Power Analysis Resistant AES Implementation with Instruction Set Extensions

Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2007

Stefan Tillich **and Johann Großschädl**

IAIK – Graz University of Technology
Stefan.Tillich@iaik.tugraz.at
www.iaik.tugraz.at

Outline

- ## The Setup: Crypto Extensions for AES

- ## The Goal: Implementation Security

- ## Results so Far: Software Countermeasures

- ## This Work: Hardware Countermeasures

  - ### 3 Proposed Approaches

  - ### Security and performance analysis

- ## Conclusions

Motivation

- Many proposals for cryptography enhancements of general-purpose processors

- Impact on performance and area well studied

- But: Implementation security still a topic

- Public-key crypto extensions
  - -> Use SCA countermeasures for software

- Secret-key crypto extensions
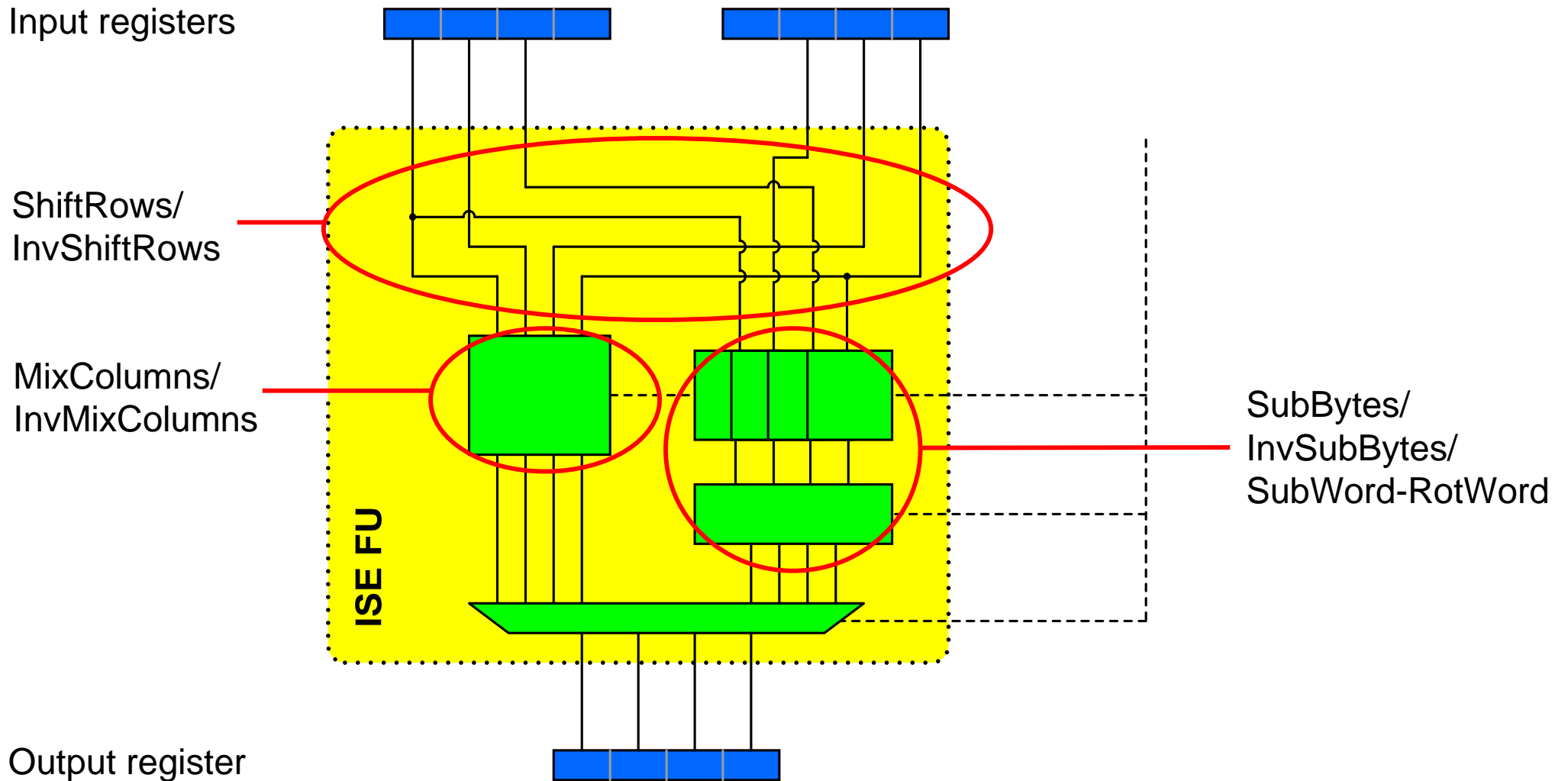 - Largely still an open problem

