Penetration Testing for Certification: What we care about

Marcel Medwed
marcel.medwed@nxp.com

Outline

- Common Criteria
- Evaluation Assurance Levels
- JHAS
- Penetration Testing and Countermeasures

Common Criteria

Ideas

- General framework to certify the security of IT products by a third party

- Involved parties
  - Manufacturer / Sponsor
  - Evaluation Facility
  - Overseer (Certification Facility)

- Common Criteria Recognition Arrangement

- Asset-centered approach

Framework

- Common Criteria
  - Part 1: Introduction and General Model
  - Part 2: Security Functional Requirements
  - Part 3: Security Assurance Requirements
  - Common Criteria Evaluation Methodology

Security Evaluation

Common Criteria – Subjects to be addressed

Threats

- confidentiality
- integrity

Environment

- Development
- Production
- End usage

Assets

Security Functional Requirements (What?)

Security Assurance Requirements (How?)

Target of Evaluation

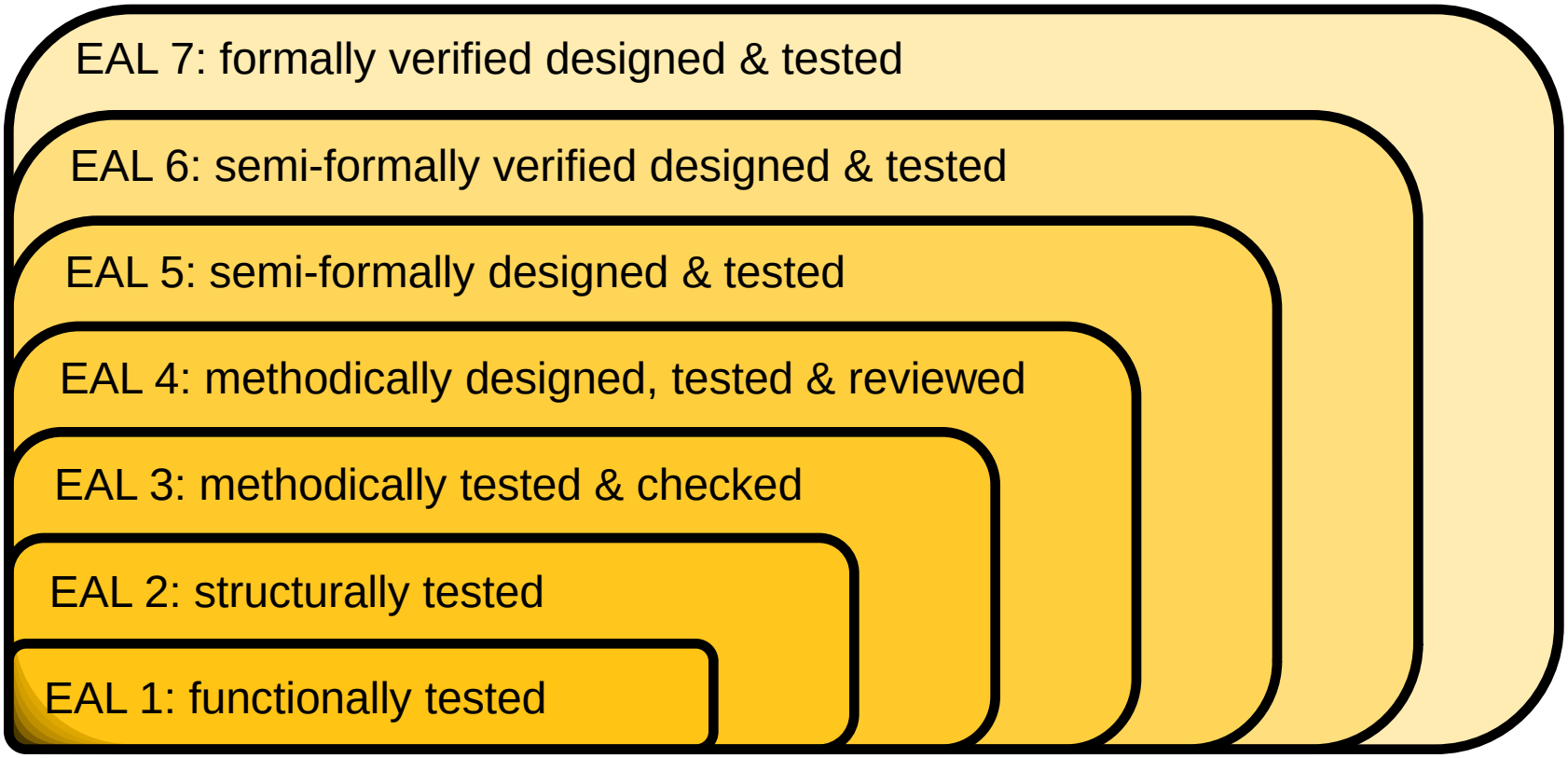
TOE's Security Functions

- correct operation
- memory integrity
- SPA/DPA resistance
- cryptographic support
- resistance against physical attacks

Basis for evaluation

Protection Profile → **Security Target**

Evaluation Assurance Levels (1)



Evaluation Assurance Levels (2)

- EALn: Predefined Packages…
- Difference is in the component level
  - The higher the number
    - the more formal the description has to be
    - the more details are requested
- EAL5+
  - What is the '+' ?
- '+' = Augmentation
  - At least one component from a higher level has been taken (which one is defined in PP)

Assurance class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Development	ADV_ARC		1	1	1	1	1	1
	ADV_FSP	1	2	3	4	5	5	6
	ADV_IMP				1	1	2	2
	ADV_INT					2	3	3
	ADV_SPM						1	1
	ADV_TDS		1	2	3	4	5	6
Guidance documents	AGD_OPE	1	1	1	1	1	1	1
	AGD_PRE	1	1	1	1	1	1	1
Life-cycle support	ALC_CMC	1	2	3	4	4	5	5
	ALC_CMS	1	2	3	4	5	5	5
	ALC_DEL		1	1	1	1	1	1
	ALC_DVS			1	1	1	2	2
	ALC_FLR							
	ALC_LCD			1	1	1	1	2
	ALC_TAT				1	2	3	3
