Smart Card Laboratory

Introduction to Side-Channel Analysis

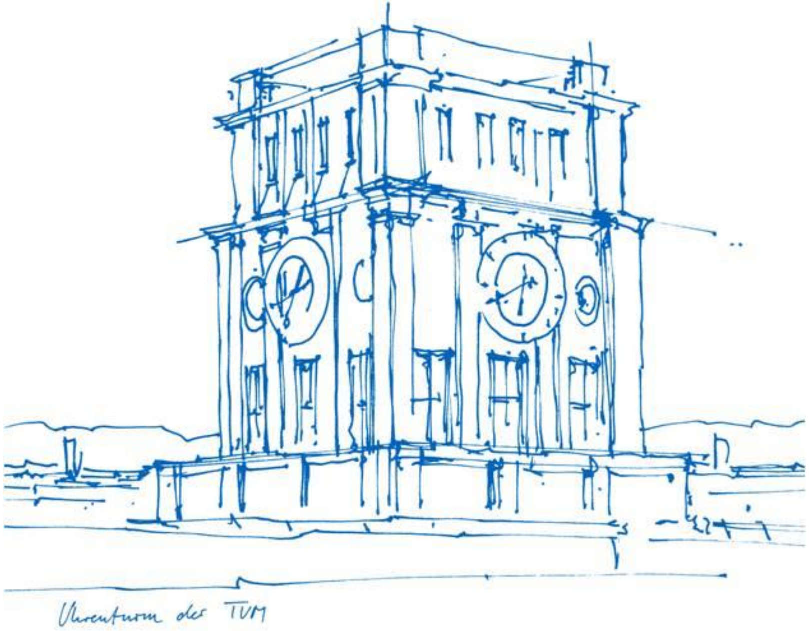
Tim Music

Technische Universität München

TUM School of Computation, Information and Technology

Department for Security in Information Technology

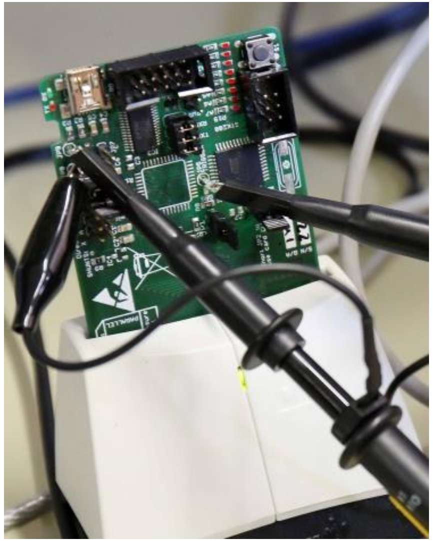
München, 31.10.2024



The Side-Channel Lecture in a Nutshell

What You will be Learning in today’s lecture

- Assumptions in Cryptographic Scenarios (Black, Grey, White Box Crypto)
- Classifications of attacks on embedded systems
- Summary of AES cipher internals
- Simple & Differential Power Analysis + Countermeasures



Recap: Smart Card Characteristics and Use Cases

Characteristics

Typical Use Cases

Hardware Components

Recap: Challenges

Implementation
Challenges

Outline

- Introduction
- Attacks on Embedded Systems
- The AES Cipher
- The Power Side-Channel, SPA & DPA
- Differential Power Analysis Countermeasures
- Brief Introduction into HDF5
- Hardware Handout

Introduction

(Cryptographic Assumptions)

Introduction

Black-, Gray- and White Box Assumptions are essential when discussing cryptographic scenarios, as they lay out the capabilities of the attacker.

Black Box Assumptions

Symmetric Cryptography

AES introduced 1999 is still cryptographically secure

Asymmetric Cryptography

RSA with 2048 Bit key will be secure for the next years

Elliptic curves cryptography with smaller key size and equivalent security as an alternative

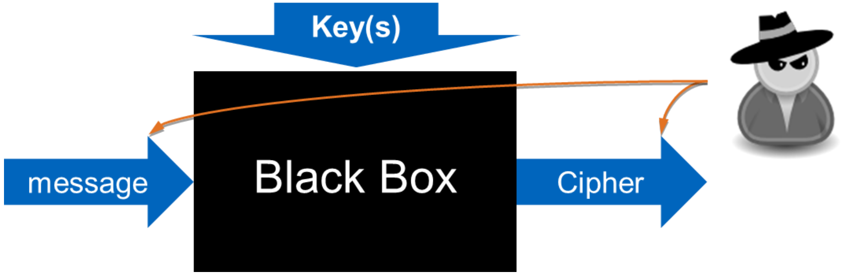
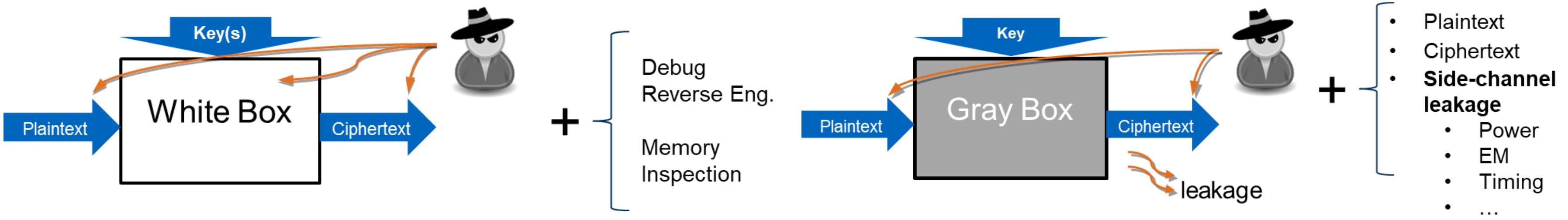


Image source: http://www.opensecurityarchitecture.org/cms/library/icon-library

Introduction

White- and Grey Box Assumptions



Debug
Reverse Eng.

Memory
Inspection

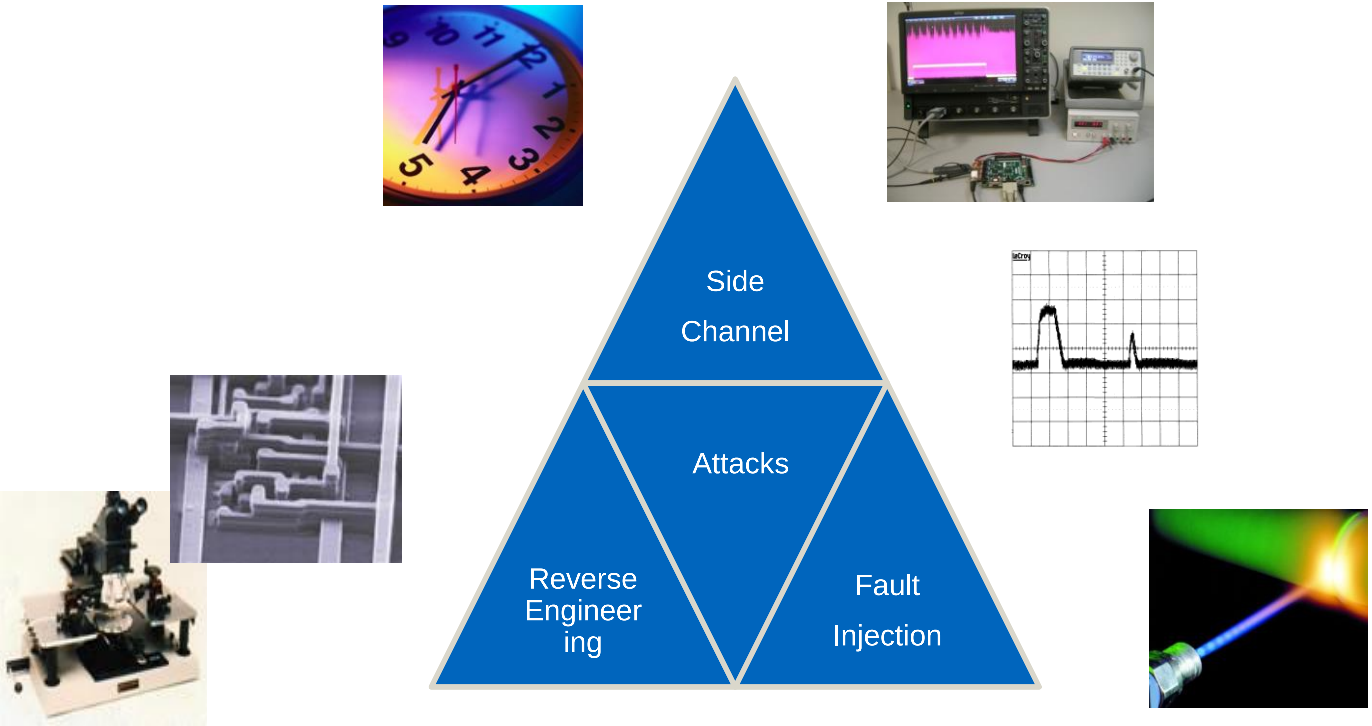
- Plaintext
- Ciphertext
- **Side-channel leakage**
  - Power
  - EM
  - Timing
  - …

For Implementations of cryptographic algorithms, the black box assumptions are no longer valid.

Attacks on Embedded Systems

(there are quite a few...)