Previous Work

- ## Focusing on implementation security with instruction set extensions (ISEs) for AES

- ## Adaptation of software countermeasures for the use of ISEs

  - Masking

  - "Randomization": Operation shuffling & Dummy operations

- ## Significant performance loss compared to unprotected implementation

  - Security ~250x <-> Overhead ~19x

  - Security ~10^4x <-> Overhead ~100x

This Work

- ## Design of hardware countermeasures for ISE application

  - Increase implementation security & performance

  - Hardware overhead should be tolerable

  - Should be easy to implement

- ## We propose three options

  - One pure-hardware solution

  - One pure-software solution

  - One hardware/software solution

The Analyzed AES Extensions

Input registers

ShiftRows/
InvShiftRows

MixColumns/
InvMixColumns

ISE FU

SubBytes/
InvSubBytes/
SubWord-RotWord

Output register

First Basic Step

- General method for reducing PA-signal leakage

- Goal: Reduce the impact, that a critical intermediate value can have on the power consumption

- Prevent unnecessary propagation of such values (also good for lowering power consumption)
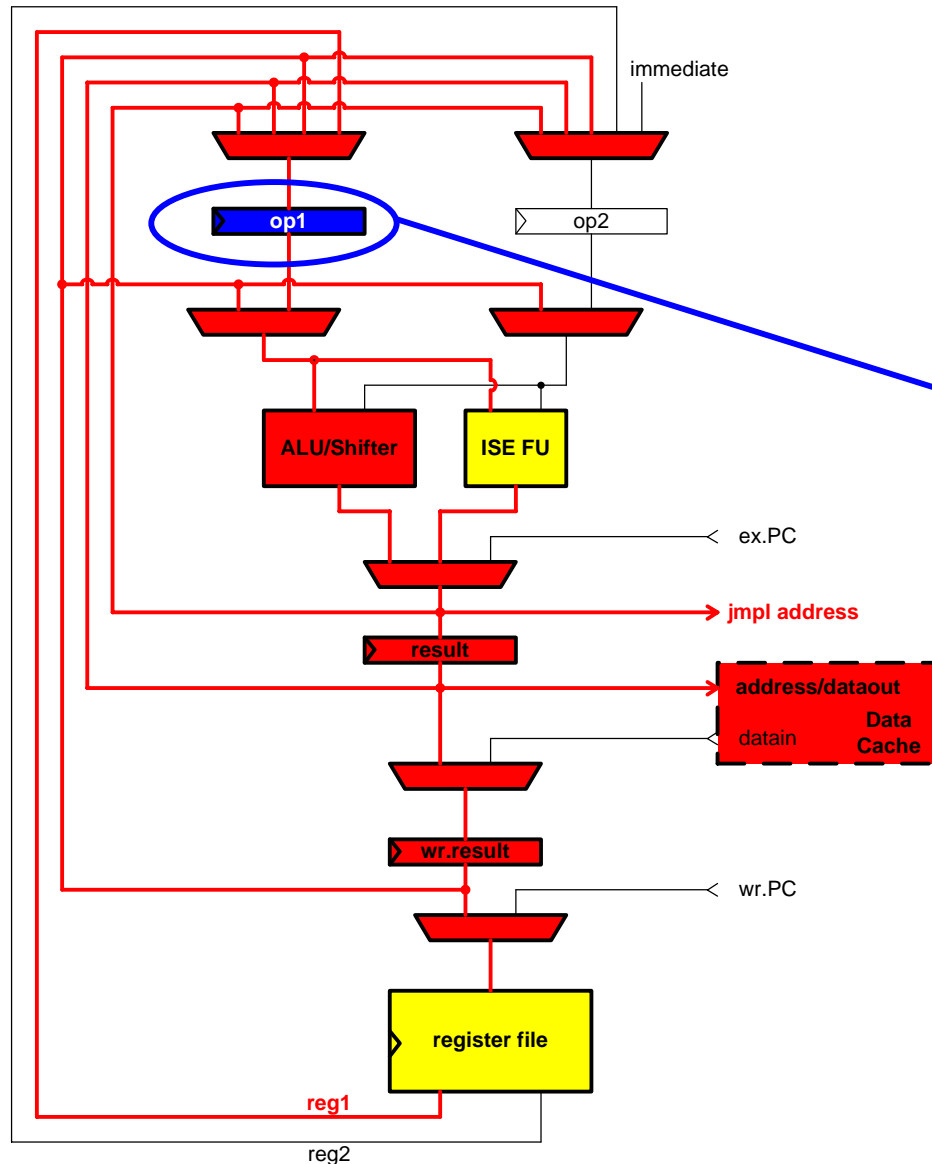
Generic 4-Stage Pipeline
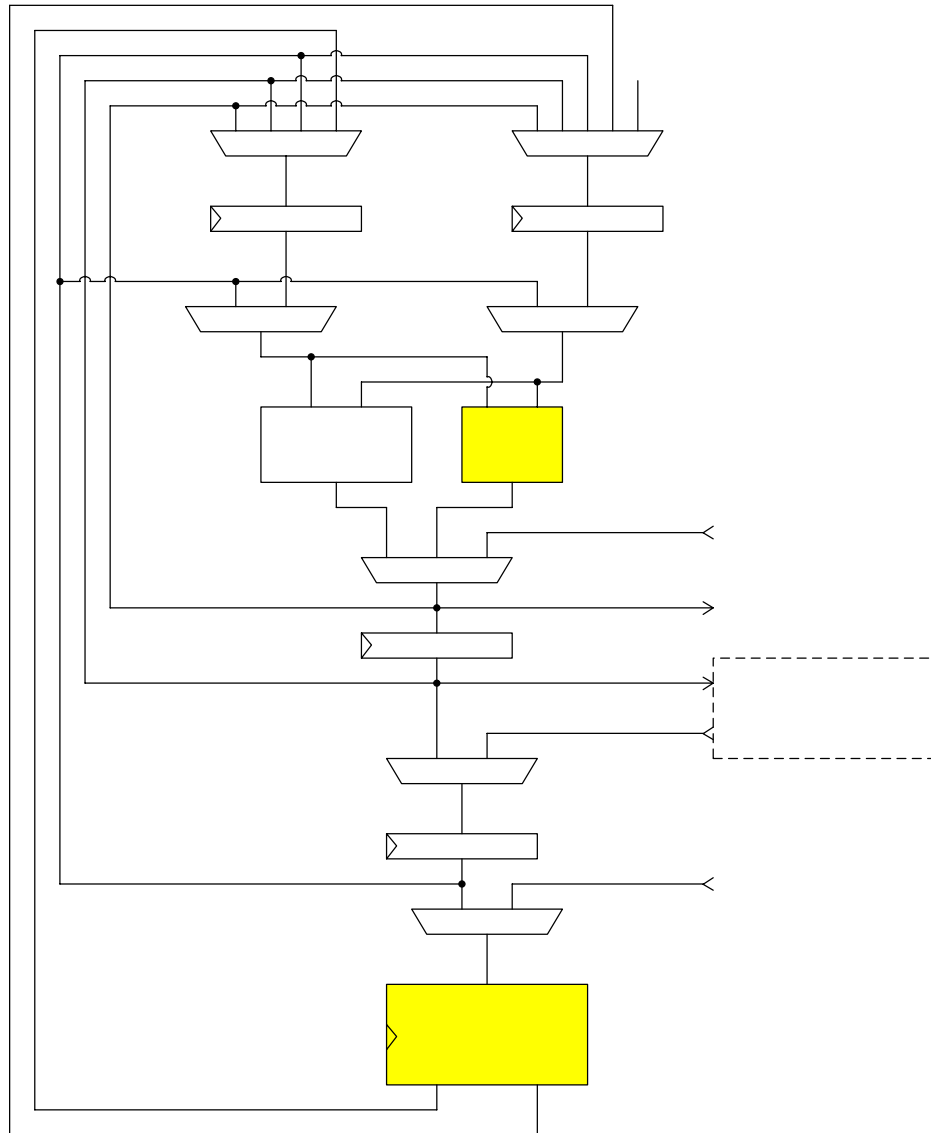
- **Simplified structure of a 4-stage pipeline of an embedded RISC processor**

