Aspects of the Development of Secure and Fault-Resistant Hardware

Wieland Fischer
FDTC 2008
10. August 2008

Introduction

I would like to give…

- … an overview over the problems and hardships during the development of secure hardware like smart card controllers.
  - Many attacks have to be defeated, and just implementing all the proposed countermeasures provided by the scientific literature is often not possible – e.g. for economical reasons.
  - So the developer of secure hardware has to carefully choose the methods he is implementing.

- … an impression of what the developer has to do in order to make the right decision on the countermeasures he is going to implement.

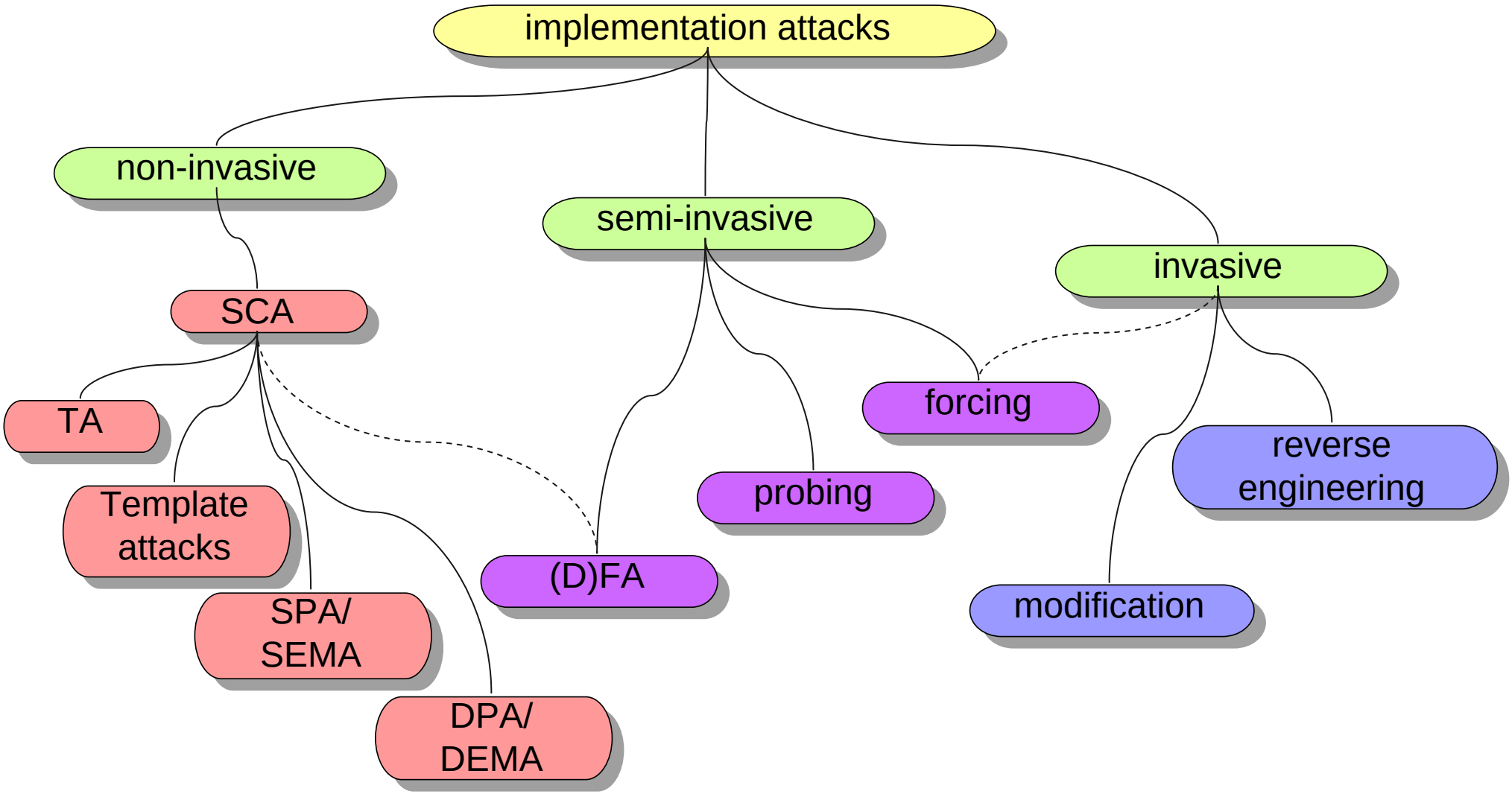
- … a hint of the types of questions that have to be solved to help the designer.

