Power and Fault Analysis Resistance in Hardware through Dynamic Reconfiguration

Nele Mentens[1,2], Benedikt Gierlichs[1], Ingrid Verbauwhede[1]

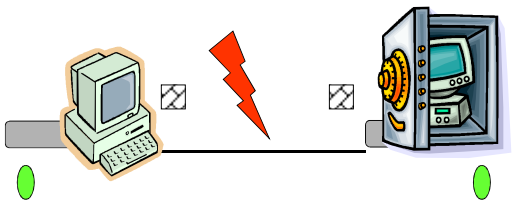
[1] K.U. Leuven, ESAT/SCD-Cosic
[2] KH Limburg, IWT

firstname.lastname@esat.kuleuven.be
www.cosic.be

Outline

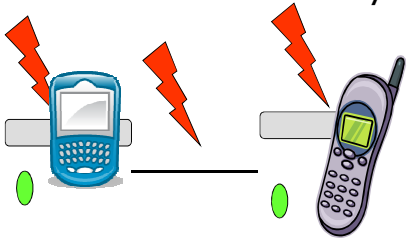
- Physical attacks
  - Passive (Side-Channel) Attacks
  - Active (Fault Injection) Attacks
- Partial dynamic reconfigurability
  - System overview
  - AES reference architecture
- Three types of countermeasures for reconfigurable devices
  - Temporal jitter
  - Spatial and temporal jitter
  - Fault detection
- Conclusions

Black-box Security



- The model for classical Cryptanalysis (simplified view):
- Attack on channel *between* communicating parties
- Cryptographic operations in *black* boxes
- Protect link with strong cryptography
- Provable security, computational, information-theoretic, etc.

Embedded Security



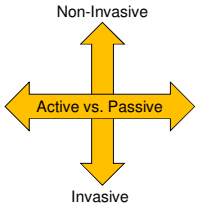
- The model for the embedded world (also simplified view):
- Attack on *channel and endpoints*
- Cryptographic operations in *gray* boxes
- Protect link with strong cryptography
- Protect cryptography with secure implementation

Physical Attacks

- Physical attacks ≠ classical cryptanalysis
  (gray box, physics) (black box, maths)
- Algorithm is an abstract mathematical object;
  Implementation is a physical instance of the mathematical object
- Breaking the physical instance does not imply breaking the mathematical object (while the opposite holds)
- An embedded device is exposed to its possibly hostile environment
  (interaction with external world, depends on power supply etc.)
- Physical attacks exploit weaknesses that were introduced when the algorithm was implemented

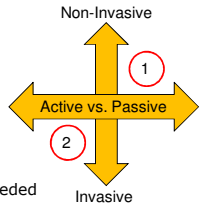
Classification of Physical Attacks (1)

- Active versus passive
  - Active: Perturbate and conclude
  - Passive: Observe and infer
- Invasive versus non-invasive
  - Non-invasive: chip package remains intact
  - Invasive: physical contact with chip
  - Semi-invasive: chip package open but no contact with circuit



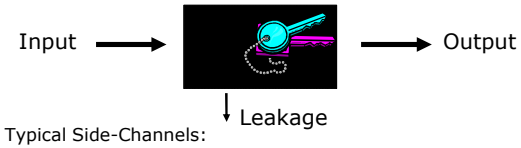
Classification of Physical Attacks (2)

1. Side-channel: passive and non-invasive
   - Very difficult to detect
   - Cheap to set up
   - Needs many measurements
2. Circuit modification: active and full-invasive
   - Very expensive to detect an invasion
     (you may be off power)
   - Expensive equipment and a lot of expertise needed
   - Very powerful (in-security à la carte)



- 1) and 2) are both difficult to detect, but for different reasons

Side-Channel Leakage and Attacks



- Typical Side-Channels:
  - Timing (non-constant execution time)
  - Power dissipation (data-dependency of dynamic power)
  - Electromagnetic radiation (data-dependency of local radiation)
  - Light, Accustic, Temperature and probably more
- Observe physical quantities in the device's vicinity and use the additional information during cryptanalysis
- "Attack" intermediate variables, which often depend on only a few key bits divide and conquer

Principle is nothing new...