Security Target evaluation	ASE_CCL	1	1	1	1	1	1	1
	ASE_ECD	1	1	1	1	1	1	1
	ASE_INT	1	1	1	1	1	1	1
	ASE_OBJ	1	2	2	2	2	2	2
	ASE_REQ	1	2	2	2	2	2	2
	ASE_SPD		1	1	1	1	1	1
	ASE_TSS	1	1	1	1	1	1	1
Tests	ATE_COV		1	2	2	2	3	3
	ATE_DPT			1	2	3	3	4
	ATE_FUN		1	1	1	1	2	2
	ATE_IND	1	2	2	2	2	2	3
Vulnerability assessment	AVA_VAN	1	2	2	3	4	5	5

PP for Smartcards

- Requirements are the same for all
  - Threats
  - Security Objectives
  - SFRs
- Level of assurance is at least EAL4+
  - Usually we go for EAL5+ or EAL6+ for whole products.
  - If you read about EAL7 check the ST for the scope
- The augmentation for AVA_VAN is always the highest
  - Special treatment for smartcards
  - JIL Hardware Attack Subgroup

JHAS

Security Evaluation

Common Criteria – JHAS Mission Statement

The JHAS group

- Meets bi-monthly and consists of a wide variety of members
- State-of-the Art: Assess all HW and SW attacks (new and old) that may apply to smart cards and maintain a rating of those that is consistent with the advancements of attacks (published in a confidential document available to all members)
- Quality Assurance: Support evaluating labs to perform & assess attacks uniformly across all members, thereby helping to create a level playing field for all
- Promote the use of CC methodology for vulnerability analysis

JHAS group in CC Scheme – ~36 Members

Security Evaluation

Common Criteria – JHAS Documents

Application of Attack Potential to Smartcards

- Status: **Public**
- Rating tables and methodology

JIL Attack Methods for Smartcards

- Status: **Confidential**
- List of all attack classes
- Description of many attacks (not exhaustive, though!)
- Example ratings
- Serves as guideline for CBs, evaluation labs and vendors

Security Evaluation

Common Criteria – JHAS Attack Classification

Major attack classes are:

- Physical Attacks (e.g. Reverse Engineering)
- Overcoming Sensors and Filters
- Perturbation Attacks
- Side-channel Attacks
- Exploitation of Test Features
- Attacks on RNG
- Ill-formed Java Card Applications
- Software Attacks
- ...