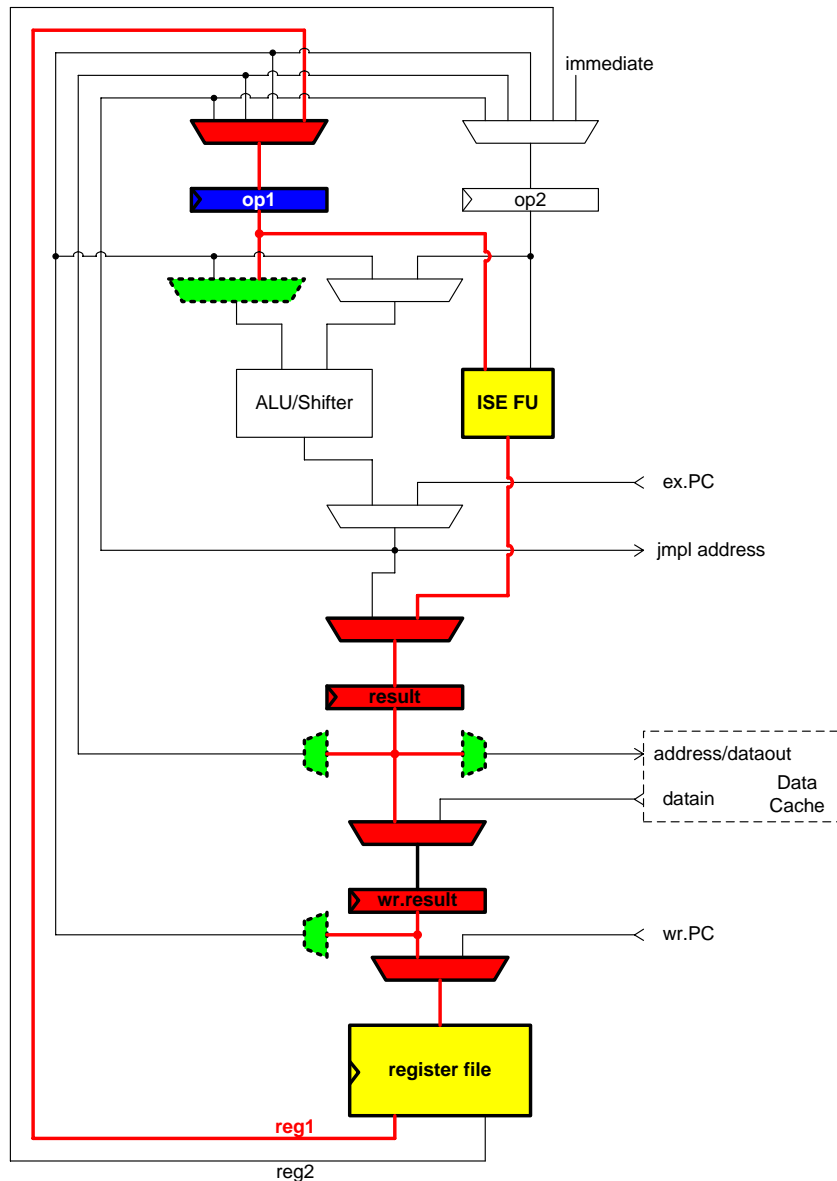- **The "real" action happens in the ISE FU and regfile (yellow)**

op1

Potential Vulnerabilities

- Potential impact of a critical value in op1 (blue) on the components of the pipeline (red)

