the device as k_{dev}

CPA Step 3: Hypothetical intermediate values

- ▶ In our device $y = \text{Sbox}(in \oplus k_{dev})$, but k_{dev} is unknown!
- ▶ For a given input in we can compute the value y for all possible keys $k \in \{0, 1, \dots, 255\}$
- ▶ ForAll $in \in \mathbf{in}$
 ForAll $k \in \{0, 1, \dots, 255\}$
 Compute $y(in, k) = \text{Sbox}(in \oplus k)$

in1


in2

in3

in4

in5

in6



Value-prediction matrix

k=0	k=1		k=255
Sbox(in1 XOR 0)	Sbox(in1 XOR 1)	...	Sbox(in1 XOR 255)
Sbox(in2 XOR 0)	Sbox(in2 XOR 1)	...	Sbox(in2 XOR 255)
Sbox(in3 XOR 0)	Sbox(in3 XOR 1)	...	Sbox(in3 XOR 255)
Sbox(in4 XOR 0)	Sbox(in4 XOR 1)	...	Sbox(in4 XOR 255)
Sbox(in5 XOR 0)	Sbox(in5 XOR 1)	...	Sbox(in5 XOR 255)
Sbox(in6 XOR 0)	Sbox(in6 XOR 1)	...	Sbox(in6 XOR 255)

No inputs x No keys

- ▶ One of the columns of this value-prediction matrix is the correct one!
- ▶ The intermediate value also called the sensitive variable needs to be fairly small
- ▶ Divide and Conquer strategy

- We map the hypothetical intermediate values to hypothetical power consumption values, producing the power-prediction matrix

Power-prediction matrix

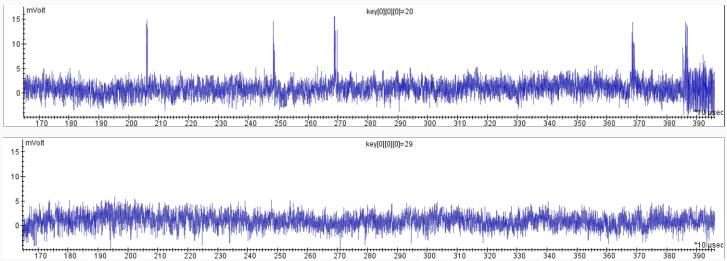
k=0	k=1		k=255
HW(Sbox(in1 XOR 0))	HW(Sbox(in1 XOR 1))	...	HW(Sbox(in1 XOR 255))
HW(Sbox(in2 XOR 0))	HW(Sbox(in2 XOR 1))	...	HW(Sbox(in2 XOR 255))
HW(Sbox(in3 XOR 0))	HW(Sbox(in3 XOR 1))	...	HW(Sbox(in3 XOR 255))
HW(Sbox(in4 XOR 0))	HW(Sbox(in4 XOR 1))	...	HW(Sbox(in4 XOR 255))
HW(Sbox(in5 XOR 0))	HW(Sbox(in5 XOR 1))	...	HW(Sbox(in5 XOR 255))
HW(Sbox(in6 XOR 0))	HW(Sbox(in6 XOR 1))	...	HW(Sbox(in6 XOR 255))

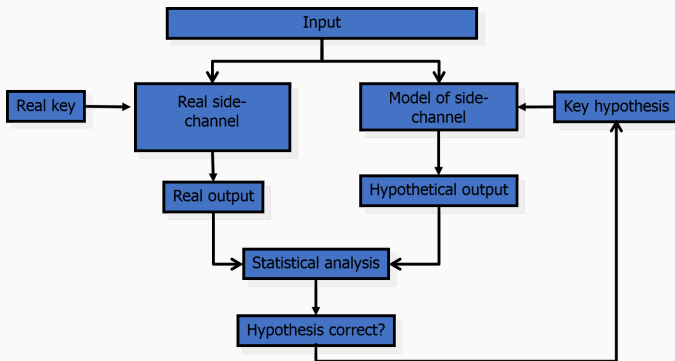
No_inputs x No_keys

- A common choice is Hamming weight but keep in mind that other models may be applicable

- ▶ Finally we compare the hypothetical power consumption values with the real measurements using Pearson correlation
- ▶ ForAll columns of measurement matrix
 - ForAll columns of power prediction matrix
 - Compute the correlation between columns
- ▶ The highest correlation value reveals the key!

CPA Step 5: Comparison





- ▶ Eric Brier, Christophe Clavier, Francis Olivier: Correlation Power Analysis with a Leakage Model. CHES 2004.
- ▶ Stefan Mangard, Elisabeth Oswald, Thomas Popp: Power analysis attacks – revealing the secrets of smart cards. Springer 2007.

Countermeasures

Purpose: break the link between (actual) intermediate computation values and power consumption

► Masking:

- A random mask concealing every intermediate value
- Can be on all levels (arithmetic → gate level)

► Hiding:

- Making power consumption independent of the intermediate values and of the operations
- Special logic styles, randomizing in time domain, lowering SNR ratio

- ▶ Time randomization:
 - the operations are randomly shifted in time
 - use of NOP operations
 - add random delays
 - use of dummy variables and instructions (sequence scrambling)
- ▶ Register renaming and nondeterministic processor
 - Idea is to exploit ILP within an instruction stream
 - Processor selects an instruction and a memory access randomly
- ▶ Permuted execution
 - rearranged instructions e.g. S-boxes
- ▶ Masking techniques

- ▶ Noise generation:
 - hardware noise generator requires RNG
 - total power is increased (problem for low-cost devices)
- ▶ Desynchronization:
 - Introducing some fake clock cycles during the computation or using a weak jitter
- ▶ Power signal filtering:
 - ex.: RLC filter (R-resistor, C-capacitor, L-inductor) smoothing the pow. cons. signal by removing high frequency components
 - one should use active comp. (transistors) in order to keep pow. cons. relatively constant - problem for mob. phones
- ▶ Novel circuit designs e.g. special logic styles

- ▶ A conventional first-order Boolean masking scheme splits the intermediate variable X into two shares i.e. two randomized variables X_1 and X_2 such that $X_1 \oplus X_2 = X$.
- ▶ The leakage $L(X) = HW(X_1, X_2)$ depends on two variables.
- ▶ It does not reveal any information on the value of X when a DPA is performed

x	x_1	x_2	$\mathcal{L}(x)$	$\text{Mean}(\mathcal{L}(x))$	$\text{Var}(\mathcal{L}(x))$
0	0	0	0	1	1
	1	1	2		
1	0	1	1	1	0
	1	0	1		