Advance topics in Hardware Security from Theory to Practice 049017

Prof. Avi Mendelson –
avi.mednelson@technion.ac.il

Lecture 7 – Side channel – II and Fault attack.

Agenda

- Rehash
- Correlation power analysis
- Template attacks
- Fault attack
- Exercise #2

Types of attacks

- White Box – the internal structure of the target is know; e.g., the algorithm, hardware structure
  - SPA – simple power attack
  - Timing attacks
- Black Box – only external observations are known, but no information regarding the internal information can be taken into account
  - DPA – differential power attack
- Grey Box – Some information is known
  - Pattern attack

Statistical behavior - cont

- In Figure 5 the assumption is that the power consumption when LSB=0 is different than the power consumption when LSB=1
- The graph shows the traces where the LSB=1 (M=16.9, SD=10.7) and where LSB=0 (M=121.9 SD=9.7).
- To aluminate the need for information about the target device, we averaging the measurements over time (offsets) within the traces.
- The basic DPA process examines the difference of these averages at each point in the set of traces

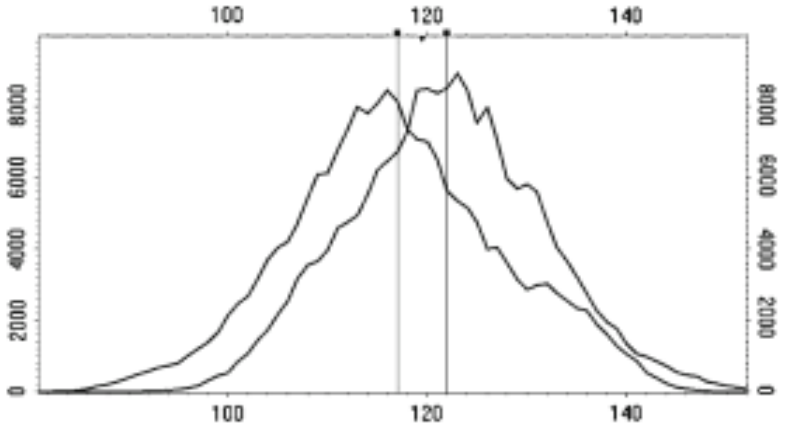


Fig. 5 Distributions of power consumption measurements for traces with the LSB of the output of the first S-box being 1 (left) and 0 (right)

DPA – Cont.

Four traces are shown;

1. The average of the traces: LSB = 1
2. The average of the traces: LSB = 0
3. the difference of the top two traces – entire trace
4. The difference of the averages when focused on the “focused area” (*Y* axis scaling increased by a factor of 15.

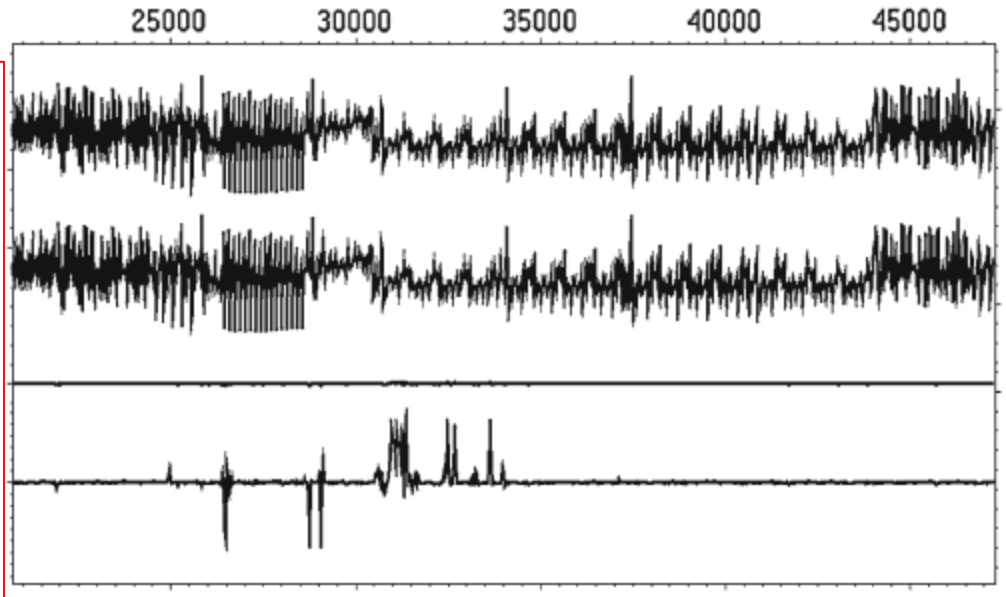


Fig. 6 Typical DPA result showing (from top to bottom) the average of the traces where the LSB of the output of first S-box in round 1 is 1, the average of traces where the LSB is 0, the difference between the top two traces, and the difference with the *Y* axis magnified by a factor of 15

DPA Mathematically

- Attacker now computes a k-sample differential trace $\Delta_D[1..k]$ by finding the difference between the average of the traces for which D(…) is one and the average for which D(…) is zero.

$$\Delta_D = \frac{\sum_{i=1}^{m} D(C_i, b, K_s) T_i[j]}{\sum_{i=1}^{m} D(C_i, b, K_s)} - \frac{\sum_{i=1}^{m} (1 - D(C_i, b, K_s)) T_i[j]}{\sum_{i=1}^{m} (1 - D(C_i, b, K_s))}$$

Principle: If K_s is wrongly guessed, D behaves like a random guess. Thus for a large number of sample points, ΔD[1..k] tends to zero. But if its correct, the differential will be non-zero and show spikes when D is correlated with the value being processed.

CPA - Correlation Power Analysis

- Correlation Power Analysis (CPA) is a statistical type of attack that uses the Pearson correlation coefficient to correlate data.
- CPA main advantage is that it requires less number of power trace
- The target is usually an intermediate value generated during the cryptographic operation (encryption or decryption) which is a function of a variable data value and part of the key
- $I = f(d; k)$
  - I is the intermediate value.
  - d is the relevant part of the plain text or cipher text
  - k is he relevant part of the key
  - f is the transportation function
- We assume N traces (e.g., N different plain texts)
- Each trace has M number of sampling points

CPA – Correlated power analysis

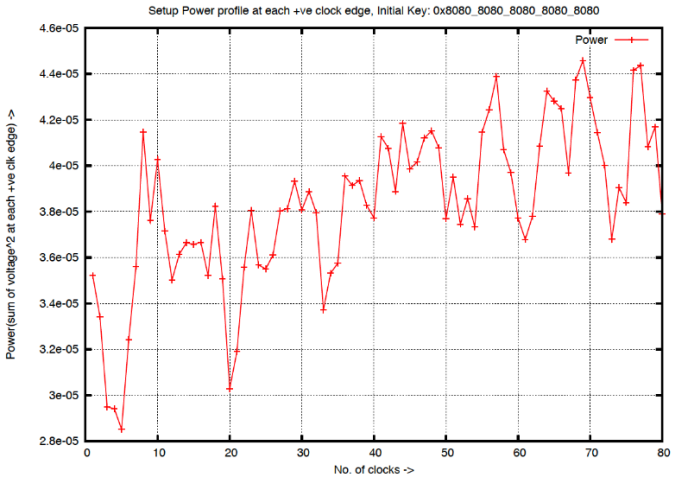
- We can try to estimate the power consumption based on model
- The estimation can be as simple as HW (Humming Weight) or HD (Humming Distance)
- But, CPA may involve evaluating the degree of correlation between the power consumption of the device under test and different options of the secret. Such model assumes a known dependencies between the model and the data being leaked.

Why HW/HD are reasonable models

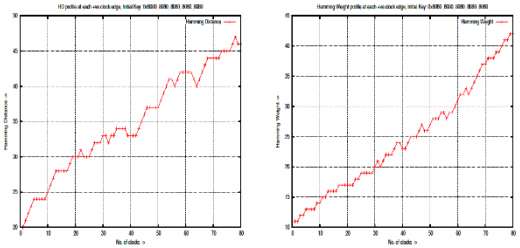
s	HW(s)	Target bit (LSB)
0000	0	0
0001	1	1
0010	1	0
0011	2	1
0100	1	0
0101	2	1
0110	2	0
0111	3	1
1000	1	0
1001	2	1
1010	2	0
1011	3	1
1100	2	0
1101	3	1
1110	3	0
1111	4	1

- Assume power leakage follows Hamming Weight.
- Divide the HW(s) into two bins:
  - 0 bin: when LSB is 0
  - 1 bin: when LSB is 1
- Difference-of-Mean (DoM)=20/8-12/8=1

HD vs. HW



Actual power traces taken as voltage drop. The power traces for 80 consecutive clock cycles after de-asserting reset



(a) Hamming Distance plot of the implemented stream cipher

(b) Hamming Weight plot of the implemented stream cipher

Power Simulation using Hamming Weight and Hamming Distance Models

Correlation Power Attack (CPA) of AES

- Like DoM based DPA, CPA also relies on targeting an intermediate computation, typically the input or output of an S-Box.
- These input values are computed from a known value, say the ciphertext and a portion of the key, which is guessed.
- The power model is then subsequently applied to develop a hypothetical power trace of the device for a given input to the cipher.
- These hypothetical power values are then stored in a matrix for several inputs and can be indexed by the known value of the ciphertext and the guessed key byte.
- This matrix is denoted as H, the hypothetical power matrix.

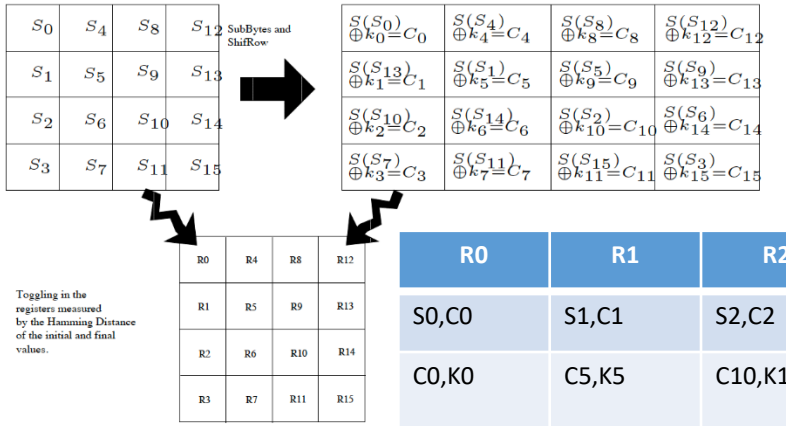
Correlation Power Attack (CPA)-Contd.