Limit Unnecessary Activity

- **AES can be implemented without the use of feedback paths**

- **Block unnecessary propagation of critical value at multiplexors (green)**

Option 1: Complete Datapath in Secure Logic

- **Idea: Implement all parts of the datapath affected by a critical value in a secure logic style**
  - E.g. Wave Dynamic Differential Logic (WDDL)

- **Generally applicable for all types of ISEs**
  - Critical value must not leave secured datapath

- **Register file must be included**
  - -> Large overhead in area

Option 2: Random Precharging

- **Idea: Change leakage from Hamming distance with a (potentially known) value to Hamming distance with an unknown (random) value**

- **Charge datapath before and after processing of a critical value with random data**

- **Can be implemented in software**
  - Prefix and suffix each critical instruction with the same instruction using random operands

Option 3: Protected Mask Unit

- Idea: Split processor in a (small) secure zone and an insecure zone (containing all the rest)

- Critical data in insecure zone protected with a Boolean mask

- Secure zone implemented in secure logic style
  - Contains
  - **mask storage**,
  - **mask generator**, and
  - **functional units** for ISEs

Protected Mask Unit
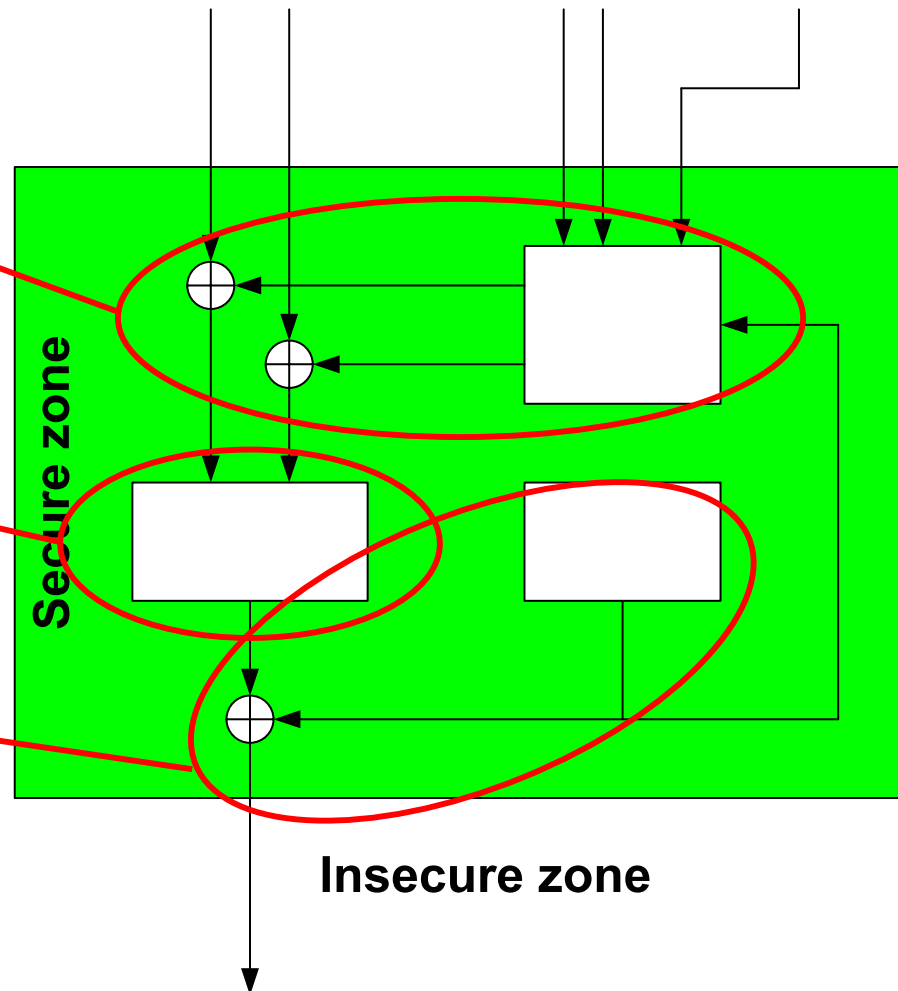

Input operands
(masked)