Overview



Side Channel

Attacks

Reverse Engineering

Fault Injection

Reverse Engineering

Retrieving cipher internals through gate level reverse engineering

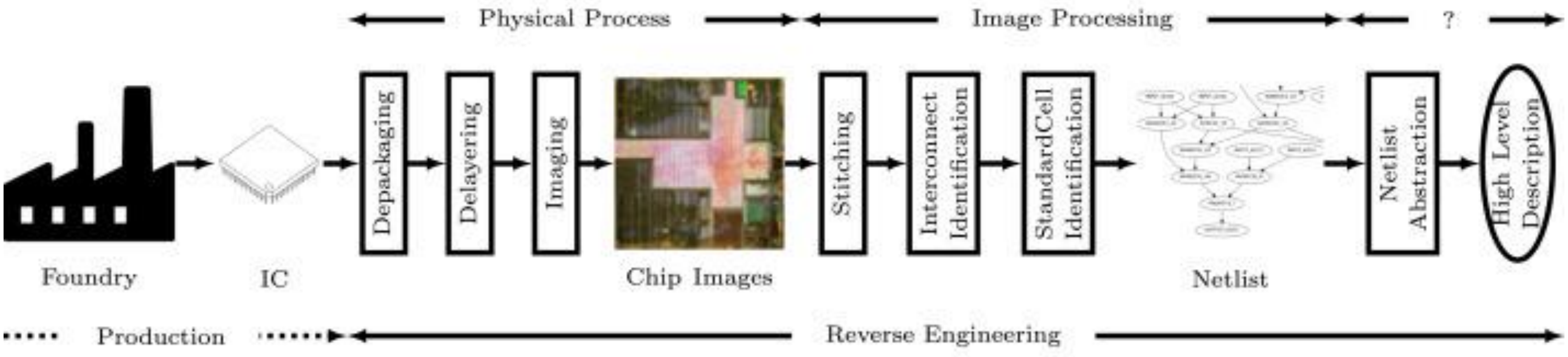


Image source: Baehr, J., et al. Machine learning and structural characteristics for reverse engineering

Attack Classification

	Active Attacks	Passive Attacks
Non-Invasive	Glitching, Temperature Change, Low Voltage, …	Side-Channel Attacks: Timing Analysis, Power Analysis, Simple EM Attacks
Semi-Invasive	Light Attacks, Radiation Attacks, …	Sophisticated EM Attacks, Optical inspection (ROM, …)
Invasive	Forcing, Permanent circuit changes	Probing Attacks, …

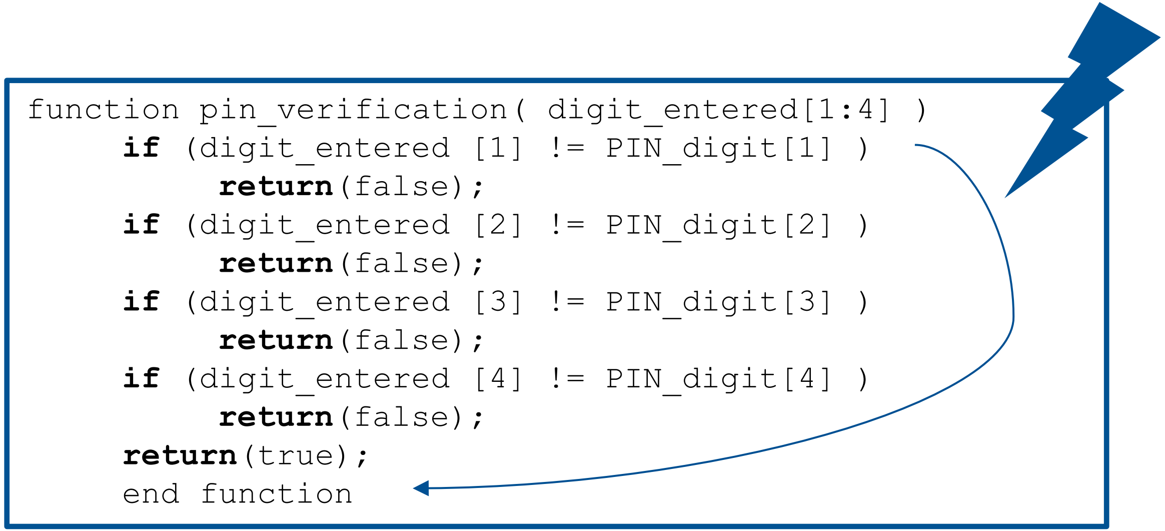
Side-Channel & Fault Attack Example Application

The User needs to enter a four-digit pin to unlock functionality.

```
function pin_verification( digit_entered[1:4] )
     if (digit_entered [1] != PIN_digit[1] )
          return(false);
     if (digit_entered [2] != PIN_digit[2] )
          return(false);
     if (digit_entered [3] != PIN_digit[3] )
          return(false);
     if (digit_entered [4] != PIN_digit[4] )
          return(false);
     return(true);
     end function
```

Fault Attack

Tampering of the program execution flow to jump instructions.

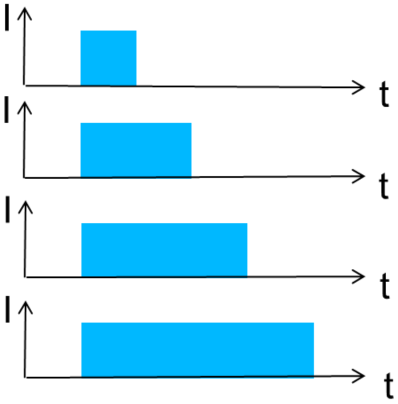


```
function pin_verification( digit_entered[1:4] )
     if (digit_entered [1] != PIN_digit[1] )
          return(false);
     if (digit_entered [2] != PIN_digit[2] )
          return(false);
     if (digit_entered [3] != PIN_digit[3] )
          return(false);
     if (digit_entered [4] != PIN_digit[4] )
          return(false);
     return(true);
     end function
```

Timing Side-Channel Attack

Measurement of execution time leaks information about the processed secret.

```
function pin_verification( digit_entered[1:4] )
     if (digit_entered [1] != PIN_digit[1] )
          return(false);
     if (digit_entered [2] != PIN_digit[2] )
          return(false);
     if (digit_entered [3] != PIN_digit[3] )
          return(false);
     if (digit_entered [4] != PIN_digit[4] )
          return(false);
     return(true);
     end function
```



Electromagnetic Analysis

Idea

- Measure electromagnetic emanation during crypto operations
- Repeat this many times (10^2 – 10^6)
- Perform statistical correlation between the traces measured and an emission model of the implementation

Pros

High resolution

No modifications to board are required

Localized information (for localized EM)

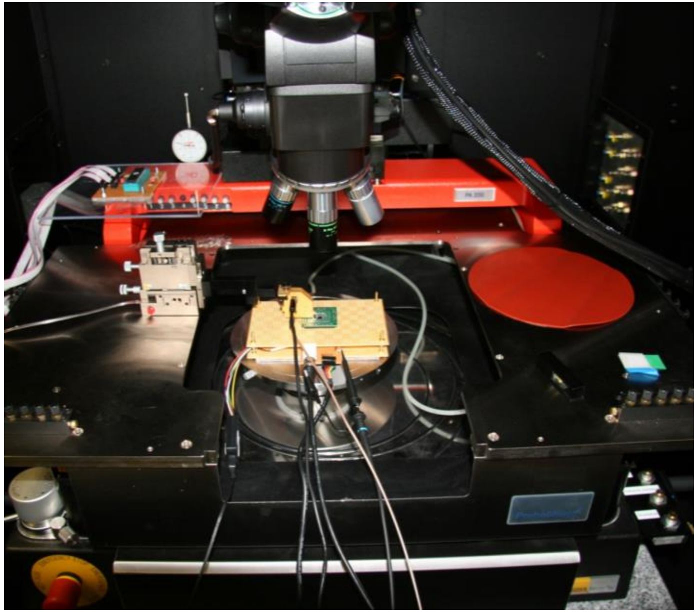
Cons

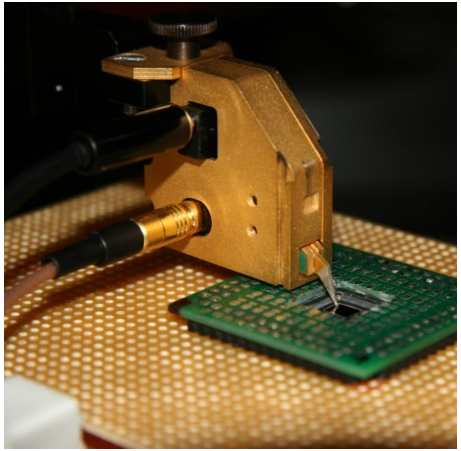
Expertise required

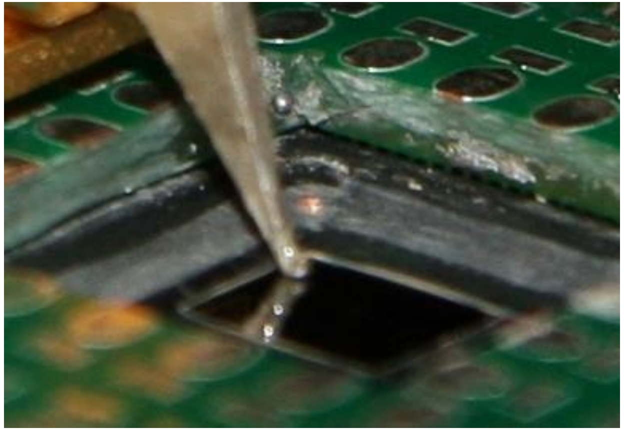
Chip de-capsulation (improves results drastically)