Security Evaluation
Common Criteria – JHAS Attack Phases

Identification Phase:

- Perform the attack **once** to demonstrate its feasibility and / or achieve a one-time benefit (learning phase)

Exploitation Phase:

- Perform the attack **multiple times** for commercial exploitation

Information Flow between these Phases:

- One of the outcomes of the Identification Phase is a **virtual script** that tells the attacker of the Exploitation Phase how to perform the attack

Common Criteria for Smart Cards – Rating Tables

Range of values CC 3.x	TOE resistant to attackers with attack potential of:
0-15	No rating
16-20	Basic
21-24	Enhanced-Basic
25-30	Moderate
31 and above	High

We need to achieve 31 points for VLA.4 / VAN.5 (part of EAL 4+, 5+, 6+) for each and every attack path!

"Application of Attack Potential to Smartcards" (developed for JIL by JHAS group)

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
Multiple Expert	7	6
Knowledge of the TOE		
Public	0	0
Restricted	2	2
Sensitive	4	3
Critical	6	5
Very critical hardware design	9	NA
Access to TOE		
< 10 samples	0	0
< 30 samples	1	2
< 100 samples	2	4
> 100 samples	3	6
Not practical	*	*
Equipment		
None	0	0
Standard	1	2
Specialized	3	4
Bespoke	5	6
Multiple Bespoke	7	8
Open samples		
Public	0	NA
Restricted	2	NA
Sensitive	4	NA
Critical	6	NA

Bellcore Attack on RSA w/ Countermeasures

Factor	Comment	Identification	Exploitation
Elapsed Time	A glitch perturbation is induced. No sample preparation is needed and a straightforward setup is sufficient to obtain an error.	< 1 day (1)	< 1 hour (0)
Expertise	Without any logical countermeasures, considering the chip can be relatively easily disturbed, a proficient could apply the attack in identification, as well in exploitation.	Proficient (2)	Proficient (2)
Knowledge of TOE	According to the protocol, no specific knowledge of the TOE is required.	Public (0)	Public (0)
Access to TOE (number of samples)	Access to TOE will in practice always be of the order of less than 10 samples.	< 10 (0)	< 10 (0)
Open Samples/ Known Key	Samples with known key won't ease this attack.	NA	NA
Equipment	Fault injection equipment based on glitch induction.	Specialized (3)	Specialized (4)
Sub Total		**6**	**6**
Total	**VAN.1 – "No Rating"**	**12**	

Bellcore Attack on RSA: Countermeasures

Countermeasures in Hardware

- Redundancy, check sums, etc. on the chip level
- Example Secure Fetch (NXP)