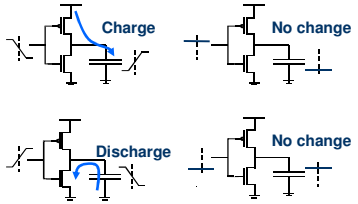
"Things are *very* different if it is possible to solve many small problems instead..."

"Breaking into a Safe is hard, because one has to solve a single, very hard problem..."

"*Divide et impera*!"

CMOS technology

- Low static power consumption
- Dynamic power consumption (switching) depends on circuit's activity



IN	OUT
0 0	0
0 1	discharge
1 0	charge
1 1	0

- Power analysis exploits that the instantaneous dynamic power consumption of a device depends on the data it processes

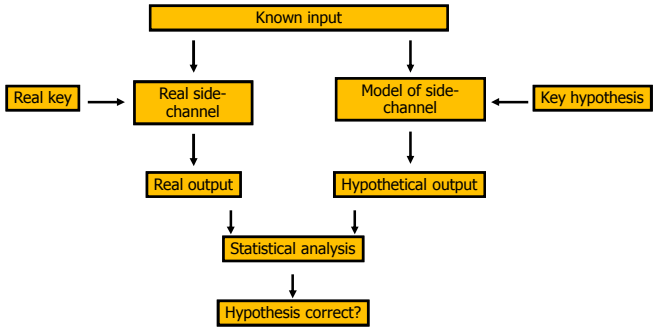
Differential Power Analysis

- Problem: we want to learn key bits
- Idea: side-channel leakage contains information
- Approach:
  - Obtain measurements, known input, unknown key
  - Establish a power consumption model
  - "Simulate" power dissipation at *early* stage of algorithm known input + guess on key bits (*divide and conquer*)
  - Compare simulation and reality by means of a statistical test to reject / accept key hypotheses

The seminal paper by Kocher et al. [1]

Power Analysis Tutorial by Aigner and Oswald [2]

Differential Power Attacks (generic)



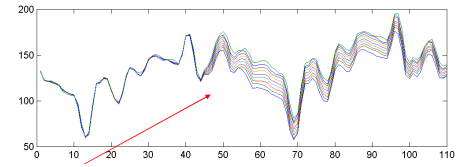
The Correlation Method (1)

- Exploits information leakage efficiently with few measurements
  - Need for a power model: e.g. # bit flips = Hamming dist. ~ dyn. Power
  - Works well, *if* power model is *meaningful*
  - Simple power model: a x HW($data_{t-1} \oplus data_t$) + b (CMOS)
- Intermediate result:
  - Several bits that depend on a few key bits and the input
  - Example AES: Sbox output in round one (1 Byte)
- Exhaustive search: for each sub-key hypothesis
  - For each measurement: "simulate" power consumption
  - Estimate Pearson correlation between simulation and measurements
  - Best key guess maximises the correlation coefficient

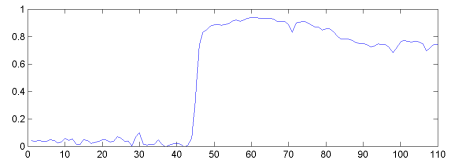
Brier et al. [3]

Hamming Distance Model

- Power



- Correlation



Countermeasures: Classification

- According to the level of implementation:
  - Protocol countermeasures
  - Software countermeasures
  - Hardware countermeasures
- According to their applicability:
  - Algorithm dependent
  - Algorithm independent
- Trade-off
  - Each countermeasure yields overhead (time, power, area)
  - Choice depends on desired security level
- Never rely on a single countermeasure!

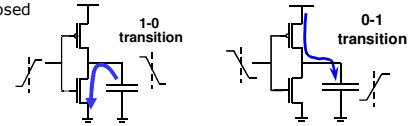
DPA countermeasures (1)

- Some are very much algorithm dependent, for example:
  - Implement algorithm with equal functionality (output as expected)
  - But make it randomize data and/or key (masking, blinding)
  - Idea is to live with side-channel leakage and to make it useless (can be overcome with higher-order attacks)
- Some are less dependent:
  - Identify independently computable values in algorithm
  - Randomize the *sequence* in which these values are computed
  - Idea: distribute the leakage over time
    - Software: random order execution [4]
- Some are independent:
  - Insert delays of random duration in the sequence
  - Idea: distribute the leakage over time
    - Software: dummy instructions, random process interrupts [5]