Image source: http://aktuell.ruhr-uni-bochum.de/mam/images/pi2011/em-messung_smartcard.jpg

Localized Electromagnetic Analysis







The AES Cipher

(the world’s most widely used one)

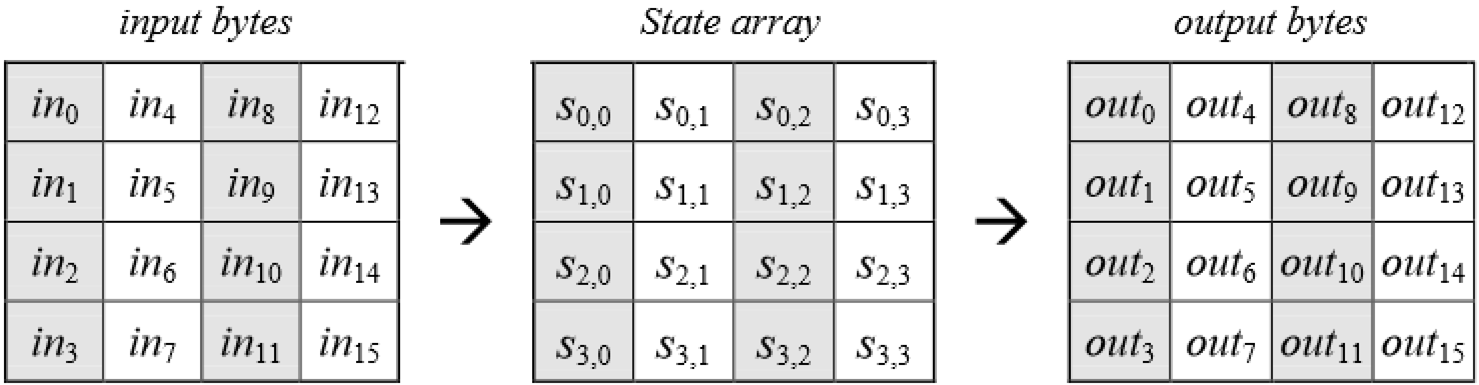
AES – History and Facts

- In 1997 NIST announced a competition to find a successor for the widely used Data Encryption Standard (DES) which was vulnerable to brute force attacks due to the short key
- There were in total fifteen designs submitted from several countries and publicly investigated by cryptographers
- In the year 2000, the winner was announced: Rijndael, created by Joan Daemen and Vincent Rijmen
- On the 26.11.2001, it was approved as FIPS PUB 197 and named the Advanced Encryption Standard (AES)
- AES has been designed to be efficient in software, as well as hardware and is the most widely used cipher in the world
- AES is a block cipher with 128 bits block size and three supported key lengths (128, 192 and 256 bits) with respectively 10, 12 or 14 rounds

The following AES overview is very coarse and aids the understanding of the topic in the lecture. Please make sure to also read the FIPS 197 specification.

AES State

The AES algorithm operations are performed on a two-dimensional array of bytes called the **state**



input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

→

State array

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

→

output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Image source: source: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

AES Structure

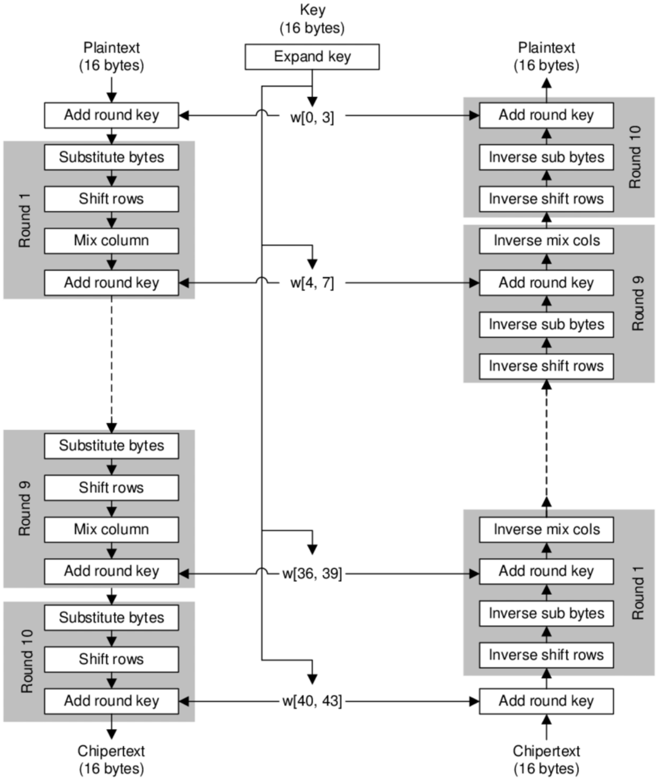


Image source: Ratnadewi, R., et al. (2017). Implementation and performance analysis of AES-128 cryptography method in an NFC-based communication system. World Transactions on Engineering and Technology Education. 15. 178-183.

AES Transformations

AddRoundKey Transformation

- The Round key is added to the state
- Simple bitwise XOR operation

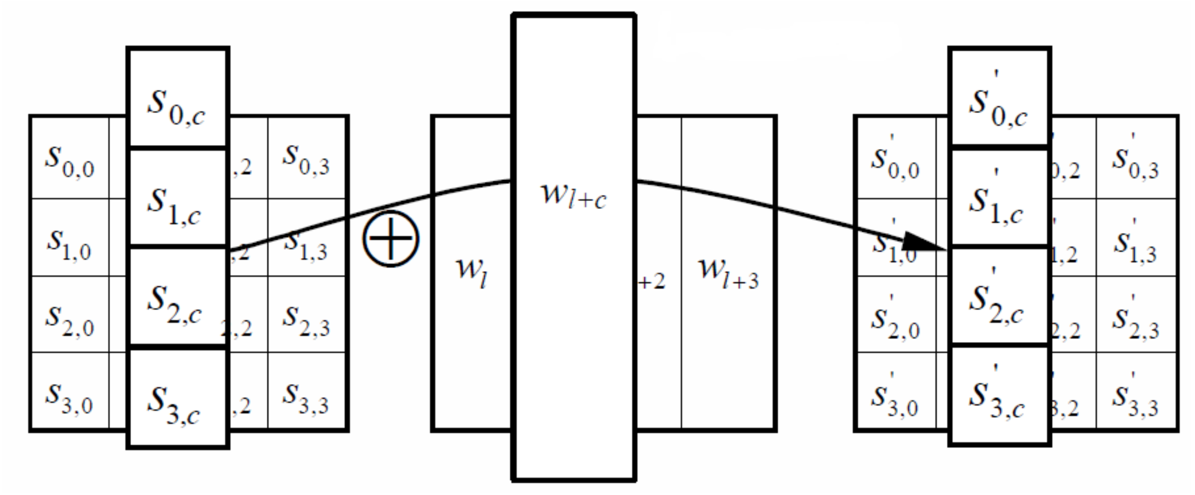
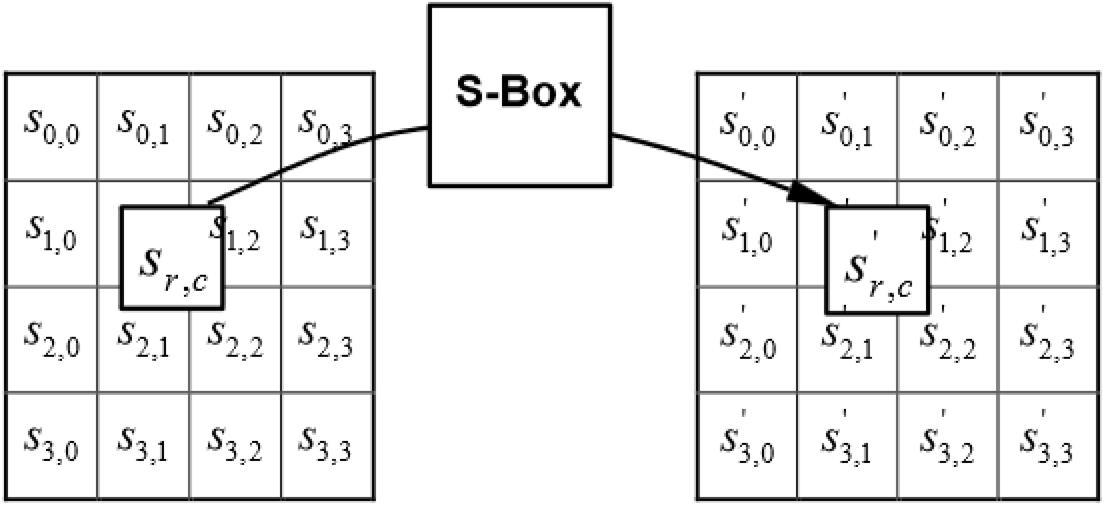


Image source: source: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

AES Transformations

SubBytes Transformation

- Non-linear byte substitution
- Operates on each byte using a substitution table



$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{r,c}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