Outline

1. Attacks – The main requirements for secure hardware
2. Reality – additional problems, constraints, and requirements
3. A Security Metric – Common Criteria
4. Application to secure hardware development with respect to fault attacks

1. Attacks – The main requirements for secure hardware
2. Reality – additional problems, constraints, and requirements
3. A Security Metric – Common Criteria
4. Application to secure hardware development with respect to fault attacks

Attacks – Taxonomy



Attacks – Methods of Fault Induction

- Spikes and Glitches
  penetration of/on power supply, clock, or IO-signals.
- Light
  flash light, laser, uv light.
- Ionizing radiation
  alpha particles, focused ion beam (FIB), X-ray.
- Temperature, Voltage, Frequency variation
  running the chip out of the specified operating range to trigger a faulty behavior.
- Forcing
  forcing signals with a probing needle.

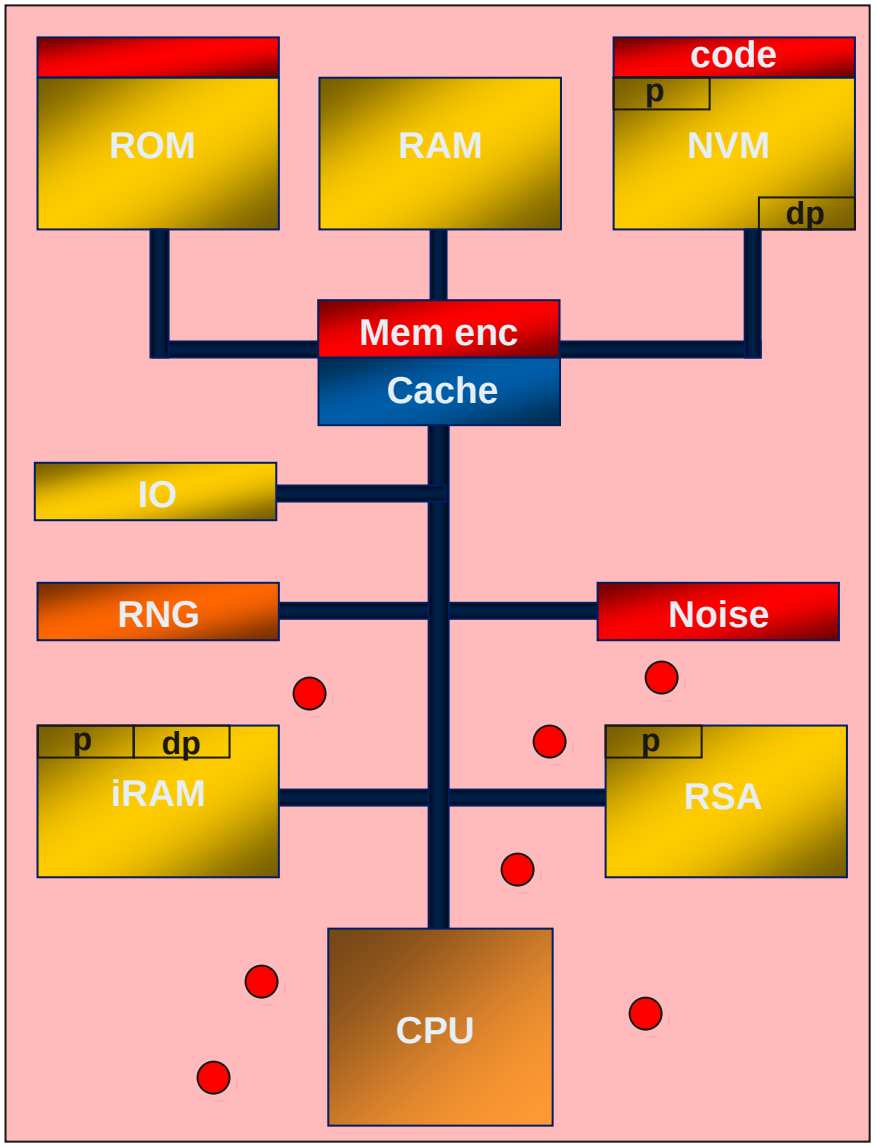
- Distinguish between:

global attacks ←→ local attacks
(whole chip is affected) (only small areas are affected)

cheap mechanisms ←→ expensive mechanisms.