(can be overcome with re-synchronization, but difficult)

DPA countermeasures (2)

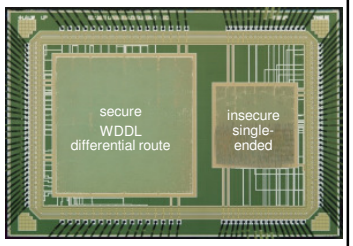
- Some are independent:
  - Tackle the problem of side-channel leakage at its root
  - Implement in a *secure logic* style (Hiding, masking)
  - Many logic styles proposed *and broken!*
  - Duplicate logic, as suggested by famous cryptographers ...



IN	$\overline{\text{IN}}$	OUT	$\overline{\text{OUT}}$
0→0	1→1	0	0
0→1	1→0	discharge	charge
1→0	0→1	charge	discharge
1→1	0→0	0	0

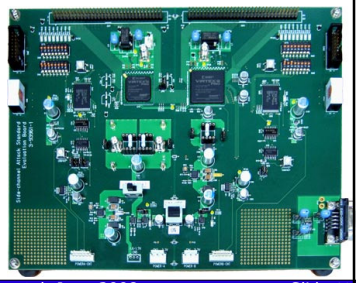
DPA countermeasures (3)

- Wave Dynamic Differential Logic (WDDL)
- Principle: switch once per cycle (0,0) -> (1,0) or (0,1) -> (0,0)
- Difficulty: balance (1,0) and (0,1)
- Solution: fat fire routing (ASIC), D-WDDL (FPGA) [16]
- WDDL can increase the number of measurements required by orders of magnitude
- Suitable for FPGAs [6]
- Price to pay: area and power



FPGA related literature

- First results for power analysis on FPGAs [7]
- Evaluation of FPGA specific DPA countermeasures [8,9]
- Fault injection attacks on FPGAs [10]
- FPGA evaluation platform for power analysis resistance: SASEBO [11]



Active attacks

- Expose the device to physical stress and bypass or infer secrets

- Fault injection processes
  - Non-invasive: Glitches, EM, Temperature
  - Semi-invasive: Photons (e.g. laser)
  - Invasive: Focused Ion Beam

[courtesy: H. Handschuh]

- Less public literature than for side-channel analysis
- Non-invasive attacks have the disadvantage that one cannot target a specific part of the chip other than by timing

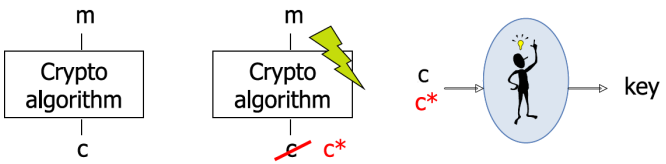
Laser Fault Injection

- Focus on semi-invasive attacks and injection of *transient* faults (SEU)
- Adversarial model given by Lemke-Rust and Paar [12]
  - p_{time} is probability to hit at the right instant
  - P_{area} is probability to hit the right spot
  - P_{volume} is probability to penetrate sufficiently deep
  - p is overall probability of *succesful* fault injection
  - Random Fault Model
    - The fault's effect cannot be controlled
- Interesting read: The Sorcerer's Apprentice Guide to Fault Attacks [13]

[www.new-wave.com]

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



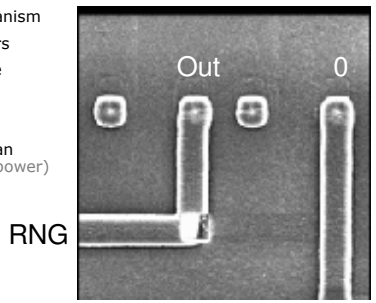
- Allows to work in the Random Fault Model

Biham, Shamir [17]

Active and Full-Invasive Attacks

- Modify circuit (worst nightmare...)
- Disconnect security mechanism
- Deactivate security sensors
- RNG stuck at a fixed value
- Reconstruct blown fuses

- Very expensive to detect an invasion (you may be off power)
- Very powerful (in-security à la carte)



[www.fa-mal.com]