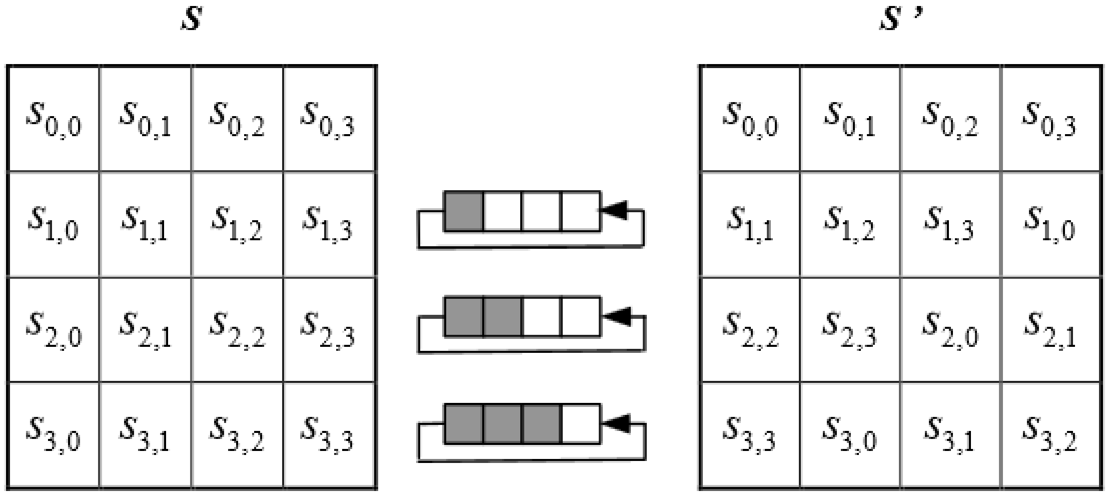
$s'_{0,0}$	$s'_{0,1}$	$s'_{0,2}$	$s'_{0,3}$
$s'_{1,0}$	$s'_{r,c}$	$s'_{1,2}$	$s'_{1,3}$
$s'_{2,0}$	$s'_{2,1}$	$s'_{2,2}$	$s'_{2,3}$
$s'_{3,0}$	$s'_{3,1}$	$s'_{3,2}$	$s'_{3,3}$

Image source: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

AES Transformations

ShiftRows Transformation

- Cyclical shift of rows
- Each row is cycled with a different offset



s

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

s '

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

Image source: source: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

AES Transformations

MixColumns Transformation

- Operates on the state one column at a time
- The columns are considered as polynomials over GF(2^8) and multiplied modulo x^4+1 with a fixed polynomial
- It can be seen as a matrix multiplication

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

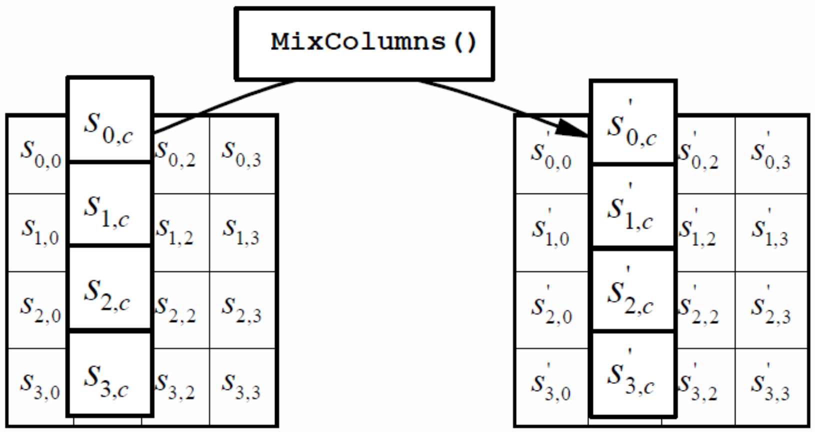

MixColumns()
$s_{0,0}$ $s_{0,c}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,c}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,c}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,c}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,c}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,c}$ $s'_{3,2}$ $s'_{3,3}$

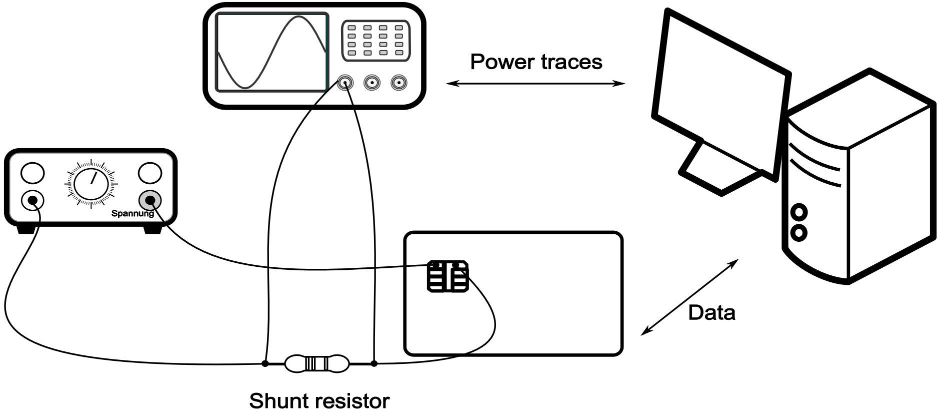
Image source: source: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

The Power Side-Channel

(and how to exploit it)

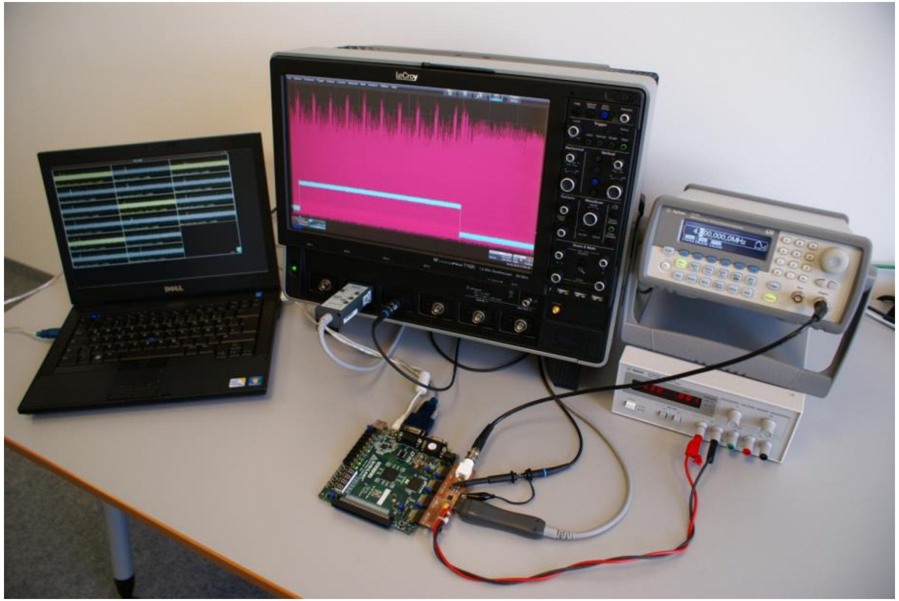
Measurement of the Power Side-Channel

- Smart card receives power from laboratory supply
- Measurement of the current consumed with a shunt resistor
- Interfacing towards the PC through a card reader

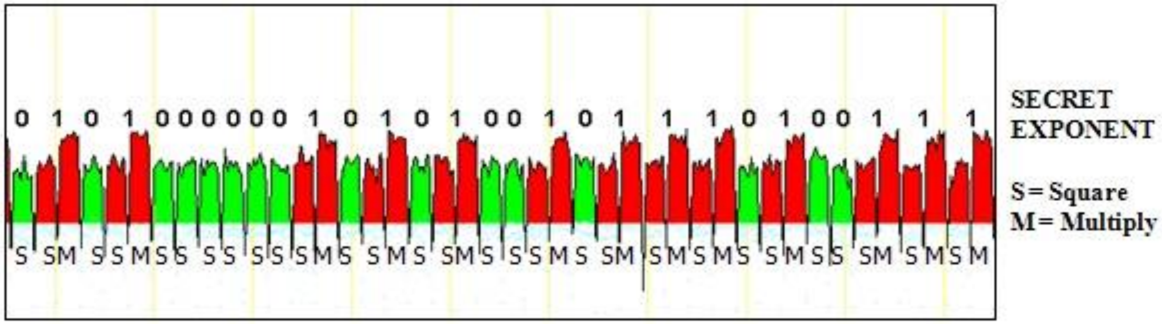


Measurement of the Power Side-Channel

- Smart card receives power from laboratory supply
- Measurement of the current consumed with a shunt resistor
- Interfacing towards the PC through a card reader



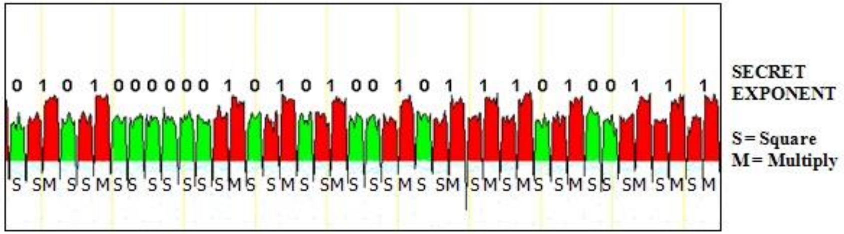
Simple Power Analysis



Simple Power Analysis

Idea

- Measure the power profile during a single crypto operation
- Power consumption is data dependent
- Extract information directly from the power trace
  e.g., square versus (square + multiply)