- The attacker also observes the actual power traces, and stores them in a matrix for several inputs.
- The actual power values can be indexed by the known value of the ciphertext and the time instance when the power value was observed.
- This matrix is denoted by T, the real power matrix.
- It may be observed that one of the columns of the matrix H corresponds to the correct key k*.
- CPA tries to compute the **similarity** between the columns of the matrix H and the columns of the matrix T, to distinguish k* from rest: similarity computed by Pearson's Correlation, usually.

Computing Correlation Coefficient for Simulated Power Traces for AES

- Like before, we simulate the power profile for the iterative AES, this time by using Hamming Distance power model applied on the state registers updated after each round.
- The real power matrix is stored in the array trace[NSample][NPoint]
  - NSample: Number of Power Traces acquired
  - NPoint: Time instances for which the power values are observed. Here NPoint is 12.

Calculating the Hypothetical Power



R0	R1	R2	R3	R4	R5	R6	R7
S0,C0	S1,C1	S2,C2	S3,C3	S4,C4	S5,C5	S6,C6	S7,C7
C0,K0	C5,K5	C10,K10	C15,K15	C4,K4	C9,K9	C14,K14	C3,K3

R8	R9	R10	R11	R12	R13	R14	R15
S8,C8	S9,C9	S10,C10	S11,C11	S12,C12	S13,C13	S14,C14	S15,C15
C8,K8	C13,K13	C2,K2	C7,K7	C12,K12	C1,K1	C6,K6	C11,K11

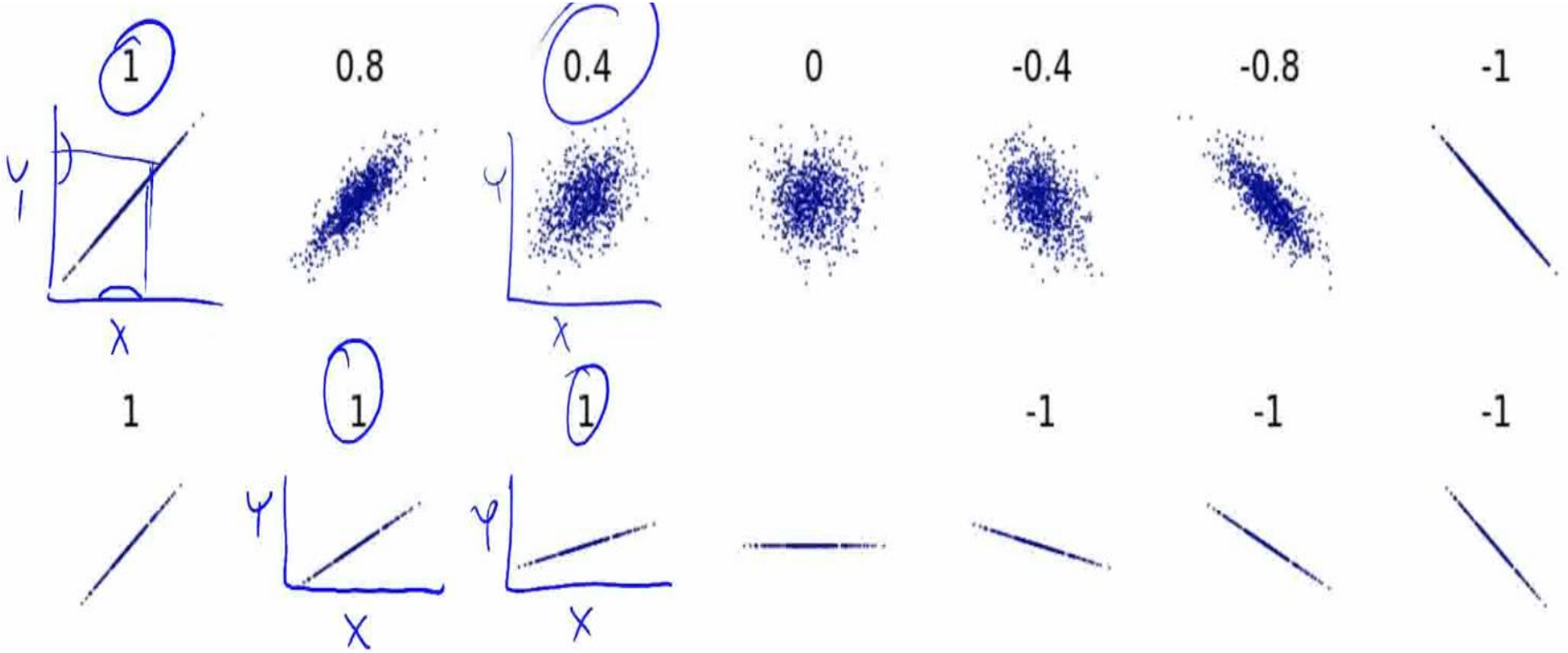
CPA - Correlation Power Analysis – cont.

- $W_{i,j}$ – is the measured power of the i^{th} -trace, at the j^{th} sample point.
- H_i – hypothetical power consumption value of the i^{th} plain text (or cipher text) with respect to the appropriate subkey

$$\hat{\rho} = \frac{N\sum_{i=0}^{N} W_{i,j}H_i - \sum_{i=0}^{N} W_{i,j}\sum_{i=0}^{N} H_i}{\sqrt{N\sum_{i=0}^{N} W_{i,j}^2 - (\sum_{i=0}^{N} W_{i,j})^2}\sqrt{N\sum_{i=0}^{N} H_i^2 - (\sum_{i=0}^{N} H_i)^2}}$$

- $\hat{\rho}$ is a called correlation number and always [-1,1]

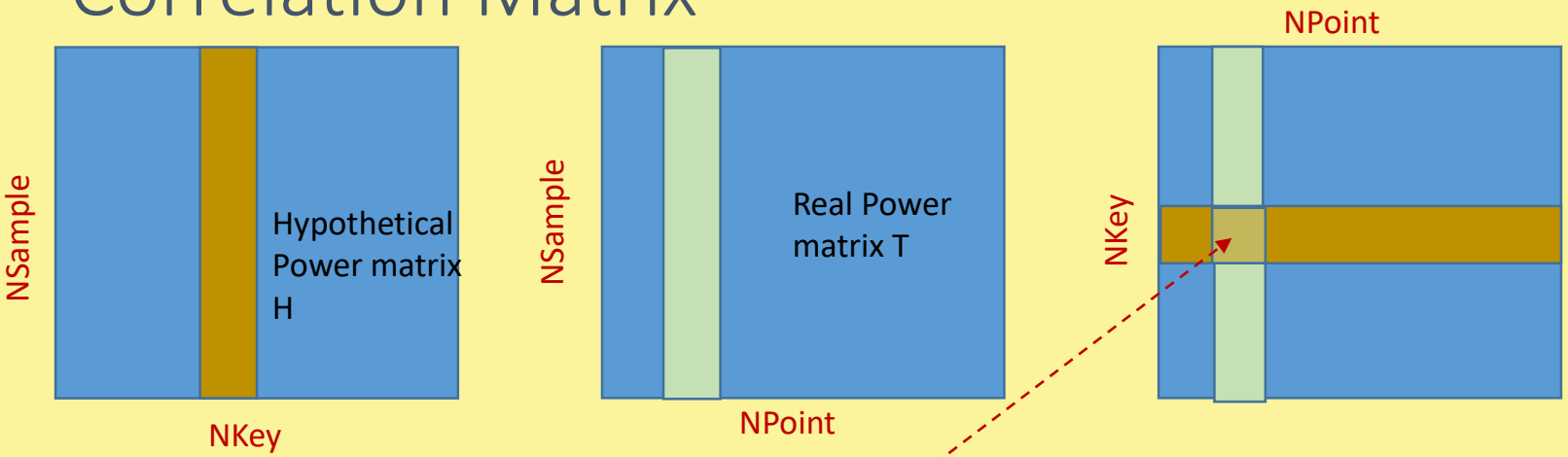
Now we need to perform Correlation between the set of expected results and the traces



Computing the Correlation Matrix

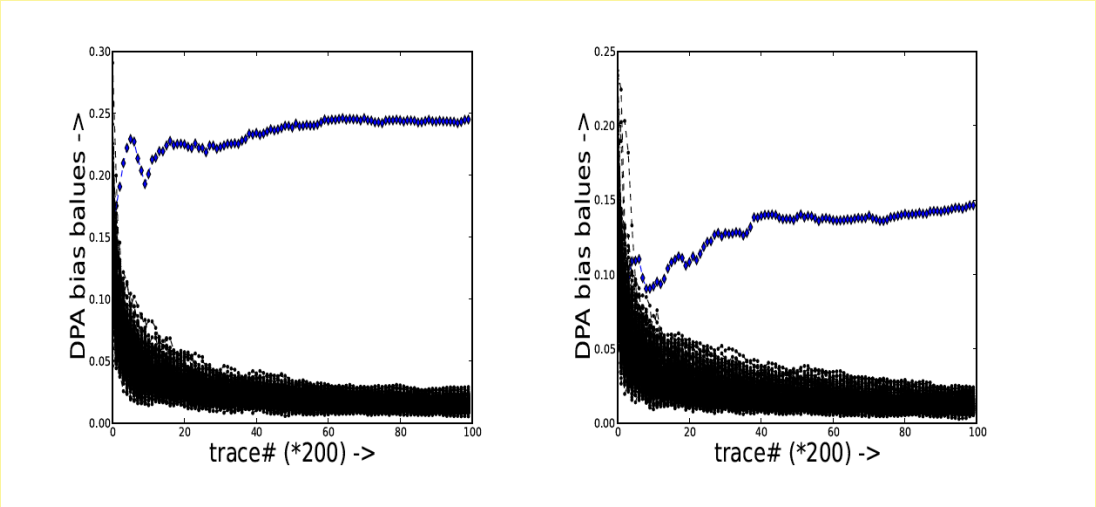
- Actual Power values for all the NSample encryptions are stored in the array trace[NSample][NPoint].
  - However, reflect as we are calculating hypothetical power using HD, the trick which we applied before for storing the traces compactly will not work.
  - Attacker first scans each column of this array and computes the average, and stores in meanTrace[NPoint].
- Likewise, the hypothetical power is stored in the array hPower[NSample][NKey].
  - Attacker scans each column and stores in the array meanH[NKey]

Correlation Matrix



$$C[i][j] = \frac{\Sigma_{k=0}^{NSample}(hPower[i][k] - meanH[i])(trace[j][k] - meanTrace[j])}{\Sigma_{k=0}^{NSample}(hPower[i][k] - meanH[i])^2 \Sigma_{k=0}^{NSample}(trace[j][k] - meanTrace[j])^2}$$

Experimental Results on Simulated Traces



CPA on first key byte of 10th Round of AES
Power is Simulated by Hamming Distance of the Registers after each Round

Power is Simulated by Hamming Distance of the Registers after each Round, with superimposed Gaussian Noise.