Fault Attack Countermeasures

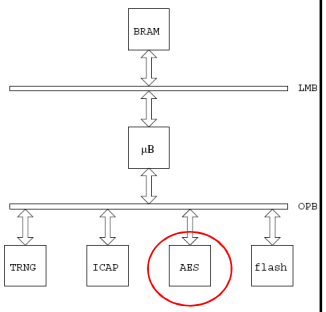
- Type 1: make it difficult to exploit a fault – do not react
  - Make it hard to inject at fault at the right time *(and/or the right spot)*
  - In software: random order execution, dummy cycles
- Type 2: detect an injected fault - react after attack
  - Add redundancy and check for errors
  - Compute twice (serial, parallel) and compare results
  - Dual-rail logic with dedicated error state
- Type 3: prevent fault injection – react during attack (not covered)
  - Secure packaging, dedicated sensors
  - For example: monitor Vdd, Clk, etc.
  - Detect light in package, detect that package is opened, etc.

Outline

- Physical attacks
  - Passive (Side-Channel) Attacks
  - Active (Fault Injection) Attacks
- Dynamic reconfiguration
  - System overview
  - AES reference architecture
- Three new countermeasures for reconfigurable devices
  - Temporal jitter
  - Spatial and temporal jitter
  - Fault detection
- Conclusions

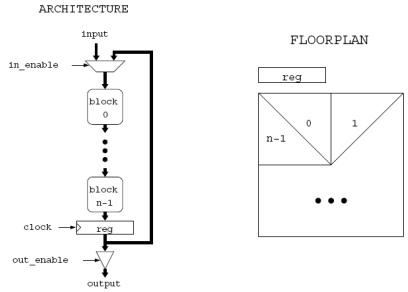
System Overview

- Platform: Xilinx Virtex-II pro
- What do we need to enable partial dynamic reconfiguration?
  - Internal Configuration Access Port (ICAP)
  - MicroBlaze (µB), code in BRAM
  - Storage for bitstreams (flash)
  - Randomness (TRNG)
  - Crypto core (AES)
- *Focus on security of crypto core*

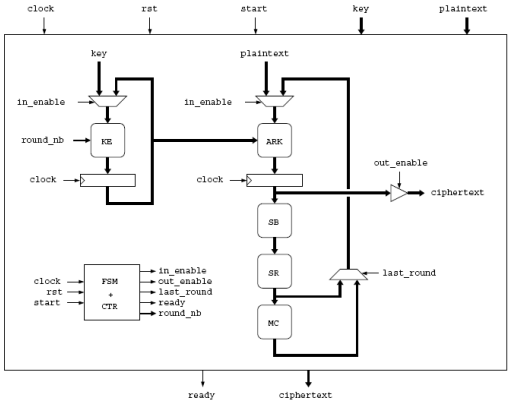


Reference Architecture

- Assumption: the cryptographic algorithm is a repetetive instruction composed of a number n of subfunctions
- This is realistic for secret key and public key crypto systems



Architecture, AES example



Outline

- Physical attacks
  - Passive (Side-Channel) Attacks
  - Active (Fault Injection) Attacks
- Dynamic reconfiguration
  - System overview
  - AES reference architecture
- Three new countermeasures for reconfigurable devices
  - Temporal jitter
  - Spatial and temporal jitter
  - Fault detection
- Conclusions

Temporal Jitter (1)

- Classical countermeasure for software implementations
  - Random process interrupts / Dummy cycles
  - Random order execution
- Differential Power Analysis
  - Need synchronized measurements
    (re-synchronization is sometimes feasible but expensive)
  - Countermeasure's effect: de-synchronized measurements; repeated invocations don't lead to the same sequence of operations over time
- Fault injection
  - Needs correct timing
  - Countermeasure's effect: hard to hit at the right moment
    (you may get a faulty output, but which operation has been faulted?)