Pros

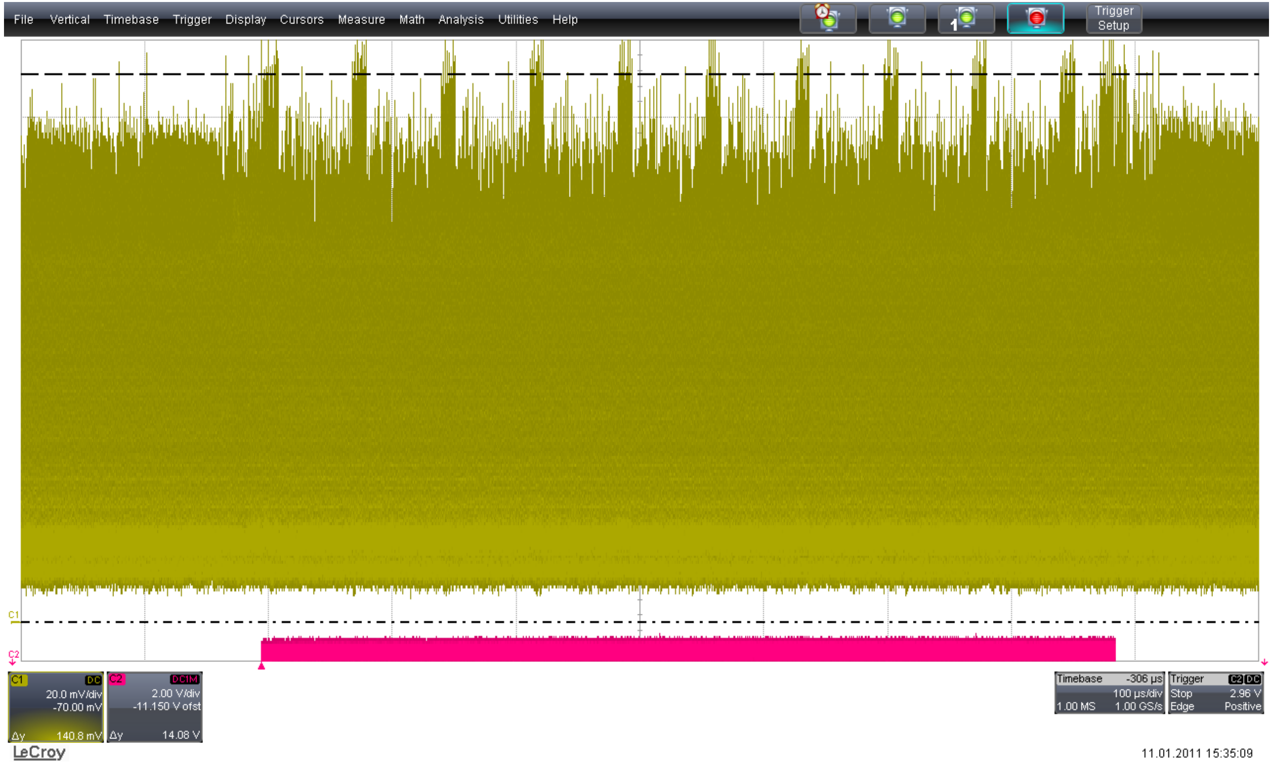
- Cheap equipment
- Low knowledge required

Cons

- Bad resolution: critical in case of low signal to noise ratio
- Modifications to the board are required
- Difficult in SoCs with a single power supply; no local information

image source: eetimes.com

Simple Power Analysis



C1 DC
20.0 mV/div
-70.00 mV
Δy 140.8 mV

C2 DC1M
2.00 V/div
-11.150 V ofst
Δy 14.08 V

Timebase -306 µs
100 µs/div
1.00 MS 1.00 GS/s

Trigger C2 DC
Stop 2.96 V
Edge Positive

11.01.2011 15:35:09

Simple Power Analysis

Observations from single power trace by visual inspection

- The parameters of the algorithm (e.g., # rounds in AES ➞ key length)
- A closer look may identify characteristics of single instructions
- Length differences of instructions (e.g., MUL, DIV) may help make educated guesses about the parameters
- Differences in the instruction flow (key dependent jumps) may allow extracting secret data
- Memory accesses often show characteristic power profiles
- Cache hit and miss behavior can be easily identified

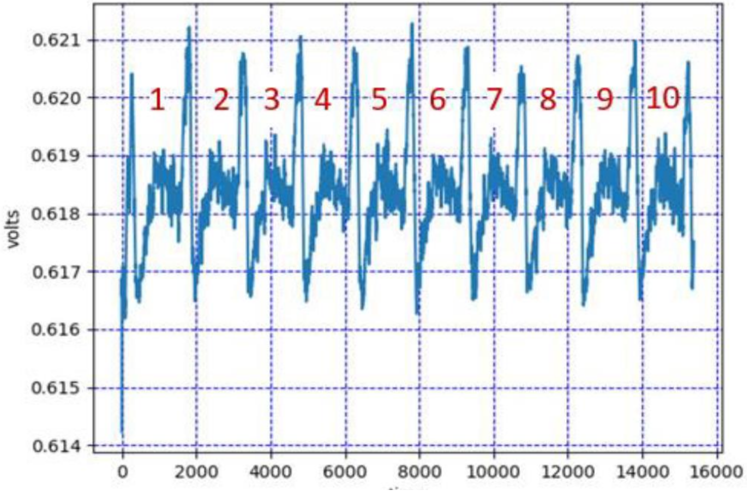


image source: Randolph, Mark & Diehl, William. (2020). Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. Cryptography. 4. 15. 10.3390/cryptography4020015.

Simple Power Analysis

Observations from single power trace by visual inspection

- The parameters of the algorithm (e.g., # rounds in AES ➞ key length)
- A closer look may identify characteristics of single instructions
- Length differences of instructions (e.g., MUL, DIV) may help make educated guesses about the parameters
- Differences in the instruction flow (key dependent jumps) may allow extracting secret data
- Memory accesses often show characteristic power profiles
- Cache hit and miss behavior can be easily identified

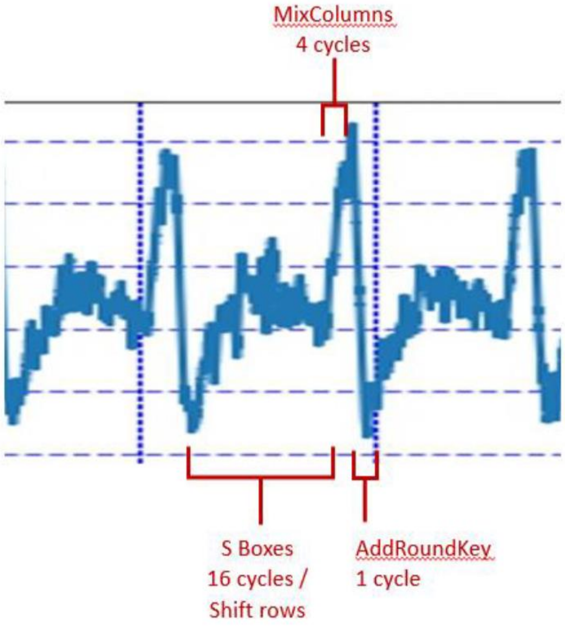


image source: Randolph, Mark & Diehl, William. (2020). Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. Cryptography. 4. 15. 10.3390/cryptography4020015.

Differential Power Analysis

What is a DPA?

- Power analysis attack which goal is to reveal the secret keys of a cryptographic device
- It assumes that the power consumption is a function of the secret data being processed
- The shape of the trace is not important (as in SPA)
- Uses a large number of power traces taken during the operation of the device
- Linear data relationships are determined by using statistical tools

Differential Power Analysis

History

- DPA was published by Paul Kocher, Joshua Jaffe, Benjamin Jun in Proceedings of Crypto 1999
- Paul Kocher founded a company Cryptography Research Inc. CRI, which holds most of the patents for countermeasures against SCA (including DPA)
- On June 6, 2011, Cryptography Research was bought by Rambus in a deal worth $342.5 million
- For further information and nice videos about DPA see: http://www.cryptography.com/

Differential Power Analysis

Attack Strategy

Choosing an intermediate result of the executed algorithm

Measuring the power consumption

Calculating intermediate values

Mapping intermediate values to hypothetical power consumption values

Comparing the hypothetical power consumption values with the power consumption traces