CPA example

- We want to discover the **correct key value (c_k)** and **when it is used (c_t)**
- Idea:
  - On the **correct time**, the power consumption of **all traces** is **correlated** with the **correct key**
  - On **other times** and **other keys** the traces should show **low correlation**

CPA Example -- Attacker has

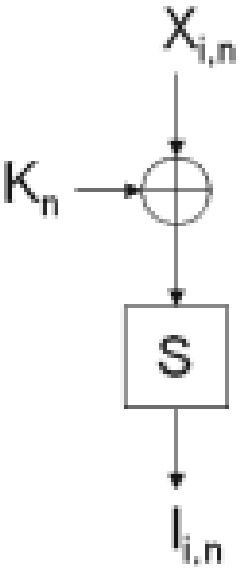
- **Power traces** $t_{d,j}$ (j=1,2,…,T is the time index and d is the trace number).

- Thus the attacker makes D measurements, each one T points long.
- If the attacker knew exactly where a cryptographic operation occurred, they would need to only measure a single point such that T=1.
- For each trace D, the attacker also knows the plaintext or ciphertext corresponding to that power trace, defined as P_d .

- A **model** of how the power consumption of the device depends on some intermediate value; e.g., power consumption dependents on the hamming weight of the intermediate value.
- Define a function $h_{d,l} = l(w(p_d,i))$
  - l(x) is the leakage model for a given intermediate value,
  - w(p,i) generates an intermediate value given the input plaintext *p* and the guess number i=1,2,….I.

Input Data	Power Measurement
0xC7	
0x1F	
0x2C	
0x89	
0x01	
0xD2	

D traces



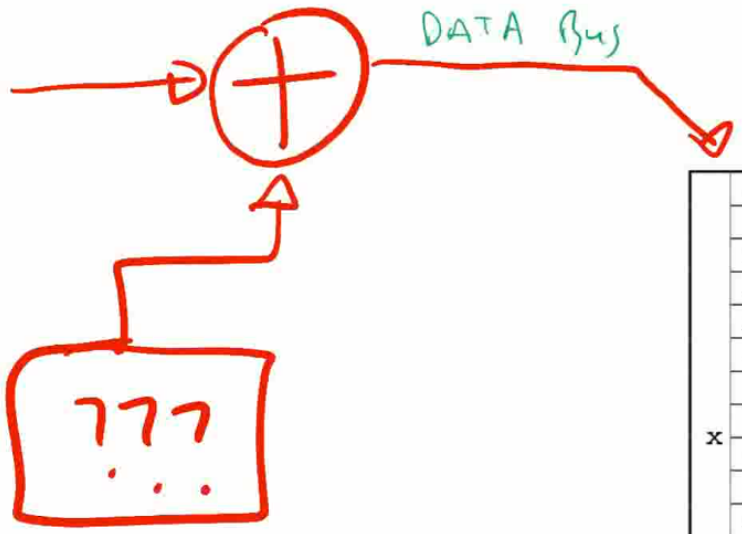
T point of time

Leakage function

- This intermediate value *w* depends on the input plaintext. P, and a small portion of the secret key. For example with AES, each byte of the plaintext (p) is XOR'd with each byte (subkey) of the secret key (i)
- $$l(x) = HammingWeight(x)$$
$$w(p,i) = p \oplus i$$
- This implies that the input plaintext is being attacked a single byte of the AES key at a time
  - While we still need to enumerate all possibilities for this subkey, we now have to check only $16X2^8$ options instead of 2^{128}possibilities for AES-128.

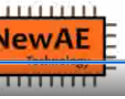
Lookup Table (AES Substitution Box or S-Box)

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16



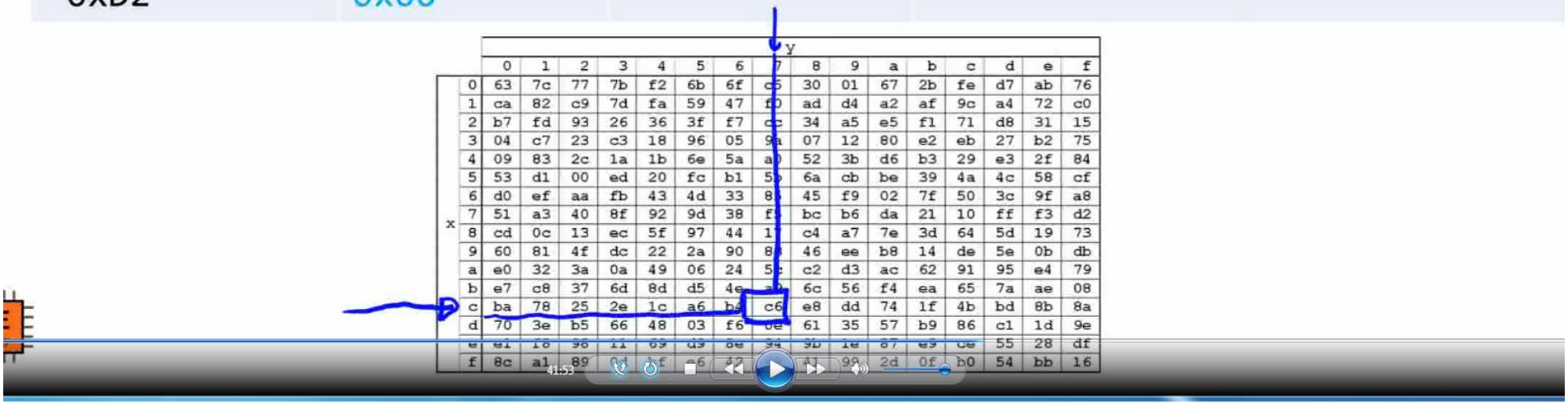
Lookup Table (AES Substitution Box or S-Box)

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16



Attack Results

Input Data	Hyp. Key	XOR Output	Hyp. Output	Number 1's
0xC7	0x00	0xC7	0xC6	4
0x1F	0x00	0x1F	0xC0	2
0x2C	0x00	0x2C		
0x89	0x00	0x89		
0x01	0x00	0x01		
0xD2	0x00			



Attack Results

Input Data	Hyp. Key	XOR Output	Hyp. Output	Numbe
0xC7	0x00	0xC7	0xC6	4
0x1F	0x00	0x1F	0xC0	2
0x2C	0x00	0x2C	0x71	4
0x89	0x00	0x89	A7	5
0x01	0x00	0x01		
0xD2	0x00			

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

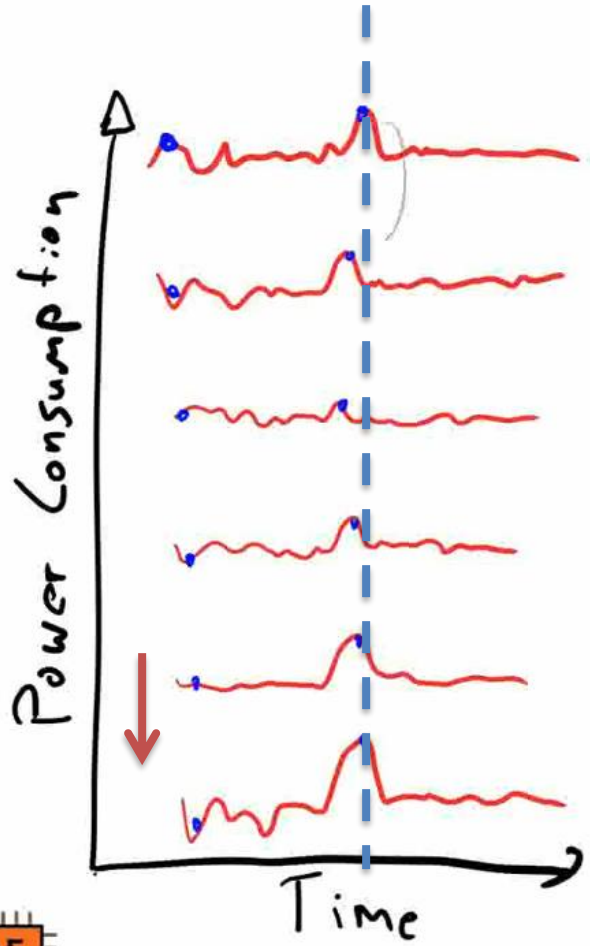
Attack Results

Input Data	Hyp. Key	XOR Output	Hyp. Output	Number 1's
0xC7	0xFF	0x38	0x07	3
0x1F	0xFF	0xE0	0xE1	4
0x2C	0xFF	0xD3	0x71	4
0x89	0xFF	0x76	0x38	3
0x01	0xFF	0xFE	0xBB	6
0xD2	0xFF	0x2D	0xB5	5