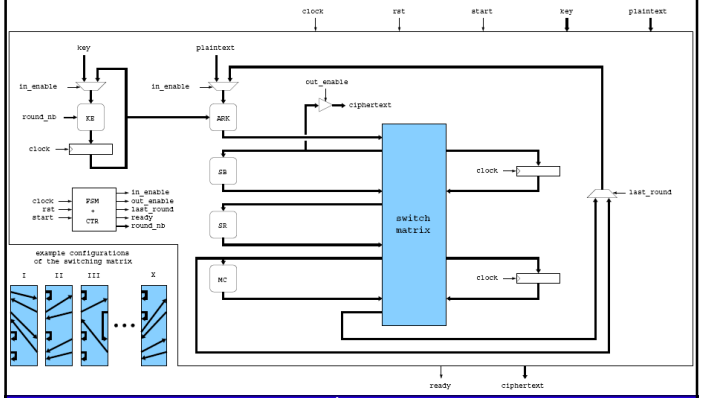
Temporal Jitter (2)

- It is hard to realize temporal jitter in hardware since hardware cannot morph
  - [8] proposes pipelining and processing of random inputs
  - Effect is not the same: sequence of operations is constant but adversary cannot predict meaningful Hamming distances
- Desired effect is to make things happen at varying time indexes
- Can we make morphing hardware? Yes, dynamic reconfiguration!
- What causes delay in hardware?
  - A register causes a delay of one clock cycle
  - Propagation delay in logic (more complex logic more delay)
    (For simplicity we disregard delays due to wires)

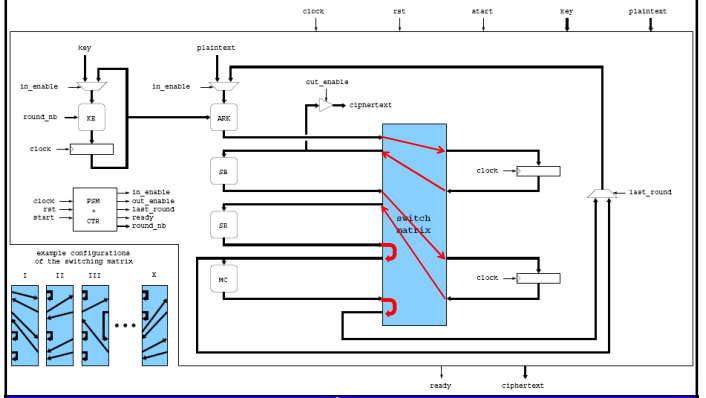
Temporal Jitter (3)

- Dynamically change position(s) of register(s)
- Dynamically re-order functional blocks if applicable
- Naïve approach
  - Multiple implementations of the entire architecture
  - Expensive in terms of bitstream size and reconfiguration time
- Better approach
  - Change only the wiring of functional blocks and registers
  - Reconfigurable central switch matrix
  - Advantage: smaller bitstreams, faster reconfiguration

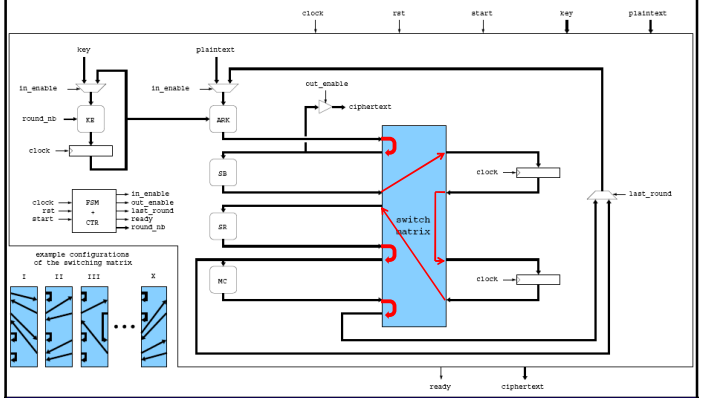
Temporal Jitter, AES example (1)



Temporal Jitter, AES example (1)



Temporal Jitter, AES example (1)



Temporal Jitter, AES example (2)

- # of configurations depends on # of functional blocks (n) and # of registers (m)
  - Increases if we allow *cascaded registers* between functional blocks
- In the example: n = 4 blocks, m = 2 registers (cascading allowed)
- Number of *distinct* configurations $c = \binom{n+m-1}{m} = \binom{4+2-1}{2} = 10$
- Active configuration determined by *secure* TRNG
- Probability to observe a given configuration is 1/c
- Number of temporal shifts is bounded above by c
- Bottlenecks
  - More configuration options require more bitstreams and memory
  - More registers increase processing time