DPA Step 1 – Viable Attack Points in AES

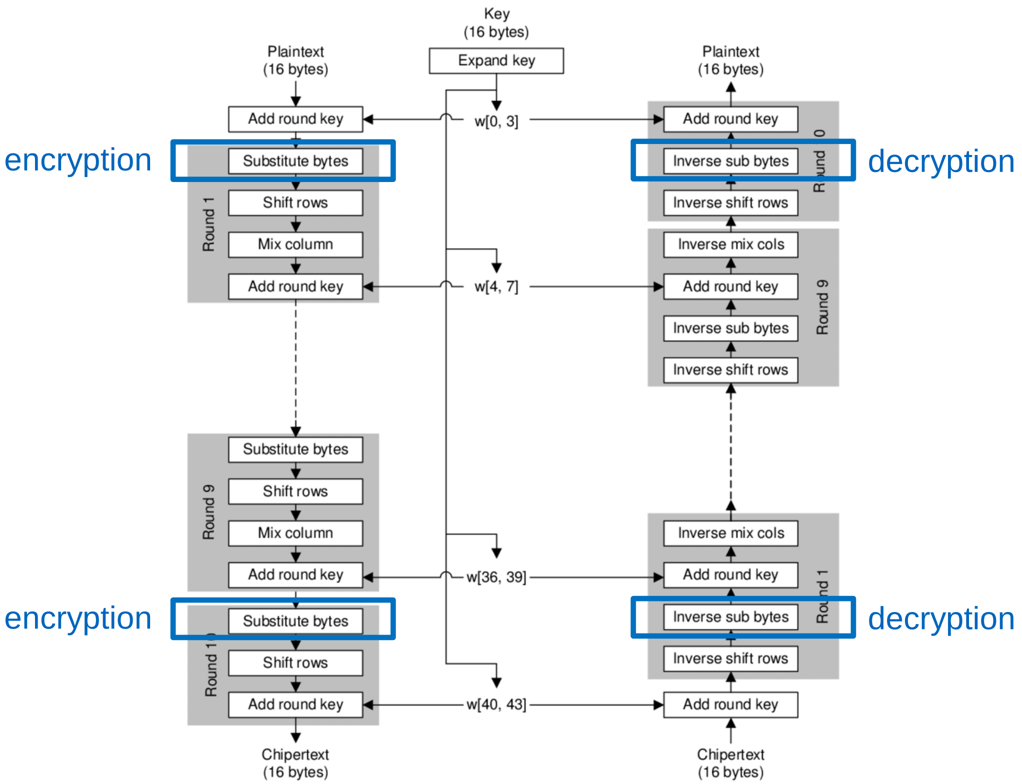


Image source: Ratnadewi, R., et al. (2017). Implementation and performance analysis of AES-128 cryptography method in an NFC-based communication system. World Transactions on Engineering and Technology Education. 15. 178-183.

DPA – Step 1

Choose an intermediate result of the algorithm which depends on data (known) and the key (unknown): f(d,k)

Most effective attacks on AES can be mounted at the

- S-box output of the first round (for encryption)
- S-box input of the last round (for decryption)

Reason

- Number of key hypothesis for the S-box is small (i.e., 2^8)
- The intermediate value depends on small part of the key k and known data d
- Non-linear elements of the S-box make the DPA more effective
- (i.e., a one-bit difference at the input of an S-box leads to a difference of several bits at the output)

DPA Step 2 – Power Consumption Measurement

- Choose a set of D data values which will be encrypted (or decrypted). Store this values as a vector: d = (d_1,…d_D)’

- Measure the power consumption for each encryption of a data value d_i and store it in a power trace t'_i = ($t_{i,1}$, …$t_{i,T}$)’ with T samples

- Store all collected traces inside a matrix T of dimensions DxT.
  - It is important to align the traces
  - All the values in a column of T have to belong to the same operation
  - Using a unique trigger for trace collection with the oscilloscope yields the best results

$$T = \begin{bmatrix} t_{1,1} & \cdots & t_{1,j} & \cdots & t_{1,T} \\ \vdots & \ddots & & & \vdots \\ t_{i,1} & & t_{i,j} & & t_{i,T} \\ \vdots & & & \ddots & \vdots \\ t_{D,1} & \cdots & t_{D,j} & \cdots & t_{D,T} \end{bmatrix}$$

DPA Step 2 – Power Consumption Measurement

Trace Compression (Optional)

Concept

- Power traces contain many points with lots of redundancy
- In order to speed up analysis it is favorable to reduce the number of points
- Such techniques receive the name of trace compression

There are many methods to compress a trace, e.g.:

- Sum of absolute values over a time interval
- Sum of squared values over a time interval
- Maximum value in a time interval

DPA Step 3 – Calculate Intermediate Values

List all possible key values and store it in a vector of size K (e.g., for the S-box K = 256 possible values since each key part is 8-bit long)
k = (k_1,…, k_K)’ (e.g., **k** = (0,1,2,… ,255)’ for an 8-bit value)

Calculate a matrix **V** of possible intermediate results for all data and key hypothesis with elements $v_{i,j}$ = *f* (d_i, k_j) using the function which was chosen in the first step

$$V = \begin{bmatrix} v_{1,1} & \cdots & v_{1,j} & \cdots & v_{1,K} \\ \vdots & \ddots & & & \vdots \\ v_{i,1} & & v_{i,j} & & v_{i,K} \\ \vdots & & & \ddots & \vdots \\ v_{D,1} & \cdots & v_{D,j} & \cdots & v_{D,K} \end{bmatrix}$$

DPA Step 4 – Hypothetical Power Consumption Values

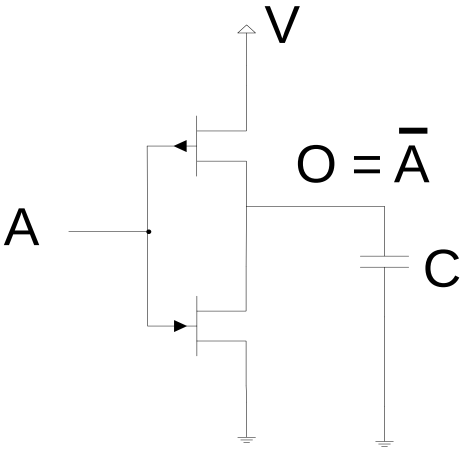
Wait, but how do we know how much power the device draws?

A brief excurse into CMOS Models…

CMOS Power Consumption Models

Hamming Distance

- Power consumption proportional to the number of transitions from 0 to 1 and 1 to 0
- Assumptions
  - Transitions from 0 to 1 and 1 to 0 consume the same power
  - Transitions from 0 to 0 and 1 to 1 do not consume power



$P = C{*}V^2$ if $O(t-1)=0$ and $O(t)=1$

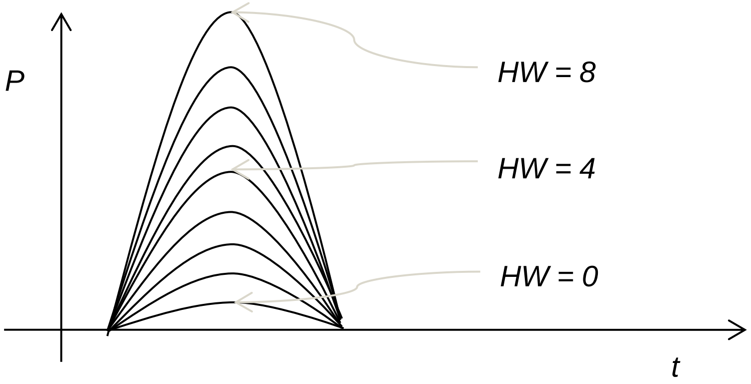
$P = 0$ if $O(t-1) = O(t)$ or
$O(t-1) = 1$ and $O(t)=0$

Hamming Distance = $|A(t) – A(t-1)|$

CMOS Power Consumption Models

Hamming Weight

- Most commonly used
- Simpler than the Hamming Distance model
- Used when the attacker only knows one data value being transferred (i.e. no information about the previous value)
- Assumption
  - Power consumption is proportional to the number of bits that are set in the processed value



DPA Step 4 – Hypothetical Power Consumption Values

- Calculate the power consumption for each element in the matrix **V** of possible intermediate results
- Make use of a power model to estimate the power consumption
  - Hamming weight
  - Hamming Distance
- This results in a matrix **H** of hypothetical power consumption

$$H = \begin{bmatrix} h_{1,1} & \cdots & h_{1,j} & \cdots & h_{1,K} \\ \vdots & \ddots & & & \vdots \\ h_{i,1} & & h_{i,j} & & h_{i,K} \\ \vdots & & & \ddots & \vdots \\ h_{D,1} & \cdots & h_{D,j} & \cdots & h_{D,K} \end{bmatrix}$$

This matrix contains a column with j = ck (correct key hypothesis) where all power values will be correlated with a power value in the measurements.

DPA Step 5 – Correlate Power Consumption Values

Note

In this lecture we are not performing the traditional DPA, but a further advancement of it, the correlation based power analysis, short CPA.

Advantages of CPA in comparison to classical DPA

- Allows for more complex power models (classical DPA is limited to binary models, i.e., attacking a single bit)
- When a trend can be seen in correlation values, the total number of traces required may be estimated

DPA Step 5 – Correlate Power Consumption Values

In this step, the correlation coefficient between the columns of the matrix of measured traces **T** and the columns of the matrix of hypothetical values **H** is calculated for all points in time. The estimated correlation coefficient $r_{i,j}$ is calculated by taking column *i* from matrix **H** and column *j* from matrix **T**.