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

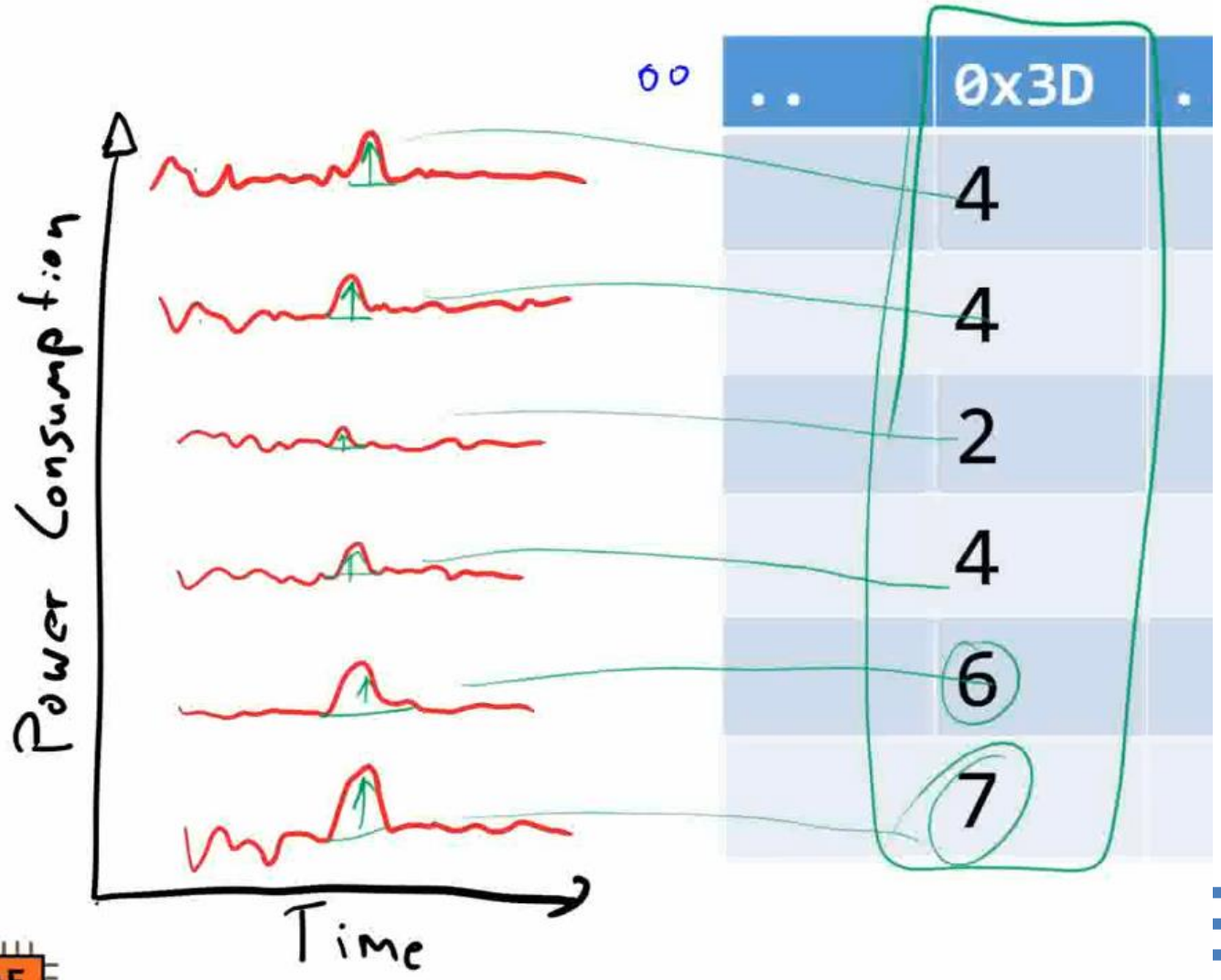
Related to power consumption at t_a

No good correlation

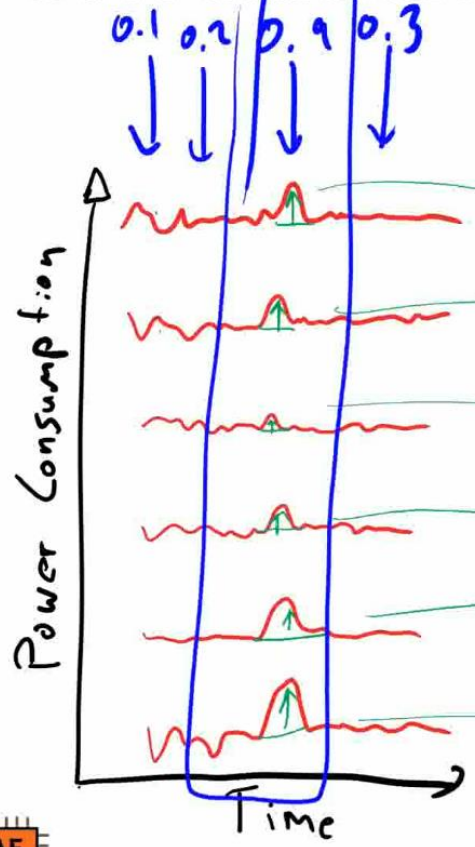


..	0x3D	..	0xFF
	4		3
	4		4
	2		4
	4		3
	6		6
	7		5

Good Correlation



Correlation Power Analysis



..	0x3D	..	0xFF
	4		3
	4		4
	2		4
	4		3
	6		6
	7		5

1. Most likely value.

2. Most likely at this point in time.

8 bits set to "1"

NewAE

Alignment of traces

- In “real-life” alignment of traces are often needed; e.g., if starting point of the measurement vary
- CPA like techniques can be used to align the traces. For that purpose we can use a CONV(X,Y) function to find the best alignment of the traces.

Template attacks

- The goal here is to generate an accurate power model; e.g., this will help to improve correlation attacks

Each point of a power trace can be modeled as the sum of an operation-dependent component P_{op}, a data-dependent component P_{data}, electronic noise $P_{el.noise}$, and a constant component P_{const}.

$$P_{total} = P_{op} + P_{data} + P_{el.noise} + P_{const} \quad (4.1)$$

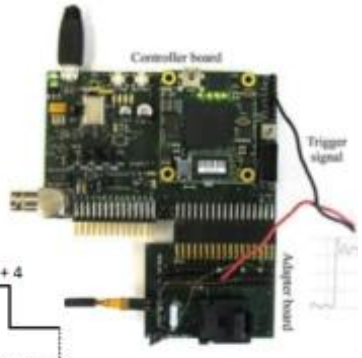
From: Power Analysis Attacks: Revealing the Secrets of Smart Cards Softcover reprint of hardcover 1st ed. 2007 Edition (reference in Moodel)

Fault Attacks : A Brief Overview

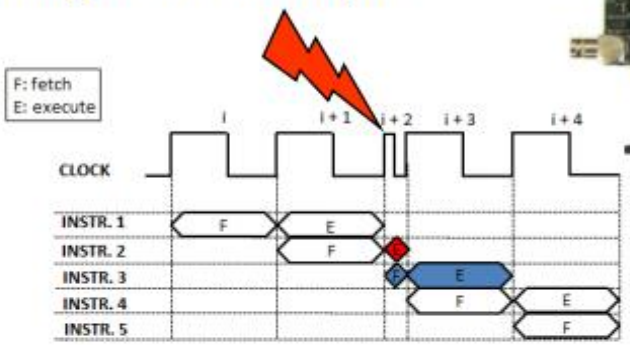
- Introduction of faults in the normal execution of cryptographic algorithms and analysis of faulty output to obtain the key
- First conceived in 1996 by Boneh, Demillo and Lipton
- E. Biham and Adi Shamir developed Differential Fault Analysis (DFA) of DES
- Today there are numerous examples of fault analysis of block ciphers such as AES under a variety of fault models and fault injection techniques

Fault Injection Attacks - Non-invasive

- Most popular form of non-invasive attacks
- Both precise timing control, single or multiple



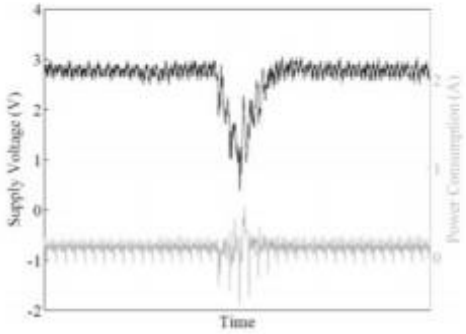
- Clock glitches
  - Temporal overclocking
  - Critical path violations



[BGV11]

- Voltage spikes
  - Temporal switch to higher (or lower) voltages

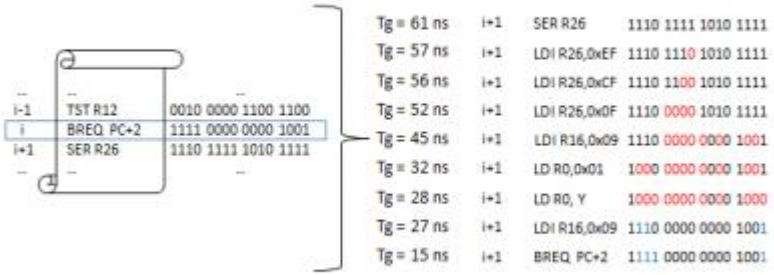
[KQ07]

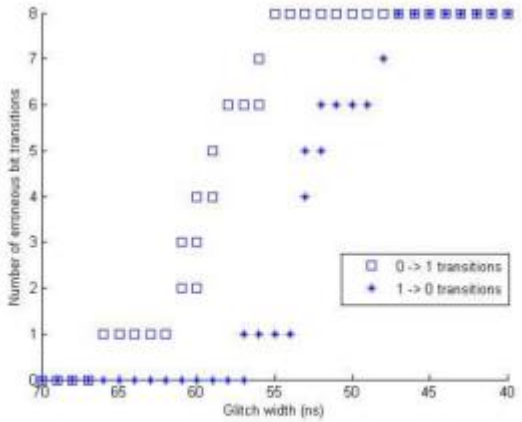


[SH08]

Glitches and spikes

- Effects on program flow
  - Replacement of instructions (sometimes skipping)
  - Tampering with loops and conditional statements
  - Change of program counter
    - Effects on data flow
    - Computation errors
    - Corrupted memory pointers
    - No bit transitions on data bus

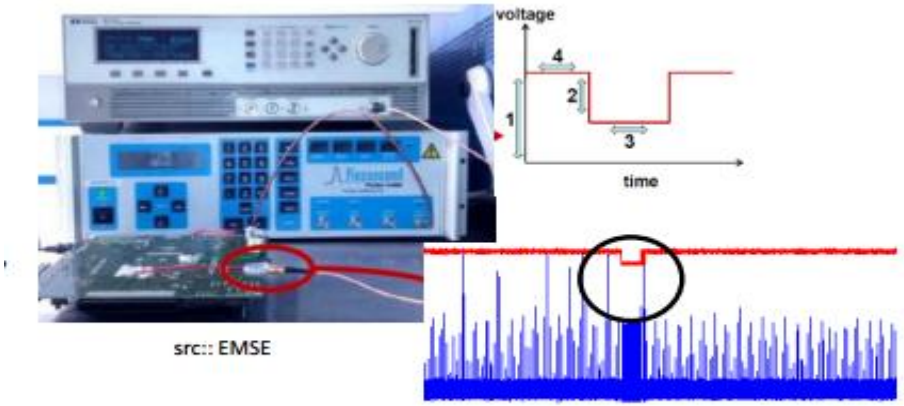




[BGV11]