Temporal Jitter, performance

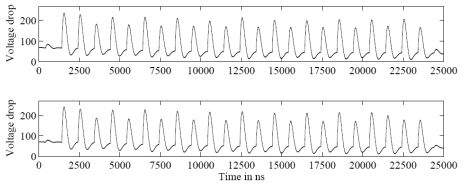
- Implementation results on Virtex-II pro
- Static design with 1 register, dynamic design with 2 registers

	occupied area (# slices)	max. clock frequency (MHz)	through-put (Gbit/s)	reconf. time (ms)	reconf. data size (kB)	# conf. options
Static design	685(5%)	111	1.3			1
Prototype: static/dynamic	3251(23%) 1547(11%)/1704(12%)	33	0.2	3	91	10

- Reconfiguration time 3ms, modern FPGAs at least 10x faster
- Max. clock frequency decreases due to communication between static and dynamic part
- Prototype's static part is larger than fully static design due to additional register and communication logic

Temporal Jitter, Security

- Obvious approach to circumvent the countermeasure is to distinguish the different configurations
- Can Timing Analysis [14] distinguish configurations?
  - No, execution time is constant 11 * m clock cycles
- Can Simple Power Analysis [1] distinguish configurations?
  - No, but we have to pre-load the intermediate registers with random data to hide their position



Config. 1

Config. 2

Temporal Jitter, DPA resistance

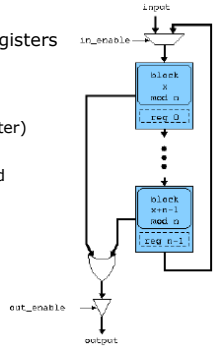
- Mangard studied the effect of temporal jitter as a countermeasure against standard DPA [15]
- Number S of measurements needed to break the implementation $S = 3 + 8\left(\frac{Z_\alpha}{\ln(\frac{1+\rho' \cdot \hat{p}}{1-\rho' \cdot \hat{p}})}\right)$
- Z_α is a confidence interval
- p' is the correlation coefficient at an unprotected implementation
- $\hat{p}$ is the probability that a certain temporal jitter occurs
- We can interchange the order of SB and SR, thus c=20
- But only 8 different temporal shifts, worst case $\hat{p}$ =6/20
- Under conservative assumptions, S increased by ~5

Temporal Jitter, Fault resistance

- Conservative assumption: $p_{volume} = 1$, $p_{area} = 1$
- 8 different temporal shifts, worst case $p_{time} = 6/20$
- Probability to inject a fault at the right spot at the right time $p = p_{volume} \times p_{area} \times p_{time} = 0.3$
- Adversary cannot distingush (non-)successful fault injections
- Some cryptanalytical methods require several *succesful* fault injections and may sieve out the correct key if input data is bad
- Fault on the switch matrix remains only until reconfiguration
- Fault on functional blocks may remain until reset
  - But exploitable modification highly unlikely in random fault model
- Functional blocks can be further protected

Spatial and Temporal Jitter (1)

- Desired effect is to make things happen at varying *locations and* varying *time indexes*
- Dynamically re-locate functional blocks and registers on chip area (temporal jitter as before)
- Our approach
  - Prepare multiple bitstreams (with/without register) per functional block
  - TRNG determines *which* of them is implemented *in which* reconfigurable region



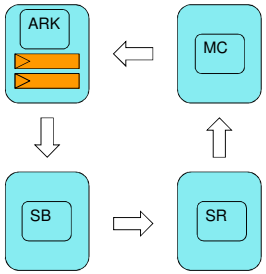
Spatial and Temporal Jitter (2)

- Suppose we have n functional *blocks* with fixed order
- One block can be mapped to one out of *n areas* n options
- Countermeasure aims at preventing local fault injection e.g. laser fault injection
- Probability to inject a fault at the right spot $p_{area} = 1/n$
- Suppose each block can be followed by at most one register
- Probability to inject a fault at the right time p_{time} bounded below by 1/(m+1)

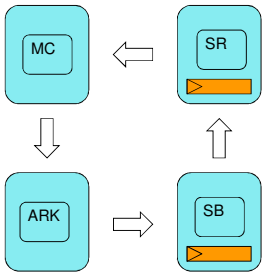
Spatial and Temporal Jitter, AES example (1)