Countermeasures in Software

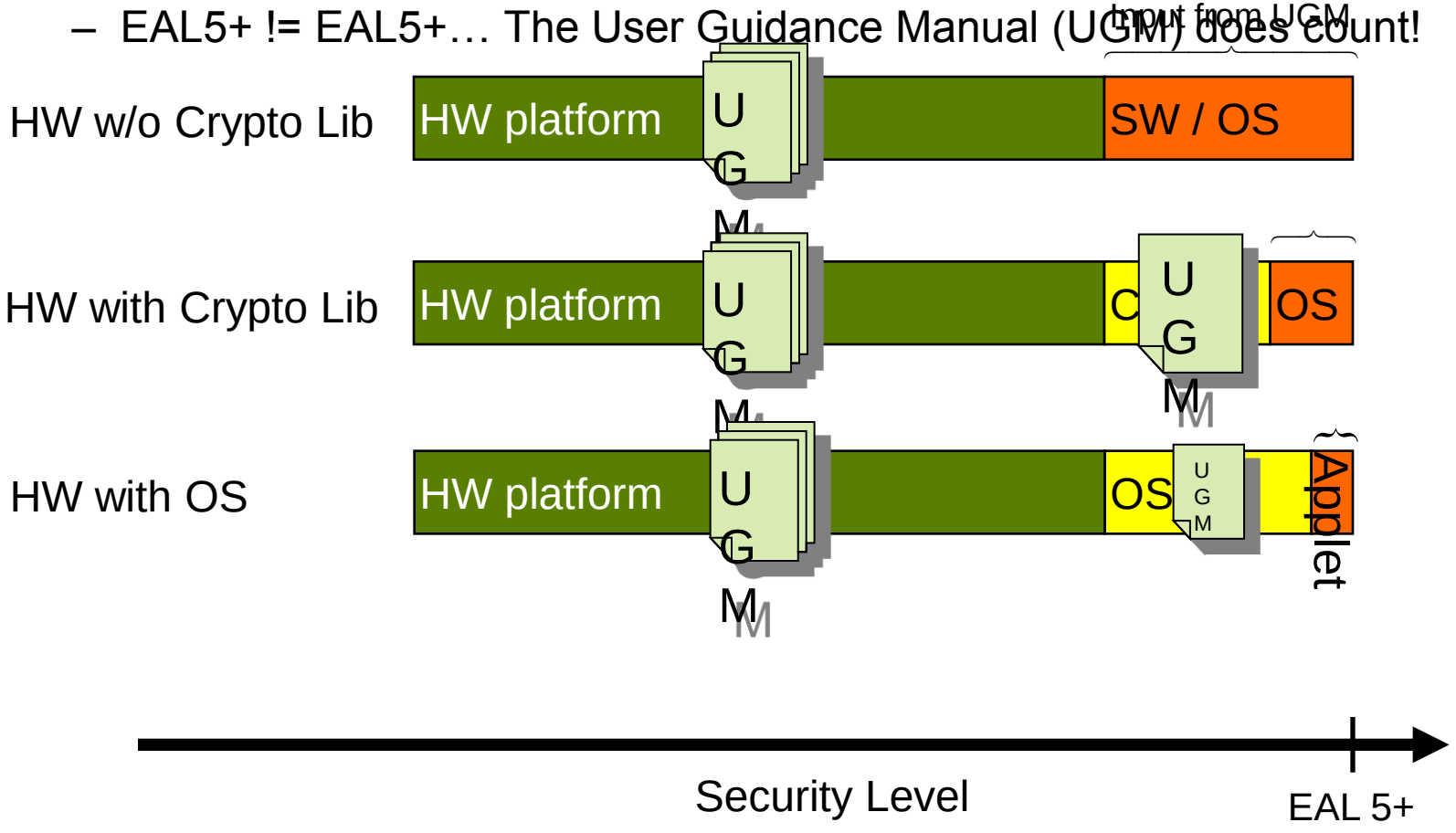
- A guidance on suitable countermeasures in SW may be given in the User Guidance Manual of the HW platform
- The implementation in the customer SW will then have to be tested in the Composite Evaluation
- Example: SW Verification of RSA (and much more…)

Both approaches can lead to an EAL5+ in HW (!)

- It is “simply” a question of where to make the cut in the HW-SW co-design of security features.

HW – SW Co-design

- **Reaching EAL5+ is always a HW/SW co-design effort in CC, so…**
  - EAL5+ != EAL5+… The User Guidance Manual (UGM) does count!