Operand addresses

Operands
unmasked

Mask storage
can hold seven
32-bit masks

Operation
performed

Mask generator
can produce 32
random bits/cycles

Result
masked
(fresh mask)

Secure zone

Insecure zone

Masked result

Properties

- ## Critical values remain masked in insecure zone

- ## Masks never leave secure zone!

- ## All critical operation performed in secure zone

  - ### Except AddRoundKey: $p_m \oplus k = (p \oplus k)_m$

- ## Simple interface between zones

- ## Insecure zone can be left unchanged

- ## Small processing overhead

  - ### Masking of plaintext and unmasking of ciphertext

  - ### Overhead of instructions in secure zone (e.g. extra cycle for WDDL precharging)

Security and Performance Analysis

- As example of a secure logic style, we used WDDL

- SPARC V8 (Leon2) in UMC 0.13 µm standard-cell library

- (Cycle counts given for one AES-128 encryption)

Applicability & Implementation Complexity

- ## Complete datapath in secure logic
  - **+** Applicable to all kinds of ISEs
  - **–** Careful partitioning of processor required
  - **–** Software needs to restrict critical operations to secure datapath

- ## Random precharging
  - **+** Also generally applicable
  - **+** Pure-software solution -> flexible
  - **–** Generation/management of random values might get compliated

- ## Protected mask unit
  - **+** Relatively easy implementation (simple, well-defined interface)
  - **+** Software can stay largely unchanged
  - **–** Only if critical operations can be limited to secure zone

Security

- ## Complete datapath in secure logic & Protected mask unit

  - Depends solely on security of chosen logic style
  - For WDDL > 750x improvement

- ## Random precharging

  - Empirical evaluation on an FPGA board
  - Comparison of unprotected and protected AES implementation
  - Protection factor ~ 26x

Performance

Implementation	Cycle count	Overhead
Baseline implementation (unprotected)	196	-
Complete datapath in secure logic	392	100 %
Random precharging	~ 400	~ 105 %
Protected mask unit	~ 230	~ 17 %

- Note: WDDL requires an extra precharge cycle / operation
- Pure SW performance: 1,637 cycles / block

Area & Delay Overhead

Implementation	Silicon Area (GEs)	Critical Path (ns)
Complete datapath in secure logic	$+ 20,500 + 940 \cdot R$	+ 0.8 ns
Random precharging	none	none
Protected mask unit	+ 28,000	+ 1.0 ns

- WDDL area overhead: ~ 3.5
- WDDL critical path overhead: ~ 1.2
- R denotes number of secured registers
- Original critical path: 4 ns

Combination with Other Countermeasures

- **Complete datapath in secure logic**
  - Can be fully combined with software countermeasures

- **Random precharging & Protected mask unit**
  - Can be combined with software countermeasures to a certain degree (depends on used secure logic style)
  - E.g. shuffling of operations & dummy operations

Conclusions

- Investigation of three approaches to increase power analysis resilience of AES software implementations

- On 32-bit RISC processors with cryptography extensions

Conclusions

- **Complete datapath in secure logic**
  - Can be fully combined with software countermeasures
  - Generic & secure
  - High implementation cost

- **Random precharging**
  - Flexible & cheap
  - Relatively low security

- **Protected mask unit**
  - Generic, secure, fast, simple to implement
  - Moderate implementation cost

Acknowledgements

We would like to thank
Stefan Mangard and **Dan Page**
for their support.

Thank you for your attention!

See you at CHES07 dinner!