- n = 4 functional blocks 4 reconfigurable areas
- Design contains m = 2 registers
- For each functional block 2 bitstreams (with/without register)
- We could interchange the order of SB and SR to get more options
- Bottlenecks:
  - Significantly higher reconfiguration time due to more regions
  - More bitstreams need to be stored

Temporal Jitter, AES example (2)



Temporal Jitter, AES example (2)



Spatial and Temporal Jitter, Security

- Resistance against local fault injection processes:
  - $p_{volume} = 1$
  - $p_{area} = 1/4$
  - $p_{time} = 1/3$
  - $p = p_{volume} \times p_{area} \times p_{time} = 1 \times 1/4 \times 1/3 = 1/12$
- Side effect: since all functional blocks are implemented in reconfigurable regions, the entire circuit can recover from transient faults
- Tradeoff security vs. reconfiguration delay should be decided having fault injection frequency in mind (laser typically < 50Hz)

Fault Detection

- Can be realized in FPGAs by reading back bitstream(s) and comparing them with the reference copy
- Comparison can be done via protected logic, CRC check, etc. inside the FPGA and at runtime
- Procedure detects faults only if reference bitstream cannot be altered in the same way (highly unlikely in the random fault model)
- Scheme can be complemented with traditional fault detection mechanisms
  - Dual-rail with error state, execute twice and compare results, etc.
- Designer's choice how to react to an alarm signal
- *Note: for some attacks checking after outputting is already too late!*

Conclusions

- Power and Fault Analysis Resistance in Hardware through Dynamic Reconfiguration
- Temporal Jitter can be generated by changing the location of registers between functional blocks and by re-ordering blocks
- Spatial Jitter can be generated by re-locating functional blocks on the chip area at runtime
- Both types of jitter can be combined
- Fault detection with negligible area overhead
- Bottlenecks in general:
  - Reconfiguration time
  - Memory needed for bitstreams

Thank you for your attention!

Questions?

Part of this talk is based on

Mentens, Gierlichs, Verbauwhede: Power and fault analysis resistance in hardware through dynamic reconfiguration, CHES 2008

References (1)

[1] Kocher, Jaffe, Jun: Differential Power Analysis, Crypto 1999

[2] Aigner, Oswald: Power Analysis Tutorial

[3] Brier, Clavier, Olivier: Correlation Power Analysis, CHES 2004

[4] Tillich, Herbst, Mangard: Protecting AES Software Implementations on 32-bit Processors…, ACNS 2007

[5] Clavier, Coron, Dabbous: DPA in the presence of hardware countermeasures, CHES 2000

[6] Tiri, Verbauwhede: A Logic Level Design Methodology for a secure DPA Resistant ASIC or FPGA Implementation, DATE 2004

[7] Örs, Oswald, Preneel: Power-analysis attacks on an FPGA – First experimental results [CHES 2003]

[8] Standaert, Mace, Peeters, Quisquater: Updates on the securityof FPGAs against Power Analysis Attacks, ARC 2006

References (2)

[9] Standaert, Örs, Preneel: Power Analysis Attack on an FPGA Implementation of Rijndael, CHES 2004

[10] Maingot, Ferron, Leveugle, Pouget, Douin. Configuration errors analysis in SRAM-based FPGAs, Microelectronics Reliability 2007

[11] SASEBO: http://www.rcis.aist.go.jp/special/SASEBO

[12] Lemke-Rust, Paar: An Adversarial Model for Fault Analysis Against Low-Cost Cryptographic Devices, FDTC 2006

[13] Bar-El, Choukri, Naccache, Tunstall, Whelan: The Sorcerer's Apprentice Guide to Fault Attacks, IEEE Vol 94, Issue 2, 2006

[14] Kocher: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and other systems, Crypto 1996

[15] Mangard: Hardware Countermeasures against DPA, CT-RSA 2004

[16] Yu, Schaumont: Secure FPGA circuits using..., CODES+ISSS 2007

References (3)

[17] Biham, Shamir: Differential Fault Analysis of Secret Key Cryptosystems, Crypto 1997