Penetration testing and countermeasures

Protecting user data and code in general (1)

- Code and data must be protected
  - Keys are not only used in coprocessors, e.g. storage in NV
  - Transfer on device, processing in CPU
  - Multiple applications
  - Code also needs protection
- Means of protection
  - Memory encryption, various key update frequencies for ROM, NV, RAM
  - Address scrambling
  - Integrity protection
  - CPU features for SPA resistance
  - Firewall management
  - Resource rights management
  - Redundant registers/HW
  - Blinded CRC
  - Sensors (Voltage, light,…)

Protecting user data and code in general (2)

- Encryption
  - Introduces latency → caching
  - Where to store the key?
- Software builds on top
  - Encrypted & integrity protected transfer (or enforced later)
  - SPA protected data transfer
- Residual information must be destroyed
  - SPA resistant clean-up
  - Take care of distances when deleting data
- Enforce code integrity
  - Code signatures calculated on the fly and checked by hardware and/or software
- Distinguish between possible attacks and attacks which are certain
  - Several counters, special counters (Micro NV)

Example Secure CRC

- Any low bandwidth transfer, possibly sequential can be potentially read out
- CRC-8 e.g. has by definition a low bus width and is order-dependent
- Cyclic redundancy check is a linear operation
  - Polynomial division
- Seemingless decryption and secret sharing to have fully protected integrity checking

Example RSA (1)

- Fault attacks
  - Bellcore attack
  - Safe-error
  - Skipping instructions
- Integrity enforcing countermeasures
- Code signatures
- Algorithms can be checked
- Fault attacks can be easily simulated/tested
  - Breakpoint
  - Change value
  - Run
- Such tests are part of ATE

Example RSA (2)

- SPA attacks
  - Regular algorithms, Montgomery ladder
- DPA attacks
  - Modulus blinding
  - Base blinding
  - Exponent blinding
- Can be checked via
  - Code review
  - Documented manual inspection

Example RSA (3)

- What remains?
- Timing
- Are SPA resistant algorithms really resistant?
  - Montgomery ladder
  - Distinction between R0 and R1 access?
  - Implement algorithms which even do resource randomization
- Is the blinding itself really secure
  - Complexity O(n^2) algorithm
  - Blinding itself needs to be protected

Example AES/DES

- These algorithms are the most complex ones to protect
- Very little algebra, but at least nowadays we understand masking well
- Profiled DPA
- Counter with a combination of masking (area) and hiding (performance)

Masking

- Randomized redundant representation
  - $v \rightarrow (v_1, \ldots, v_n) \;\; s.t. \;\; v = v_1 * \cdots * v_n$

- n th-order masking
  - n+1 shares
  - All n-tuples are independent of v
  - Adversary needs to
    - identify n+1 leakage samples (e.g. t samples per traces, n=1 → t*(t-1))
    - and combine their information

- Challenge $f(v) = f(v_1) * \cdots * f(v_n)$
  - Usually achieving
    is not straightforward

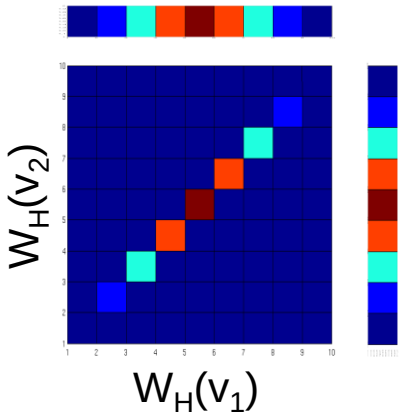
Masking Few Bits (1)

- Assume little structure (e.g. block cipher)
  - Boolean masking
    - $(v_1 = v \oplus m, v_2 = m)$
- Alternatively
  - Multiplicative masking (zero-value problem)
    - $(v_1 = v * m, v_2 = m)$
  - Affine Masking
    - $(v_1 = v * m_1 \oplus m_2, v_2 = m_1, v_3 = m_2)$

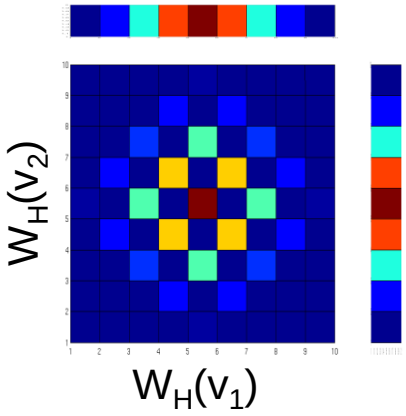
Masking Few Bits (2)