The result is a KxT matrix of correlation coefficients **R**.

$$r_{i,j} = \frac{\sum_{d=1}^{D}[(t_{d,j} - \bar{t}_j) \cdot (h_{d,i} - \bar{h}_i)]}{\sqrt{\sum_{d=1}^{D}(t_{d,j} - \bar{t}_j)^2 \cdot \sum_{d=1}^{D}(h_{d,i} - \bar{h}_i)^2}}$$

$$R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,j} & \cdots & r_{1,T} \\ \vdots & \ddots & & & \vdots \\ r_{i,1} & & r_{i,j} & & r_{i,T} \\ \vdots & & & \ddots & \vdots \\ r_{K,1} & \cdots & r_{K,j} & \cdots & r_{K,T} \end{bmatrix}$$

DPA Step 5 – Correlate Power Consumption Values

Finding the most probable result

The key byte and the time when it is used in the power trace can be obtained by finding

$$r_{ck,ct} = \max_{i,j}(abs(r_{i,j})); \qquad ck = i; \qquad ct = j$$

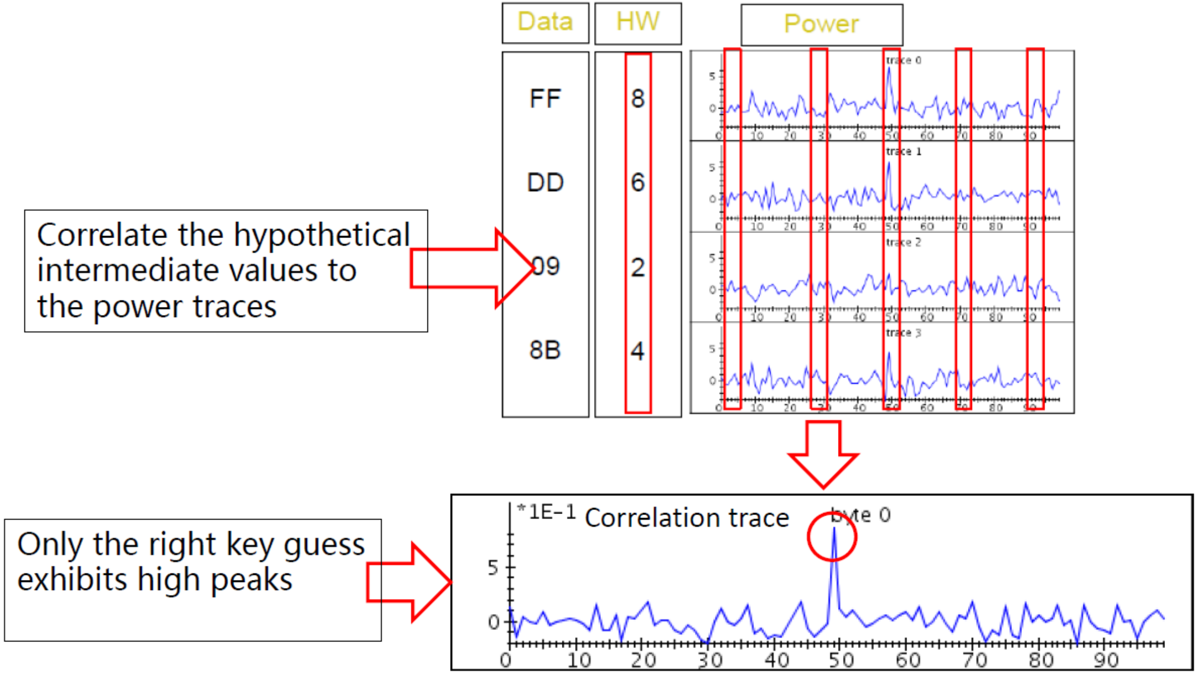
ck is the index of the row with correct key
ct is the index of the column with the correct time

$$R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,j} & \cdots & r_{1,T} \\ \vdots & \ddots & & & \vdots \\ r_{i,1} & & \enclose{circle}{r_{i,j}} & & r_{i,T} \\ \vdots & & & \ddots & \vdots \\ r_{K,1} & \cdots & r_{K,j} & \cdots & r_{K,T} \end{bmatrix}$$

key with maximum correlation

time of maximum correlation

DPA Step 5 – Testing Keys Bytes for Correctness



Data	HW
FF	8
DD	6
09	2
8B	4

Correlate the hypothetical intermediate values to the power traces

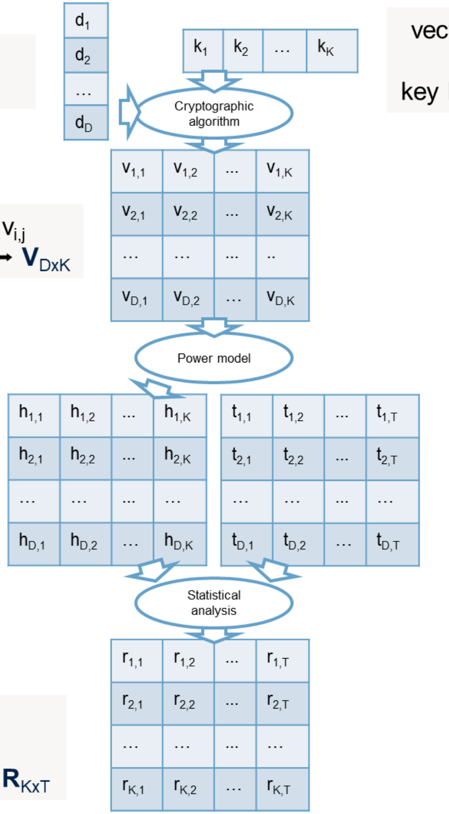
Only the right key guess exhibits high peaks

Image source: Cloning 3G/4G SIM Cards with a PC and an Oscilloscope: Lessons Learned in Physical Security – Yu Yu – Black Hat 2015

Graphical DPA Summary

vector **d** with D plaintext/ciphertexts

vector **k**' with K key hypothesis



$$\begin{matrix} d_1 \\ d_2 \\ \dots \\ d_D \end{matrix}$$

$$\begin{matrix} k_1 & k_2 & \dots & k_K \end{matrix}$$

intermediate values $v_{i,j}$ with all key hypothesis ➞ $\mathbf{V}_{DxK}$

$$\begin{matrix} v_{1,1} & v_{1,2} & \dots & v_{1,K} \\ v_{2,1} & v_{2,2} & \dots & v_{2,K} \\ \dots & \dots & \dots & .. \\ v_{D,1} & v_{D,2} & \dots & v_{D,K} \end{matrix}$$

hypothetical power values $h_{i,j}$➞ $\mathbf{H}_{DxK}$

$$\begin{matrix} h_{1,1} & h_{1,2} & \dots & h_{1,K} \\ h_{2,1} & h_{2,2} & \dots & h_{2,K} \\ \dots & \dots & \dots & \dots \\ h_{D,1} & h_{D,2} & \dots & h_{D,K} \end{matrix} \qquad \begin{matrix} t_{1,1} & t_{1,2} & \dots & t_{1,T} \\ t_{2,1} & t_{2,2} & \dots & t_{2,T} \\ \dots & \dots & \dots & \dots \\ t_{D,1} & t_{D,2} & \dots & t_{D,T} \end{matrix}$$

Matrix of correlation coefficients ➞ $\mathbf{R}_{KxT}$

$$\begin{matrix} r_{1,1} & r_{1,2} & \dots & r_{1,T} \\ r_{2,1} & r_{2,2} & \dots & r_{2,T} \\ \dots & \dots & \dots & \dots \\ r_{K,1} & r_{K,2} & \dots & r_{K,T} \end{matrix}$$

Differential Power Analysis Countermeasures

(how to secure your implementation)

DPA Countermeasures

Hiding the power consumption

Masking intermediate values

Hiding the Power Consumption

Increasing the noise

- Time domain
  - Insertion of random wait states
  - Shuffling instructions, memory accesses, etc…
- Amplitude domain
  - Perform dummy operations in parallel to crypto algorithm
  - Dedicated noise generators on smartcards

Attenuation of the side-channel signal

- Amplitude domain
  - Special circuit design styles to achieve logic value independent power consumption
  - Filtering the power supply

Hiding Techniques in Software

Randomization

- Random wait state insertion (nop’s)
- Randomize instruction execution
- Randomize memory accesses

Generate Noise

- Make use of peripherals which generate noise in parallel to the cryptographic function

Masking Internal Values

Masking

- Make the power consumption of the device independent of the intermediate values of the cryptographic algorithm through randomization
- Avoids having to modify the power consumption characteristics of the device
- Can be implemented at the algorithm level

Concept

- Each intermediate value *v* is concealed by a random value *m*
- The value *m* varies in each execution and cannot be predicted

$$v_m = v * m$$

* = Masking operation $\oplus$, +, $\cdot$

Masking Types

Types

- Boolean masking (with xor $\oplus$): $v_m = v \oplus m$
  - Mostly applied in symmetric cryptography i.e. AES
- Arithmetic masking (with + or $\cdot$): $v_m = v + m$ or $v_m = v \cdot m$
  - Mostly applied in asymmetric cryptography

Masking of linear functions

- $f(v_m) = f(v) * f(m)$

Masking of nonlinear functions is difficult

- $f(v_m) \neq f(v) * f(m)$

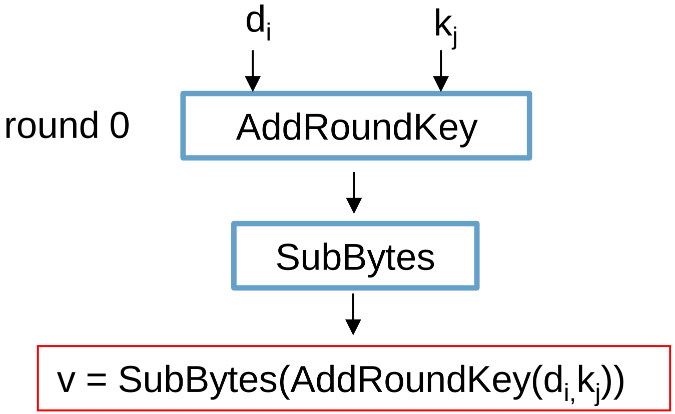
  For example, the AES S-box is non-linear:
  $S(v \oplus m) \neq S(v) \oplus S(m)$

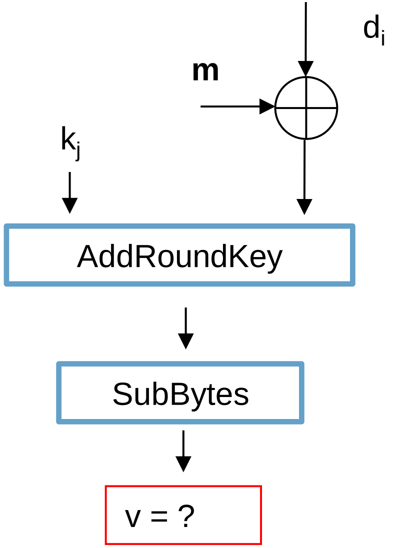
Masking Example

- Masking is the most widely used countermeasure in software
- In hardware masking can be implemented on any design level

What is the impact on masking with an **unknown mask m** on SCA?