Other Non-invasive Methods

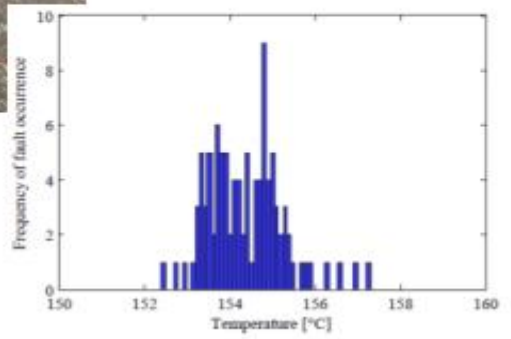
- Underpowering
  - Reduce supply voltage
  - Transient vs. Permanent
  - Increase propagation delay of combinational logic
- Temperature
  - Device on heating plate
  - Errors appear for a short window
    - Low-controlability
    - Low-frequency
  - Cooling: data retention



src:: EMSE

[BGVLV12]

[HS13]



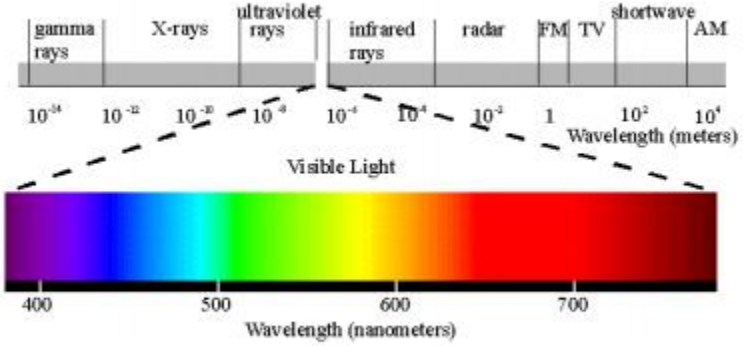
Semi Invasive attacks

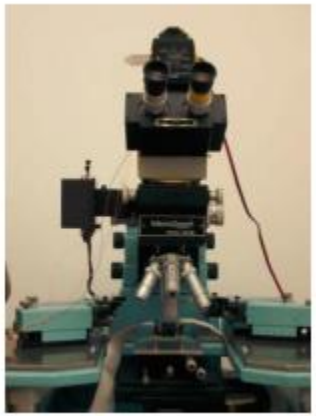
- Backside Decapsulation
  - Backside Imaging
  - Laser Scanning
  - Reverse Engineering
- Fault Injection Attacks
  - Local Heating
  - Flash Glitching
  - Laser Glitching

Optical Fault Injection

- Semiconductors are inherently sensitive to light
- Effect of optical pulses
  - Switching a transistor
- The chip die needs to be exposed
  - Semi-invasive method

- Example of fault injection setups:
  - Photo flash in micro-probing station
  - Laser beam on XY table, with microscope view and camera



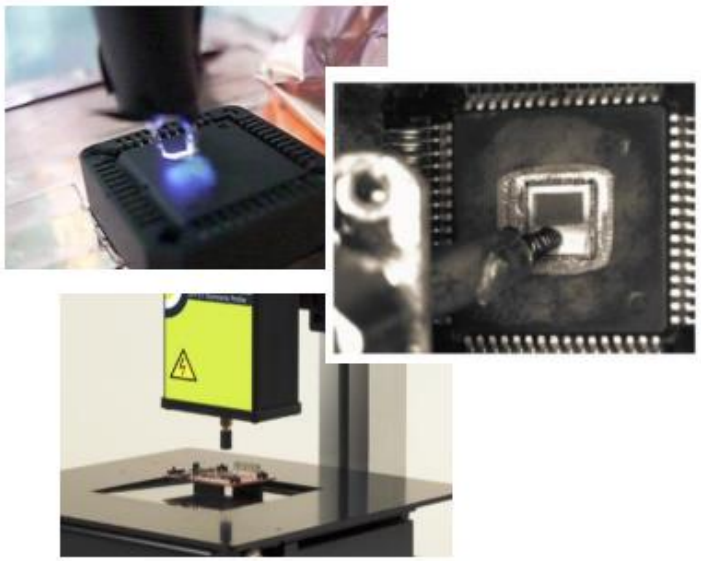


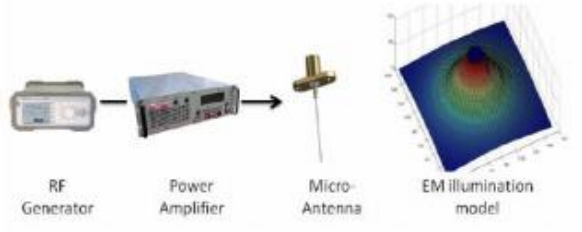
[SA02]

src: Opto

EM Fault Injection

- Injection of faults via the EM channel
  - Induction of Eddy current
    - Camera flash-gun connected to an active probe
    - Spark-gap transmitter
    - EM Pulses with micro probes
  - Effects:
    - Switching transistors
    - Critical path violations
  - (Non-) and semi- invasive approach

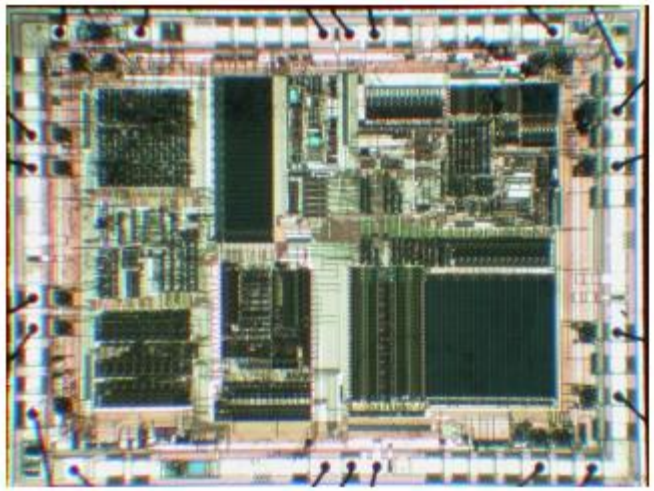




Targets of fault attacks

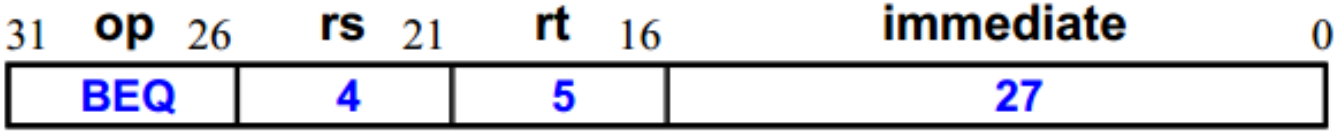
- Common components
  - CPU
  - Memory
  - I/O
  - A/D and D/A

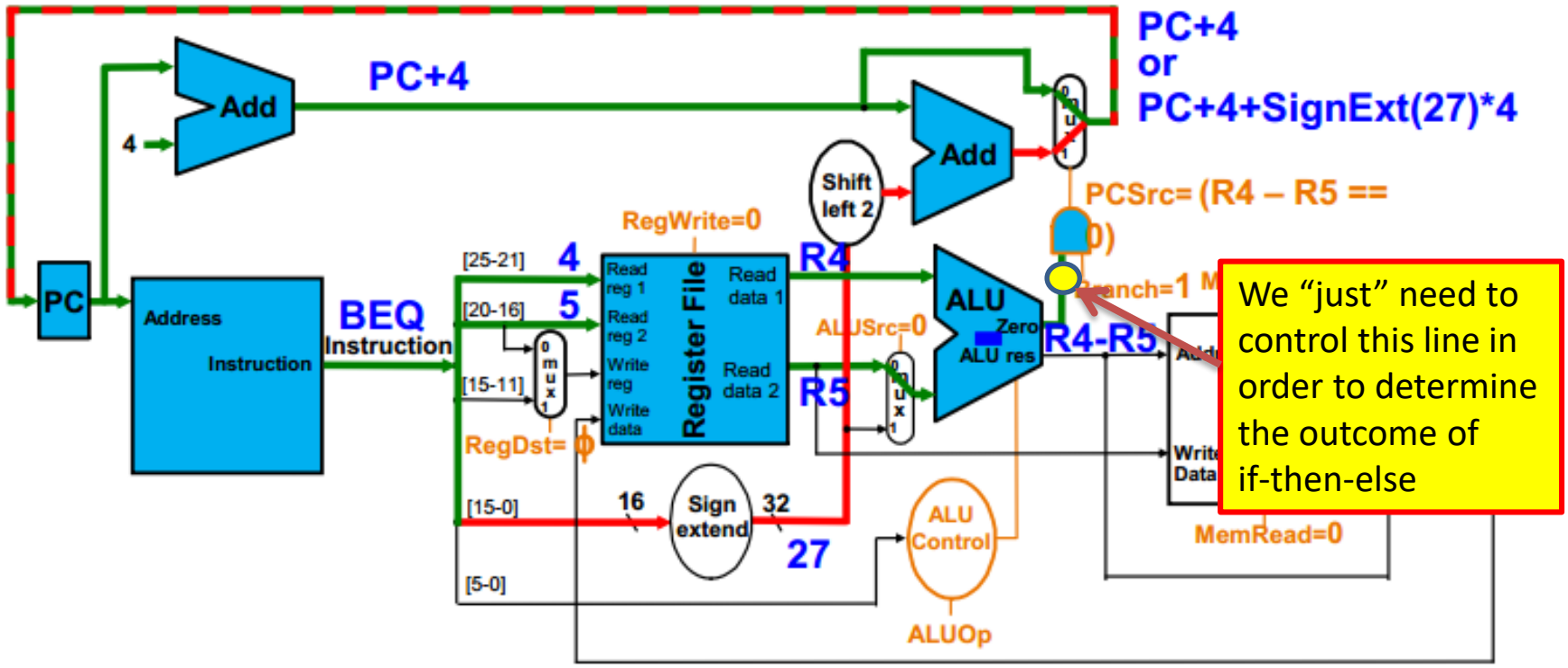
- Embedded memory
  - SRAM, EEPROM, Flash, ROM
- CPU
  - instructions, data, result, jumps
- Security
  - reset fuse, force debug, access



Executing a BEQ Instruction

BEQ R4, R5, 27 ; if (R4-R5=0) then PC ← PC+4+SignExt(27)*4 ;
else PC ← PC+4





Concept of the proposed attack

- Assumption
  - Feeding glitchy clock signal into CPU enables instruction skip(s).