- Marginal PDFs are independent → joint PDF

- WH(v)=0

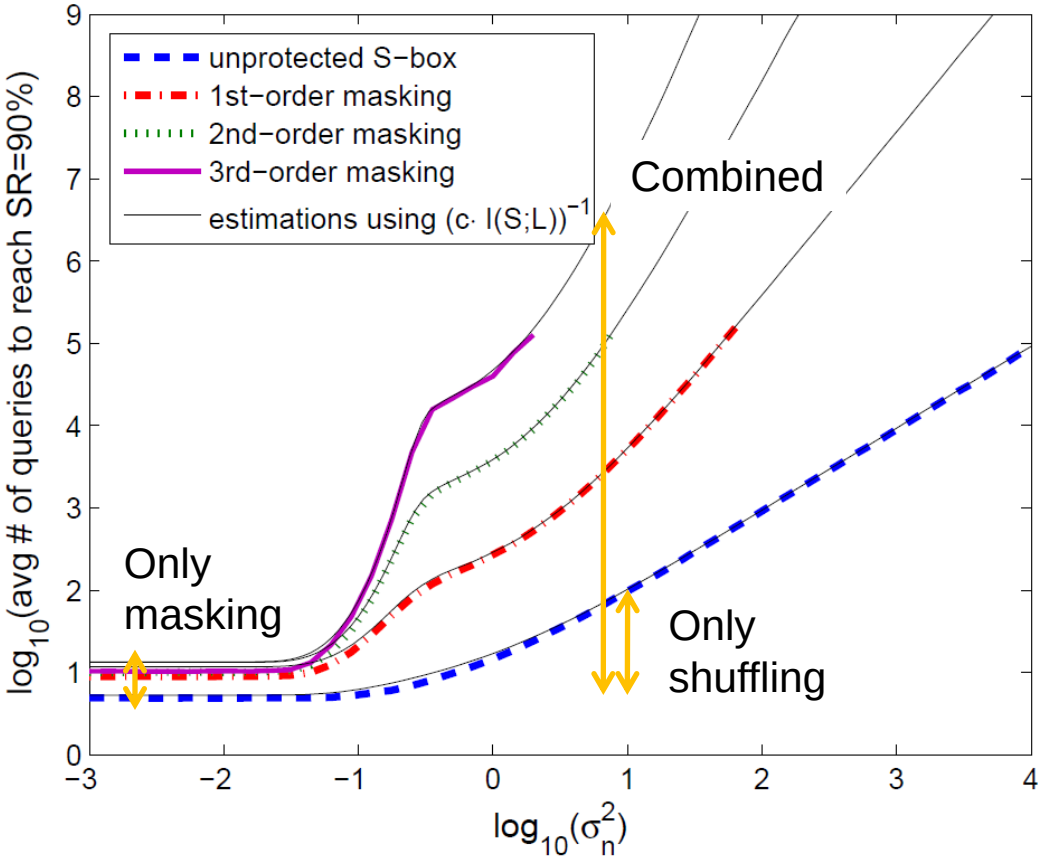


WH(v) = 4



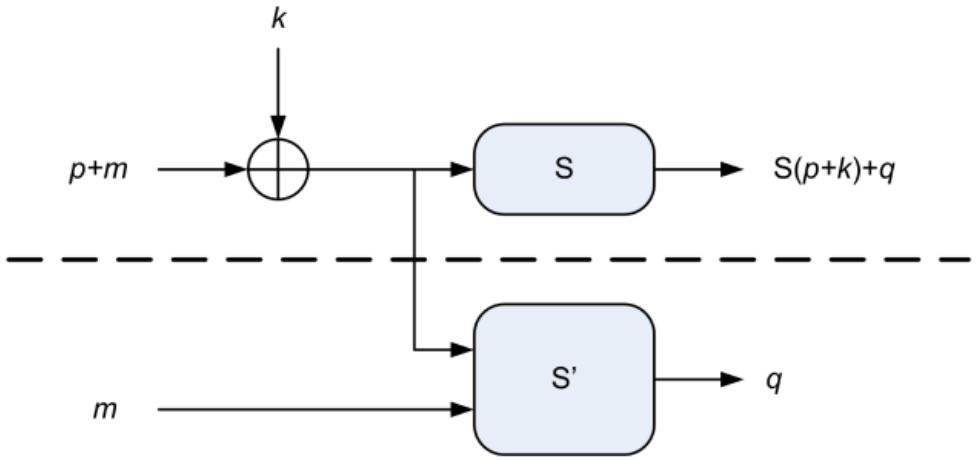
- Effect
  - k shares, sufficient noise
  - Number of traces relates to $(\sigma_n^2)^{k/2}$
  - Combination results in additional loss