Example for Attacks: RSA/CRT

1. probe NVM → memory encryption
2. change contents of NVM → codes
3. change logic (fib) → self test software
4. probe busses for plain secrets → shield
5. forcing of busses → codes
6. induce faults into plain data → algorithm
7. induce faults into RSA computation → algorithm
8. DPA → randomization
9. template attack/SPA → random noise
10. forcing/probing random data → shield, testing random data
11. induce errors in checks (glitch,flash) → sensors



Combined Countermeasures

- In some cases, countermeasures against different attacks can be combined:
  - Randomization works against DPA and timing analysis.
  - Random noise production works against template attacks and SPA – at least up to some extend.
  - Countermeasures against DPA and DFA for RSA with CRT work hand in hand quite well.

- However, sometimes this does not work so well:
  - Simple masking of signals only works against *either* DPA *or* probing: *DPA and probing are just two sides of the same medal*.

- Unfortunately, combined countermeasures against most of the known attacks are still rare.

1. Attacks – The main requirements for secure hardware
2. Reality – additional problems, constraints, and requirements
3. A Security Metric – Common Criteria
4. Application to secure hardware development with respect to fault attacks

Additional Constraints I

- Area

In high volume production of chips:
chip area ~ production costs per piece.
→ The countermeasures must not exceed a certain area in order to make the product profitable.

- Power or current consumption

Many devices (SIM cards, contactless cards) have strong restriction concerning power consumption.
→ The countermeasures must not add too much additional power consumption, in order to lie within the specified ranges of operation.

Additional Constraints II

- Forward Security

Being prepared for future, yet unknown, attacks! What happens, if a new attack becomes known

- during a late phase of the design process?
- after design, qualification or even shipment of the product?

→ Ideas for fall back solutions have to be evaluated.

- If some security features rely mainly on software, then it is still possible to adapt the software: E.g. RSA algorithms implemented on an arithmetic co-processor. In some cases this might even be possible in the field.
- If some security feature relies mainly on built in hardware mechanism, then such a case could be devastating if this happens. Therefore some fall back solutions, maybe taken over by the software are desirable.

Additional Constraints III

- Universality

Hardware manufacturer seldom deliver whole solutions. Therefore security relevant basics, like crypto-co-processors (AES accelerator) or libraries (RSA software on arithmetic co-processors) have to be secured without the context they might be used.
→ Countermeasures might have to be built in at sub-optimal levels.

  - DPA and fault attacks even on an unsecured AES-accelerator or software will not be successful if the encryption key changes after every usage. So in certain situations one might choose some key update protocol, like hashing the key after each en-/decryption. But this is only possible in certain systems where all parties can keep track of the update-procedures. The hardware provider can not rely on the fact, that the AES might only be used in the described situation.

Additional Constraints IV

- One hardware for many applications

Since it is not economical to build separate hardware for every application or customer, one chip design is used for many purposes with small changes (like individual ROM by changing one metal layer)
→ The hardware, peripherals, and coprocessors have to fulfill many different requirements, building a good compromise for sometimes contradicting requirements.

- Patent Situation

The Developer will try to avoid implementing countermeasures that are patented by a third party. But maybe some countermeasure is so good that, e.g., the saving in area, is worth the license fees then he might choose it.
→ For making this decision he has to have a good estimation of these savings and costs.

Additional Constraints V

- Early decision on countermeasures during concept phase

Since production costs for sets of masks are growing with shrinking technology, the designer has almost no possibility to test several variants of countermeasures on silicon. The decision for a certain design has to be done quite early and with a high confidence level.
→ Decisions have to be made on simulation basis

- Judging the Security Level

Since area & power consumption are the biggest constraints for an economical design, the designer has to be able to value the security level of the final product at time of concept.
→ The better the later security level can be judged, the closer the countermeasure can be designed with respect to a certain minimum level.

We have seen: The question which have to be answered at an early step of the design are numerous and of growing importance. Without a clear judgment of the quality of built-in countermeasures, no high volume security chip can be produced.