Glitchy clock signal

```
PUSH    R29          Push register on stack
PUSH    R28          Push register on stack
RCALL   PC+0x0001    Relative call subroutine
RCALL   PC+0x0001    Relative call subroutine
PUSH    R0           Push register on stack
IN      R28,0x3D     In from I/O location
IN      R29,0x3E     In from I/O location
```

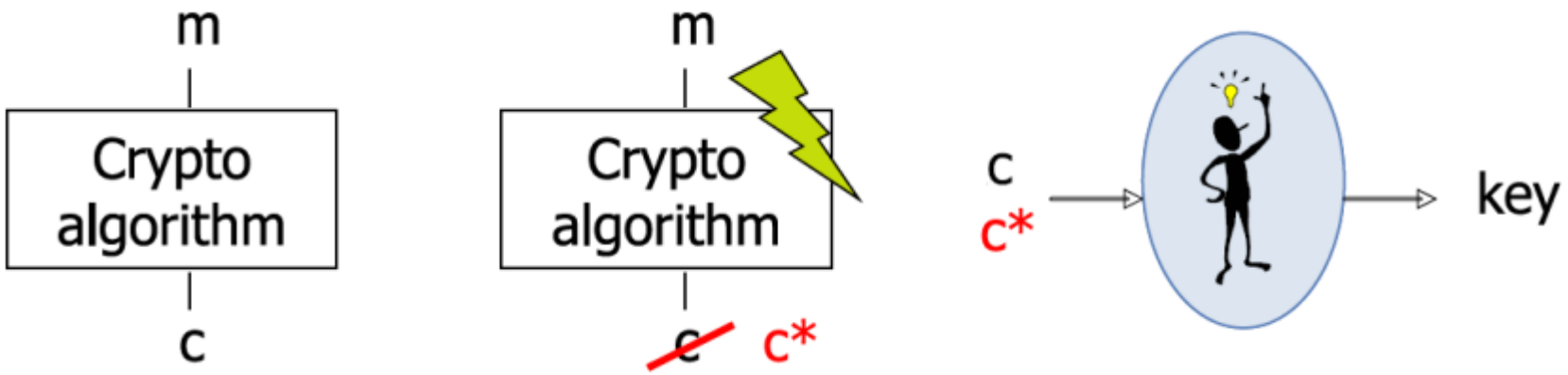
Skip multiple and arbitrary instructions

- Attack method
  - Skip a few instructions while input attack code, and invalidate boundary check to make buffers overflowed
  - Take control of CPU like common attacks

DIFFERENTIAL FAULT ANALYSIS

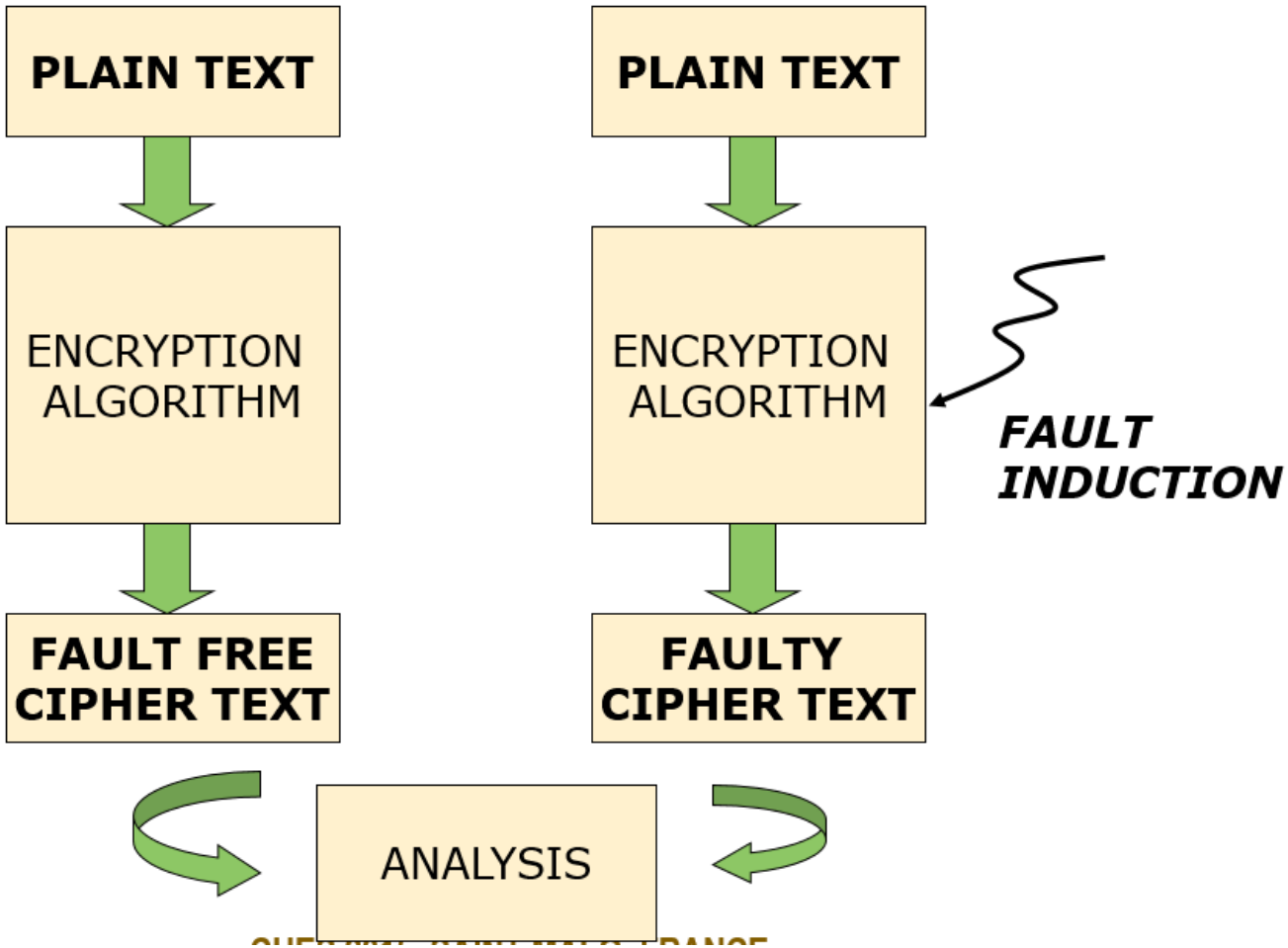
Differential Fault Analysis

- Ask for a cryptographic computation twice
  - With any input and no fault (reference)
  - With the same input and fault injection
- Infer information about the key from the output differential



- Sometimes a single fault injection is enough!

Illustration of DPA



Bellcore = Bell Communications Research

- Three employees of Bellcore (and Pr. @ Stanford) find an attack that breaks RSA by injecting a single fault.
- Reference: "*On the importance of checking cryptographic protocols for faults*". by D. Boneh, R. DeMillo, and R. Lipton. Journal of Cryptology, Springer-Verlag, Vol. 14, No. 2, pp. 101–119, 2001. Extended abstract in Proceedings of Eurocrypt'97, Lecture Notes in Computer Science, Vol. 1233, Springer-Verlag, pp. 37–51, 1997.
- http://crypto.stanford.edu/~dabo/papers/faults.ps.gz

An Attack on "RSA–CRT"

- Signature S of message x: $S \doteq x^d \mod N$, with $N = p \cdot q$.
- Using Chinese Remainder Theorem (CRT), the signature can be simplified as:
  - $S_1 = x^{d \mod (p-1)} \mod p$ and
  - $S_2 = x^{d \mod (q-1)} \mod q$, both operations working on half bitwidth.
- The signature is obtained back using the two constants:

$$\begin{cases} a = 0 \mod q \\ a = 1 \mod p \end{cases} \quad \text{and} \quad \begin{cases} b = 1 \mod q \\ b = 0 \mod p \end{cases}$$

- $S = a \cdot S_1 + b \cdot S_2 \mod N$.
- Now, if S_1 happens to be faulty: $S_1 \to \widehat{S_1}$ for whatever reason,
- $\gcd(S - \widehat{S}, N) = \gcd(a \cdot (S_1 - \widehat{S_1}), N) = q$.

General attack (no CRT)

- The attack proceeds as follows: the attacker asks the black-box to sign messages M1, M2,…,Ml.
- The attacker collects the responses until she has sufficiently many erroneous signatures S'i.
- The pairs <Mi; S'i> are then used to deduce the secret signing key d.
- We assume that for each pair <Mi; S'i> a single register fault occurs during the
- computation of S'i. The fault occurs at a random iteration during the exponentiation algorithm and flips one bit of the value stored in the variable z.

Algorithm 1

init $y \leftarrow x \quad ; \quad z \leftarrow 1.$
main For $k = 0, \ldots, n-1.$
If $d_k = 1$, then $z \leftarrow z \cdot y \pmod N$.
$y \leftarrow y^2 \pmod N$.
Output z.

1. For all lengths $r = 1, 2, 3 \ldots, m$ do:
2. For all candidate r-bit vectors $u = u_{a-1}u_{a-2}\cdots u_{a-r}$ do:
3. Set $w = \sum_{j=a}^{n-1} d_j 2^j + \sum_{j=a-r}^{a-1} u_j 2^j$. In other words, w matches the bits of d and the bits of u at all bit positions that are already exposed and is zero everywhere else.
4. Test if the current candidate bit vector u is correct by checking if one of the erroneous signatures $\hat{S}_j$ for $j = 1, \ldots, l$ satisfies

$$\exists b \in \{0, \ldots, n\} \qquad \text{s.t.} \quad \left(\hat{S}_j \pm 2^b M_j^w\right)^e = M_j \pmod N.$$

Recall that e is the public signature verification exponent. The $\pm$ means that the condition is satisfied if it holds with either a plus or minus.

5. If a signature satisfying the above condition is found, output $u_{a-1}u_{a-2}\cdots u_{a-r}$ and stop. At this point we know that $k_{i-1} = c = a - r$ and $d_{a-1}d_{a-2}\cdots d_{a-r} = u_{a-1}u_{a-2}\cdots u_{a-r}$. Hence, r more bits of d are exposed.