© IMaGE Performing Arts Promotion Veranstaltungsorganisations GmbH
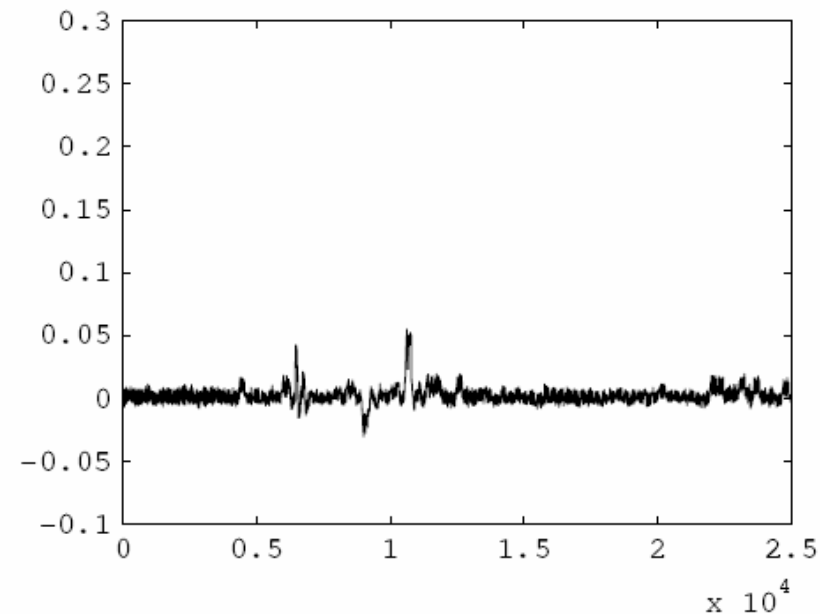
Details on Random Precharging

- **All concerned instructions produce a uniformly distributed random result when supplied with two uniformly distributed operands**

- **For "standard" DPA, it suffices to protect 20 instructions at the beginning and end of AES encryption**

  - 336 random bytes required in total

Security Evaluation of Random Precharging

- Maximum correlation reduced from 0.284 to 0.055
- Correlation reduced by 5.16 -> Number of traces increased by $5.16^2 = 26.6$
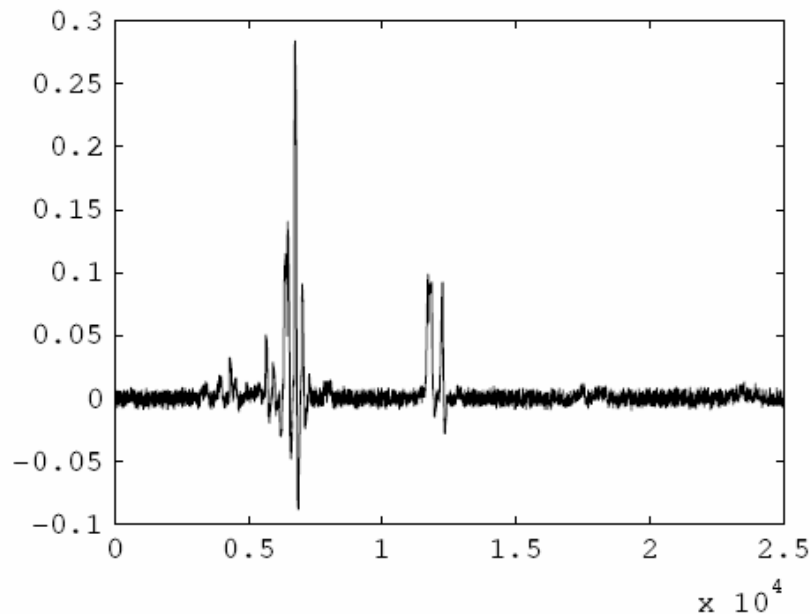
Fig. 5. Result of DPA attack on unprotected AES implementation

Fig. 6. Result of DPA attack on AES implementation with random precharging

What to Protect?

- Up to and including SubBytes for round 2

- From SubBytes of round 9

- At least 2^{40} hypotheses / key byte

- At 1,000 hypotheses / second: > 34 years