Security vs. Certifiability

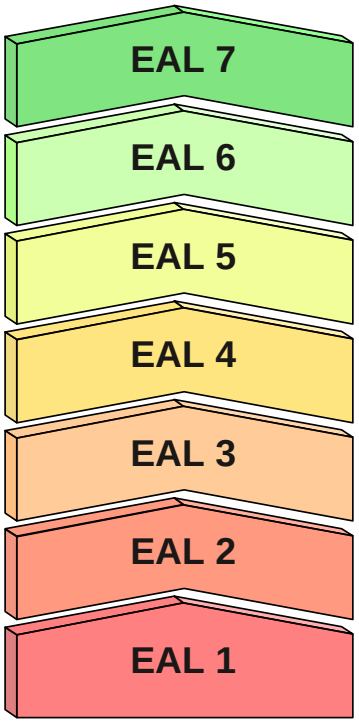
- Our chip designer needs a single metric that tells him the overall security level of the final product, he is designing.
- Unfortunately, this does not exist and probably will not exist in the near future.
- At the present time, there are attempts in single fields to develop these kind of metrics. Veritable approaches already exist. But up to now, there are no practical methods or even tools that make this feasible.

- Distinguishing between different “security levels” is also a common problem in certification processes. There, some kinds of practical metrics were define.
- One, widely used process with such a very pragmatic metric is:

Common Criteria

1. Attacks – The main requirements for secure hardware
2. Reality – additional problems, constraints, and requirements
3. A Security Metric – Common Criteria
4. Application to secure hardware development with respect to fault attacks

Common Criteria for Information Technology Security Evaluation



Formally verified design and tested

Semiformally verified design and tested

Semiformally designed and tested

Methodically designed, tested and reviewed

Methodically tested and checked

Structurally tested

Functionally tested

EAL x { (high) (medium) (basic)

The resulting *security level* is stated after the *evaluation depth*.

Further Information also on:
www.commoncriteriaportal.org

1. For the certification, a protection profile has to be defined which lists the attacks against the product (TOE) is claimed to be immune.
2. Then the evaluator rates the attack potential for single attacks by the following method:

Rating of Attack Potential

- Two parts of the attack will be rated differently:

The identification part of an attack

How big is the effort to identify the vulnerability, build up and set up a certain attack for demonstration?

The exploitation part of an attack

How big is the effort to run the attack on a second device, using the results from the former part?

Rating Factors 1

- Elapsed time:
  How much time is necessary for each of the two parts?

Rating Factors 2

- Expertise:
  How much general knowledge does the attacker need for each of the two parts?

  - Layman
    (no particular expertise needed)
  - Proficient
    (familiar with security behavior, classical attacks)
  - Expert
    (familiar with developers knowledge, algorithms, protocols, hw structure, principles and concepts of security, and techniques and tools for definition of new attacks)

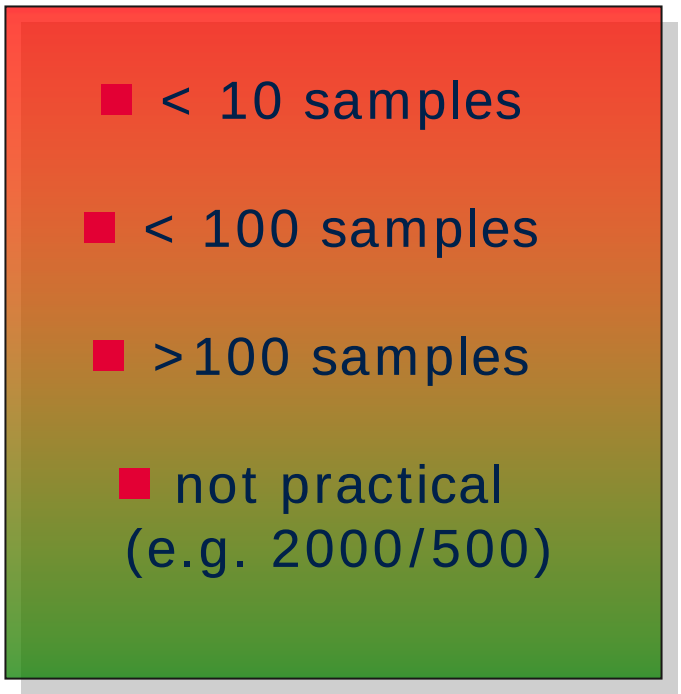
Rating Factors 3

- Knowledge of the TOE:
  How much knowledge about the TOE does the attacker need for each of the two parts?