Effect of Error on AES

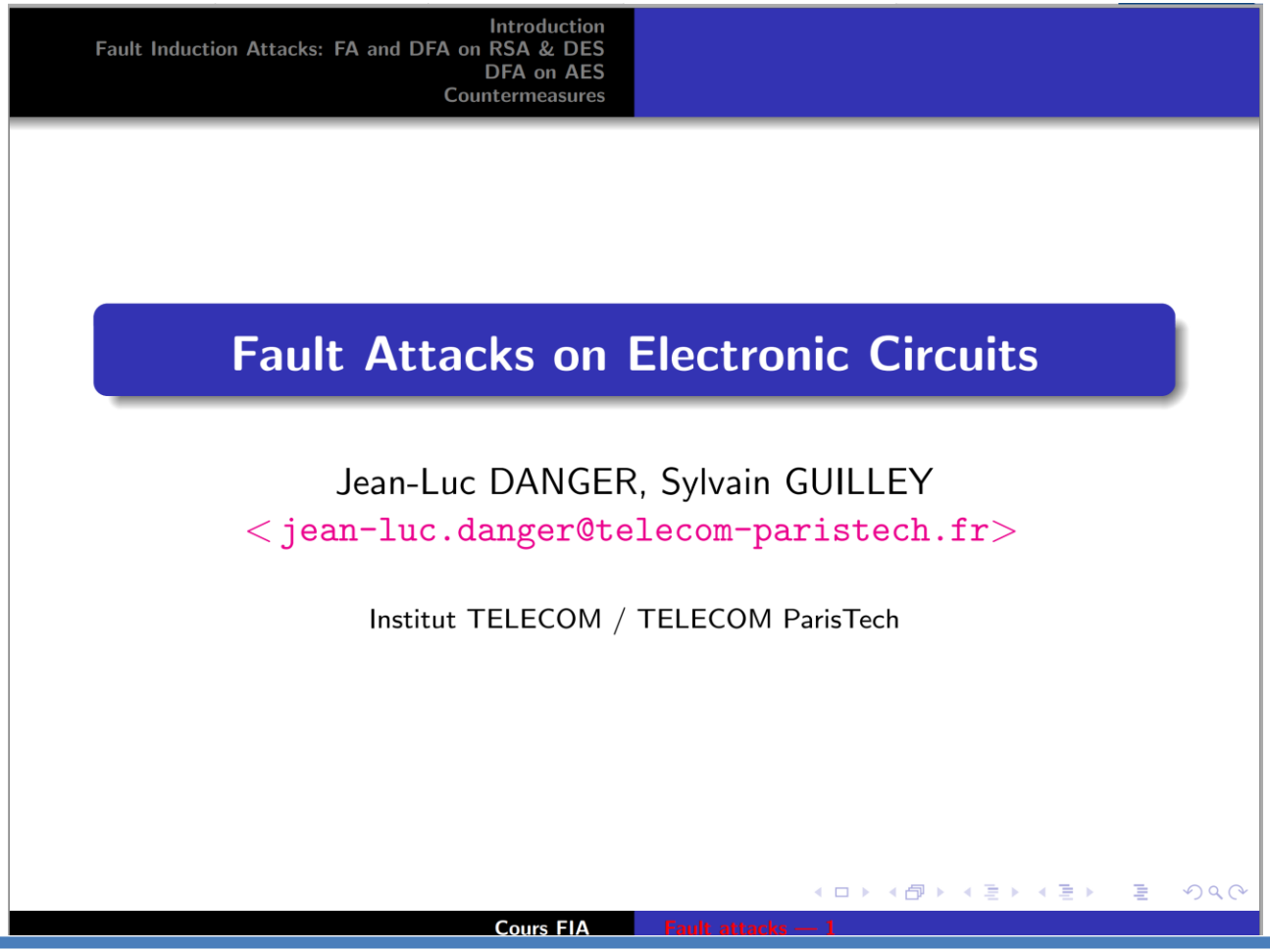
- **Plaintext:**
  32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34
- **128-bit key:**
  2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
- **Ciphertext:**
  39 25 84 1d 02 dc 09 fb dc 11 85 97 19 6a 0b 32

- **A single error in the plaintext:**
  30 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34
- **Results in the ciphertext:**
  c0 06 27 d1 8b d9 e1 19 d5 17 6d bc ba 73 37 c1
- **A single error in the key:**
  2a 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
- **Results in the ciphertext:**
  c4 61 97 9e e4 4d e9 7a ba 52 34 8b 39 9d 7f 84

- **A single-bit error results in a totally scrambled output**

Source : Koren and Krishna, Morgan-Kaufman 2007

Slides from



TECHNION
Israel Institute
of Technology

Giraud 2003

Bit-fault e_j attack on the last round

Regular encryption (C):

- $C_{\texttt{ShiftRow}(i)} = \texttt{SubBytes}(M_i^9) \oplus K^{10}_{\texttt{ShiftRow}(i)}$ for $i \in [1, 16]$

Faulted encryption (D):

- $D_{\texttt{ShiftRow}(i)} = \texttt{SubBytes}(M_i^9) \oplus K^{10}_{\texttt{ShiftRow}(i)}$ for $i \in [1, 16] \backslash \{j\}$ and
- $C_{\texttt{ShiftRow}(j)} = \texttt{SubBytes}(M_j^9 \oplus e_j) \oplus K^{10}_{\texttt{ShiftRow}(j)}$.

Attack:

- $C_{\texttt{ShiftRow}(j)} \oplus D_{\texttt{ShiftRow}(j)} = \texttt{SubBytes}(M_j^9) \oplus \texttt{SubBytes}(M_j^9 \oplus e_j)$.

Details of Giraud's attack

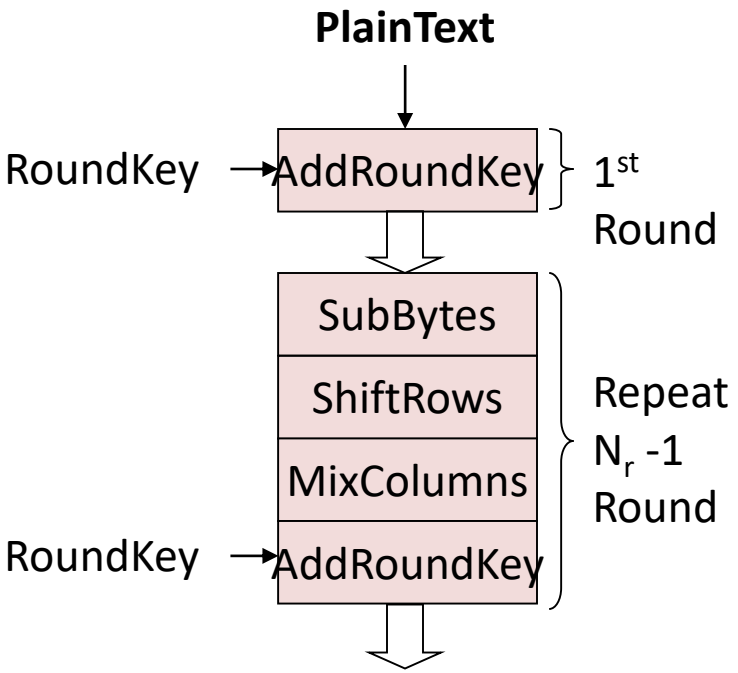
Goal: finding the value of M_j^9.

- $C_{\texttt{ShiftRows}(j)} \oplus D_{\texttt{ShiftRows}(j)} = \Delta = SubBytes(M_j^9) \oplus SubBytes(M_j^9 \oplus e_j)$ has between 2 and 14 solutions in (e_j, M_j^9) (set of 8×2^8 unknown), and 8 in average.
- However, the exact value of e_j is of no importance.

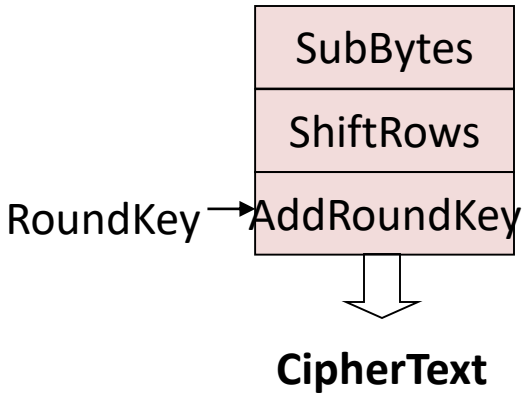
Attack Strength

- Thus, with 2 faults, there is 50 % chance to get one M_j^9.
- With 3 faults, there is 97 % chance to get one M_j^9.
- Once M_j^9 is known, we have $K_j^{10} = C_j \oplus SubBytes(M_j^9)$.