Masking Few Bits (3)



Masking in Software (1)

- First-order masking
  → Lookup tables



- Higher order masking
  - Secure table computation for 2nd order masking
  - Test all subsets!

- Check Hamming distance
  - Buses, registers,...

Masking in Software (2)

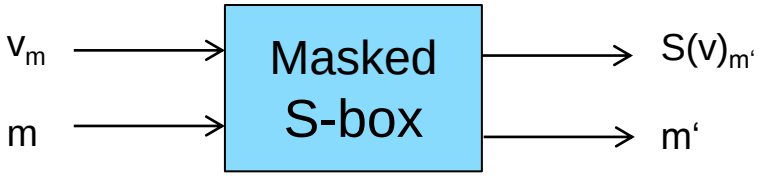
- Rivain and Prouff – CHES10
  - Provable secure masking for AES with arbitrary order
  - Based on Private Circuits

- Genelle, Prouff, and M. Quisquarter – CHES11
  - Combination of additive and multiplicative masking

- Cycle counts for a masked AES
  - Pay for security directly in execution time

Masking order	AES cycles
w/o masking	2 000
1	25 000
2	69 000
3	180 000

Masking in Hardware (1)

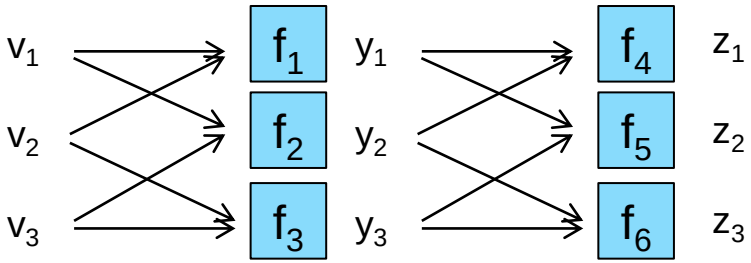


- Unclear what synthesizer does
  - Unintentional unmasking
  - Unintentional combination function

- Data dependent phenomena
  - Glitches
  - Early propagation
  - Cross-talk

Masking in Hardware (2)

- Nikova et al. – Threshold implementation
  - Independent processing of subset of shares



- If shares processed in parallel
  - Univariate leakage
  - But still higher order attack

And the goal for evaluation?

- Do not reveal the key to an attacker with high attack potential
  - Knowledge of countermeasures
  - Open samples for profiling
  - About 1M traces for the attack phase and potentially more for profiling
- “Do not reveal” means a remaining entropy of about 100 bits
  - Single DES therefore anyway out of scope
  - For AES we can loose 28 bits, for 2K3DES 12 bits
- Perform attack without timing CM
- Add time randomization to have sufficiently large margin