  - Public Knowledge
  - Restricted Knowledge
    (functional specifications, guidance documentation)
  - Sensitive Knowledge
  - Critical Knowledge
    (Implementation representation, design or source code)

Rating Factors 4

- Access to TOE:
  Availability of samples (time and cost) as well as number of samples needed to carry out an attack path.



Rating Factors 5

- Equipment:
  Expenses and availability of the equipment needed in order to carry out an attack path.

- None

- Standard
  (laser, uv-light emitter, climate camber, voltage supply, analogue oscilloscope, chip card reader, PC, signal generation and analysis sw)

- Specialized
  (university equipment, visible/uv-light microscope, micro probe workstation, laser cutter, digital oscilloscope, signal analyzer, tools for chemical edging and grinding)

- Bespoke
  (expensive tools, tools difficult to keep confidential)

Rating Factors

	Factors	Identification	Exploitation
Elapsed time	< one hour	0	0
	< one day	1	3
	< one week	2	4
	< one month	3	6
	> one month	5	8
	Not practical	*	*
Expertise	Layman	0	0
	Proficient	2	2
	Expert	5	4
Knowledge of TOE	Public	0	0
	Restricted	2	2
	Sensitive	4	3
	Critical	6	5
Access to TOE	< 10 Samples	0	0
	< 100 Samples	2	4
	> 100 Samples	3	6
	Not practical	*	*
Equipment	None	0	0
	Standard	1	2
	Specialized	3	4
	Bespoke	5	6

Evaluating the Security Level

- All the rating factors are added up for one particular attack.
- For EAL high rating, no attack defined in the protection profile must have a rating below 31.

Range of values	
0-15	No rating
16-24	Basic
25-30	Medium
31-	High

Consequences

- Although “outlawed” in the crypto community:
  Up to some extend *Security by Obscurity* is supported.

- For the hardware manufacturer:
  Parts of the TOE may not be reused identically in a less secure device, since there analysis or reverse engineering of the countermeasures might be easier.

- For the designer:
  First try to identify the rating factors which can not be improved, and then work on the other factors.

- For development of fault attack countermeasures:
  The elapsed-time part of the rating table shows, how long the TOE must be able to withstand a particular fault attack. Design the counter measure according to these numbers!

Consequences

The easier an attack is (cheap equipment, no additional knowledge), the more time should be necessary for a successful fault attack.

- Example: Bellcore attack on RSA with CRT.
  - One undetected fault might be enough to factorize the modulus.
  - Cheap attack with flashlight, since fault can be induced almost at any time and anywhere.
  - → The system must have a high *error detection probability*!($\sim 1-2^{-24}$)

- Example: AES, some other block cipher.
  - Several faults may be needed.
  - Faults have to be placed more precisely.
  - → A lower *error detection probability* might be enough.

Main task:
Computation of the **error detection probability** in advance.

1. Attacks – The main requirements for secure hardware
2. Reality – additional problems, constraints, and requirements
3. A Security Metric – Common Criteria
4. Application to secure hardware development with respect to fault attacks

Methods for the Evaluation of the Error Detection Probability of a Certain Countermeasure

- Testing on Silicon
  - too late
    - only usable for final verification
- Theoretical computation of error detection probability
  - Not clear, whether the theoretical values are really reliable
    - Could be tested by simulation, but only in some situations.
- Simulation of fault attacks and making statistics
  - If the error probability is too big ($1-2^{-24}$), simulations will take too long or are even infeasible.
    - use scaling methods, if possible
  - Difficult to describe the exact behavior of the silicon under an attack
  - Most attacks are of statistical nature: Not every fault induction manifests in an error.
    - Usage of fault models, i.e., a statistical description of these attacks on digital level.

Conclusion

The Designer of secure hardware needs …

- … the ability to evaluate the security level of the later product quite early!
  There is need for methods and reliable tools to support this, like DPA-testing on simulation basis or evaluation of error detection probability.

- … more universal countermeasures that work for many different types of attack at the same time – if they are better than all the individual countermeasures added up.
  At best, these countermeasures respect that the different fault induction mechanisms must be fought with different strength.

- … concepts for global protection of a whole chip throughout the complete data path.

Thank you for your attention!