The intermediate values are concealed





Masking AES in Software

Simple Masking Scheme for AES

- Data (or key) are XOR‘ed with a random mask at the beginning
- During each round the transformations are applied to the masked data and the mask itself
  - AddRoundKey: Apply the function only to the masked data
  - SubBytes: Need for a masked S-box table
  - Shift Rows: Shifting is applied to mask and data separately (unless all bytes in the state are masked with the same value, in that case it does not affect the masking)
  - MixColumns: Applied to mask and data separately
- After the last round, the mask is removed

Masking AES in Software

For the SW implementation 6 independent masks are used

- 2 masks *m* and *m'* for the S-box input and output
- 4 masks for each row of the state m_1,m_2,m_3,m_4 for the input of MixColumns

Pre-computation (done for every encryption)

- Generate the six masks at random
- Calculate the masked S-box, $S_m(x \oplus m) = S(x) \oplus m'$
- Execute MixColumns for the masks m_1,m_2,m_3,m_4 to create the output masks m'_1,m'_2,m'_3,m'_4

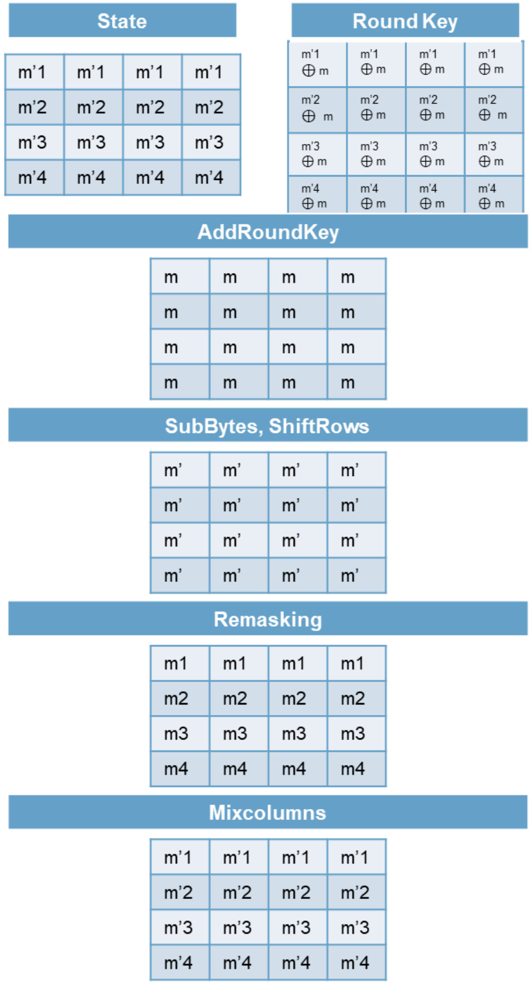
Execute the AES algorithm applying the masks on the state and round key

Masking AES in Software

Round Execution

- Mask the plaintext with [m'_1,m'_2,m'_3,m'_4]
- Mask the round key with [m'_1 *xor m*, m'_2 *xor m*, m'_3 *xor m*,m'_4 *xor m*]
- Perform AddRoundKey masks will become [*m, m, m, m*]
- Perform SubBytes and ShiftRows masks will become [*m’, m’, m’, m’*]
- Do re-masking to end up with the input masks for MixedColumns [m_1,m_2,m_3,m_4], masks after MixedColumns [m'_1,m'_2,m'_3,m'_4]

In the final round skip re-masking and MixedColumns
Then remove the masks from the ciphertext



State

m’1	m’1	m’1	m’1
m’2	m’2	m’2	m’2
m’3	m’3	m’3	m’3
m’4	m’4	m’4	m’4

Round Key

m’1 ⊕ m	m’1 ⊕ m	m’1 ⊕ m	m’1 ⊕ m
m’2 ⊕ m	m’2 ⊕ m	m’2 ⊕ m	m’2 ⊕ m
m’3 ⊕ m	m’3 ⊕ m	m’3 ⊕ m	m’3 ⊕ m
m’4 ⊕ m	m’4 ⊕ m	m’4 ⊕ m	m’4 ⊕ m

AddRoundKey

m	m	m	m
m	m	m	m
m	m	m	m
m	m	m	m

SubBytes, ShiftRows

m’	m’	m’	m’
m’	m’	m’	m’
m’	m’	m’	m’
m’	m’	m’	m’

Remasking

m1	m1	m1	m1
m2	m2	m2	m2
m3	m3	m3	m3
m4	m4	m4	m4

Mixcolumns

m’1	m’1	m’1	m’1
m’2	m’2	m’2	m’2
m’3	m’3	m’3	m’3
m’4	m’4	m’4	m’4

Attacks on Masking

Masking **protects against DPA** if

- There is no joint power consumption of the masked value and mask
- The mask is uniformly distributed

Implementation Pitfalls

- If masks are not changed frequently enough, DPA is still possible
- Masks may be biased due to insufficient statistical properties of the PRNG generating the masks
- Binning of the traces may be possible if (global) mask changes can be detected
- If masks are reused, operations with values (u, v), which are masked with the same mask m, may show the plain values
  $(u \oplus m) \oplus (v \oplus m) = u \oplus v$
- Hamming Distance of a register may leak the value being protected
  $HD(v_m, m) = HW(v_m \oplus m) = HW(v)$

Further Information

The book from Stefan Mangard, Thomas Popp, Elisabeth Oswald provides all necessary know how and detailed mathematical background to perform power attacks.
http://www.dpabook.org/

SICA lecture at TUM every winter term,
provides very good training in this topic. (taught in German)
Stefan Mangard’s material (2012) can be found under:
http://www.physical-security.org

Power analysis attacks are in the focus of the research community. Therefore new attack flavors are published every year.
www.chesworkshop.org
https://www.cosade.org/

HDF5 Introduction

(used for the measurements)

HDF5 Format

Hierarchical Data Format 5

- File format designed to store and organize large amounts of data
- BSD-like license (minimal restrictions)
- Official support for C/C++, Fortran, Java.
- Third party support for Python, Matlab, R, Perl, LabView, Julia, etc…
- Access to resources in a POSIX-like style
  - /path/to/resource

Object type:

- Datasets: Multidimensional arrays of a homogeneous type
- Groups: Container structures, they can contain:
  - Datasets
  - Other groups
- Metadata: User-defined, named attributes. May be attached to datasets and groups

HDF Chunks

Higher dimension illusion: Data in a disk is stored linearly

$$\begin{bmatrix} A & B & C \\ D & F & G \end{bmatrix} = \begin{cases} [A\ B\ C\ D\ F\ G] \text{ – } \textit{row-major} \text{ ordering} \\ [A\ D\ B\ F\ C\ G] \text{ – } \textit{column-major} \text{ ordering} \end{cases}$$

Locality: memory reads from a disk are generally faster when the data being accessed is all stored together.

$\boxed{[A\ D}\ B\ F\ C\ G]$

$\boxed{[A}\ B\ C\ \boxed{D}\ F\ G]$

Chunking let’s you specify the n-dimentional shape that best fits your access pattern.

HDF5 Cheat Sheet

Using Python’s h5py

Opening a file (read):

f = h5py.File("name.h5", "r")

Listing the group members (keys)

f.keys()

Displaying all the attributes of the root object

f.values()

Creating a group (file must be open as append or write)

group = f.create_group("/some/long/path")

Creating a dataset

data = f.create_dataset(“dataset1", (10, 10))

Reading a previously generated dataset

data = f[“ciphertext”]

Closing a file:

f.close()

References:

- https://opac.ub.tum.de/search?bvnr=BV041778278
- http://docs.h5py.org/en/latest/

Using Matlab

Importing data from a file

data = h5read(filename, datasetname)

data = h5read(‘name.h5’,’/dataset1’)

Creating a dataset

h5create(filename,datasetname,size,Name,Value)

Writing data to a file

h5write(filename, datasetname, data)

h5create(‘name.h5','/dataset1',[10 20])

Reference:

- *http://de.mathworks.com/help/matlab/high-level-functions.html*

Thank you for your Attention

(any questions so far?)