AES Algorithm: Our Target Cipher

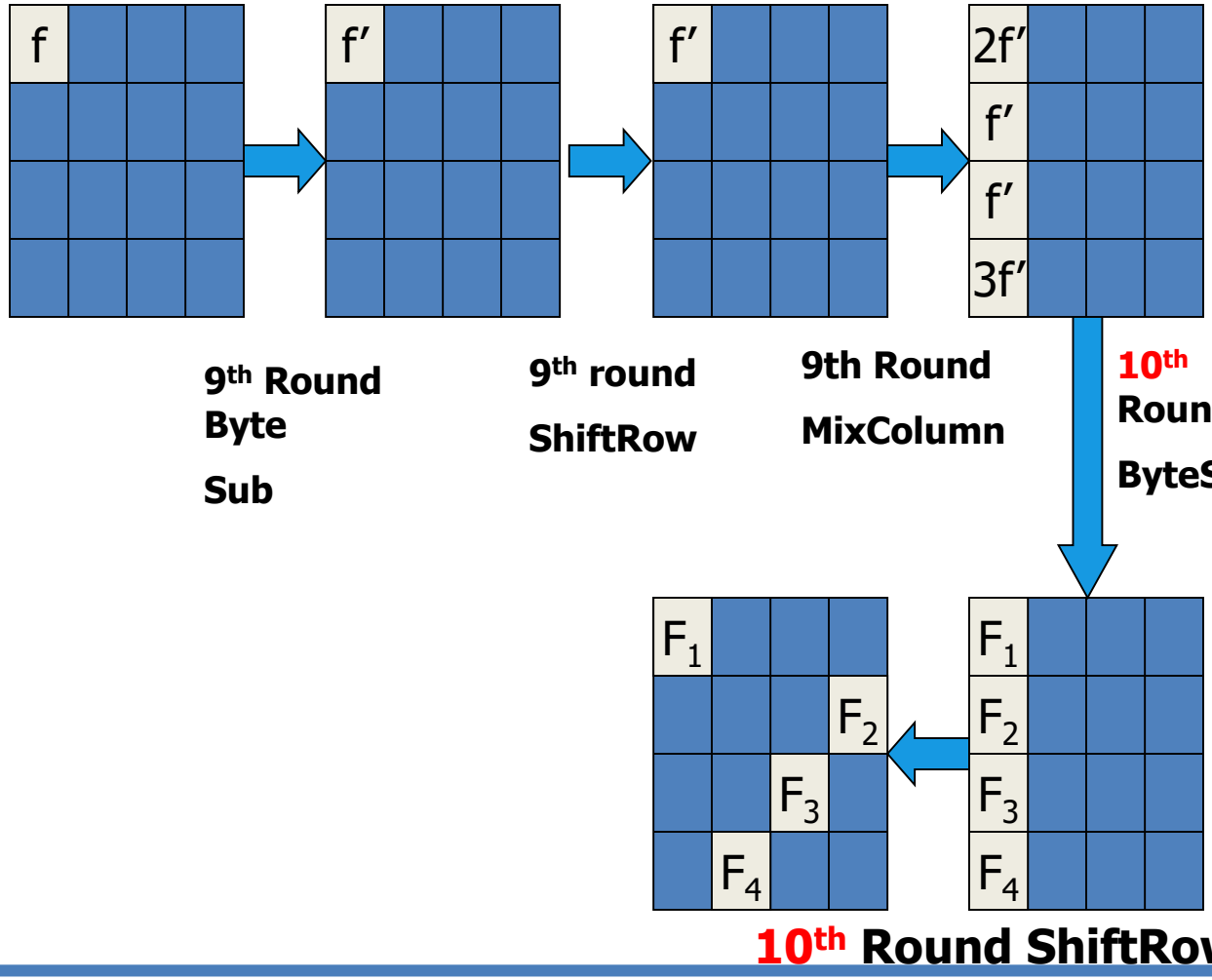


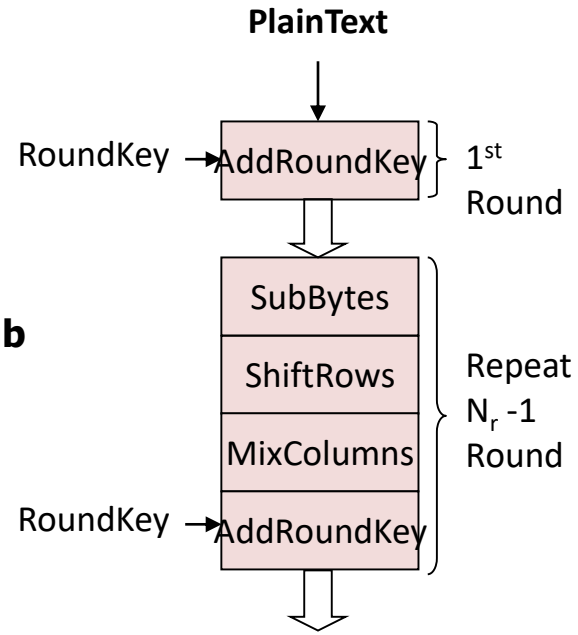
First 9 Rounds



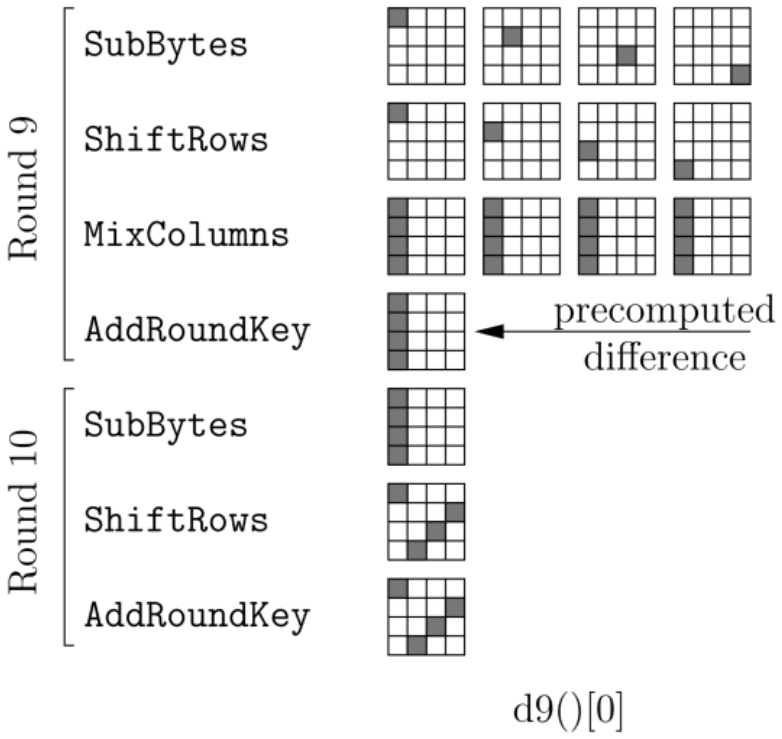
Last Round

Fault analysis on block ciphers





G. Piret & J.-J. Quisquater in 2003

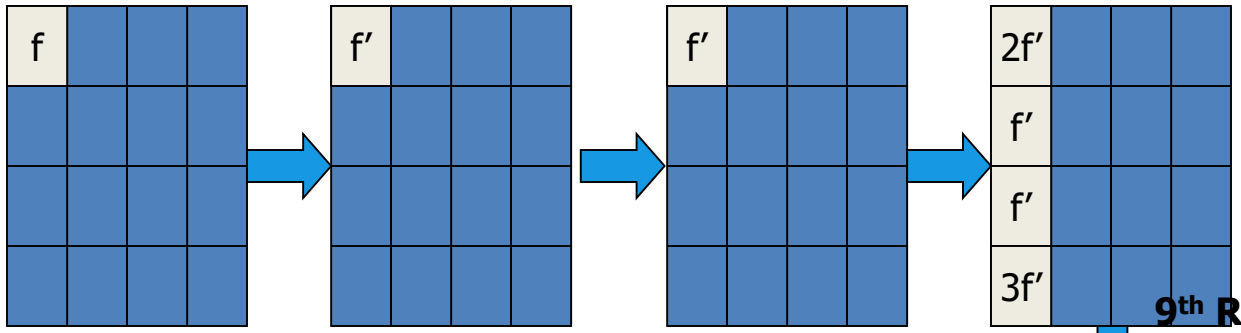


One faulty Byte at round 9 generates 4 faulty Bytes at the output
255x4 candidates for K_{10}
2 faults $\Rightarrow$ 98%
8 faults $\Rightarrow$ 100%

When the Fault is Induced in the 8th Round…

- Fault is induced at the input of 8th round
- A one byte disturbance creates a 4 byte fault at the input of the 9th round
- Let us trace the disturbance through the last 3 rounds
- Equations of similar nature…

Propagation of Fault Induced

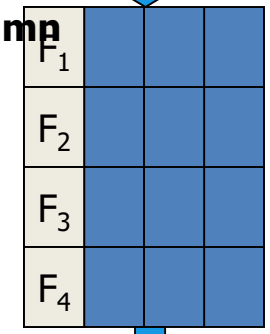


8th Round Byte Sub

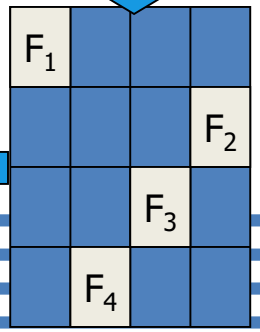
8th round ShiftRow

8th Round MixColumn

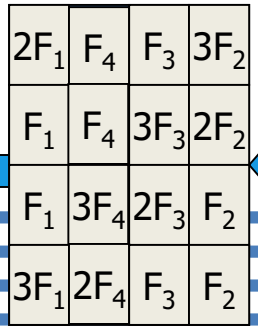
9th Round ByteSub



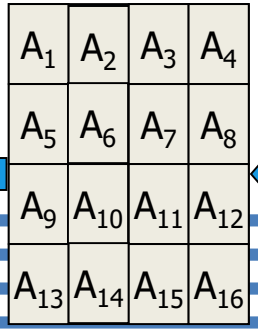
9th Round ShiftRow



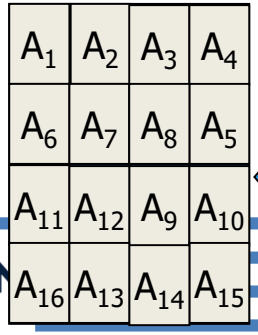
9th Round MixColumn



10th Round Byte Sub



10th Round Shift Row



Kill 4 birds with one stone

A fault at round 8 yields 4 faults at round 9! This is optimal...

Fault Attacks on RSA (Boneh et al. 1996)

Background:

Sender $M^e \bmod N \rightarrow S$ **(M- massage, e-private key, N public key)**

Receiver $S^d \bmod N \rightarrow M$ **(M- massage, d-private key, N public key)**

- Only decryption is subject to attacks
- Assume:
  1. Attacker can flip a single bit in key d
  2. S and corresponding message M known to attacker

Decryption device generates $\hat{M}$ satisfying s

$$\frac{\hat{M}}{M} = \frac{S^{2^i \bar{d}_i}}{S^{2^i d_i}} \bmod N$$

- If $d_i = 0$ then $\hat{M}/M = S^{2^i} \bmod N$
- If $d_i = 1$ then $\hat{M}/M = 1/S^{2^i} \bmod N$

Source : Koren and Krishna, Morgan-Kaufman 2007

Fault Attacks on RSA (contd.)

- Assume that the attacker flips randomly a bit in d.

- Example: (e,N)=(7,77), d=43 $d_5 d_4 d_3 d_2 d_1 d_0 = 101011_2$
  - Ciphertext=37 producing M=9 if no fault is injected and if a fault is injected $\hat{M} = 67$
  - Search for i such that $9 = (67 \cdot 37^{2^i}) \bmod 77$

    i=3 $(d_3 = 1)$ since

$$(67 \cdot 37^8) \bmod 77 = (67 \cdot 53) \bmod 77 = 9$$

Source : Koren and Krishna, Morgan-Kaufman 2007

Fault Attacks on RSA (contd.)

- Assume that the attacker flips randomly a bit in d. $d_5 d_4 d_3 d_2 d_1 d_0 = 101011_2$

- Example: (e,N)=(7,77), d=43

$\hat{M} = 67$

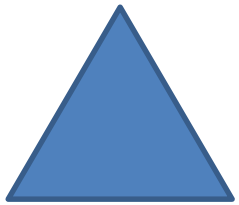
$9 = (67 \cdot 37^{2^i}) \bmod 77$

- Ciphertext=37 producing M=9 if no fault is injected and if a fault is injected $(d_3 = 1)$
- Search for i such that $(67 \cdot 37^8) \bmod 77 = (67 \cdot 53) \bmod 77 = 9$

i=3 since

Source : Koren and Krishna, Morgan-Kaufman 2007

Second exercise – part I



Backup