So, how to evaluate an implementation

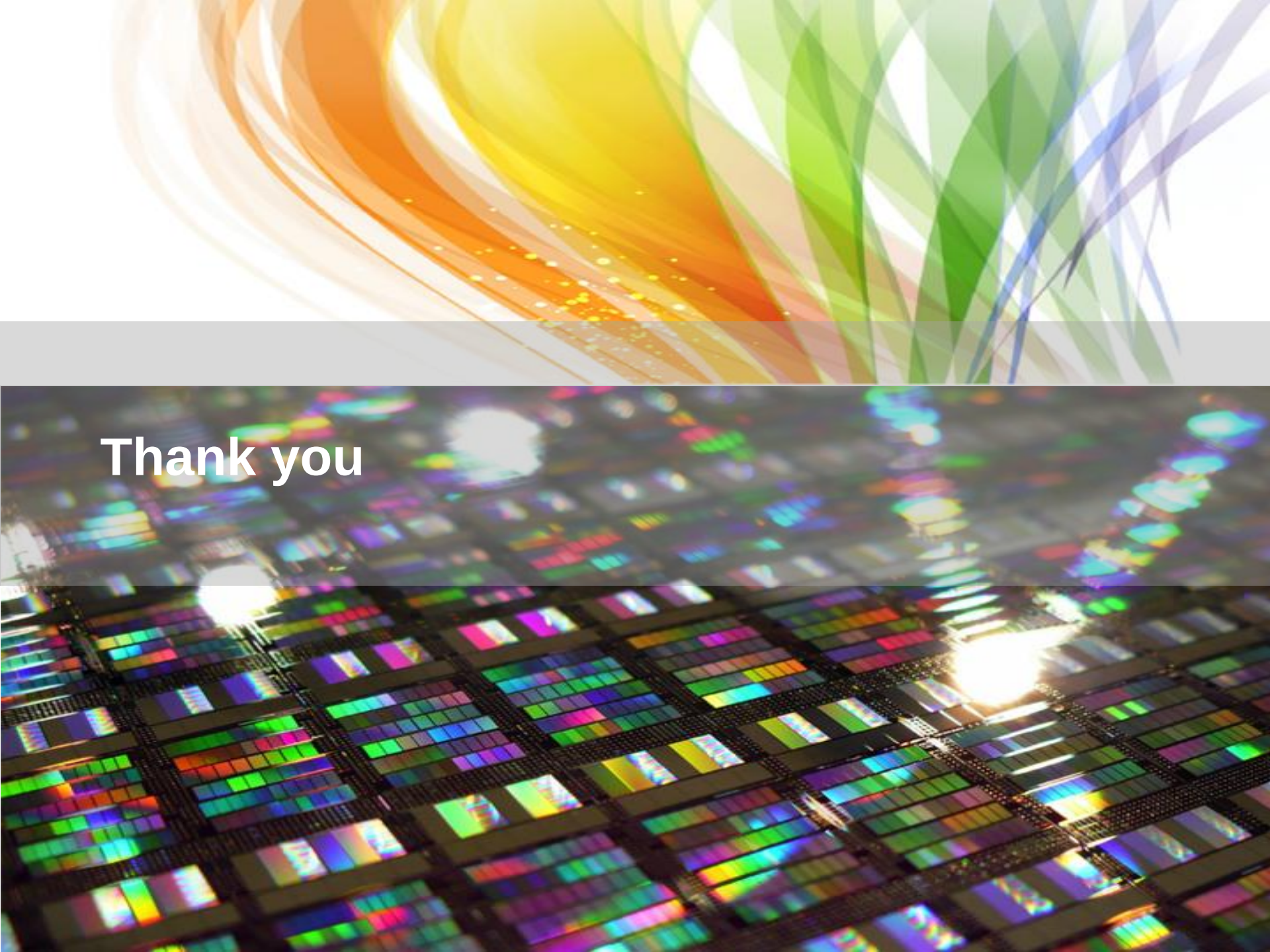
- Always use best attack path
  - Windowing
  - Profiling
  - Dimensionality reduction
- Always use cutting edge measurements
  - Power Analysis has little meaning
  - For unprotected MCU with 8-bit datapath, we usually stay below 100 traces.
  - If you need several thousand traces for masking with RNGs off, your setup can probably be improved
  - You consider noise which is unjustified and exponentially distorts your result
- Do leakage quantification at the end

Laser attacks

- Profiling done with XY table and without CMs on
- Light sensors?
- Is forward/backward enough?
- Feasibility of safe-error attacks? How many equal keys in the field (we do not build the application)
  - Points for samples
- Protect the key all the way

Interesting Topics

- Full framework at feasible costs
  - EM localization
  - Dimensionality reduction and other steps
  - Profiling
  - Leakage quantification
- Noise is a key ingredient
  - Local EM countermeasures
  - Masked architectures focused on maximizing time-randomization of shares
- CPU / memory encryption, integrity, SPA resistance support


Thank you