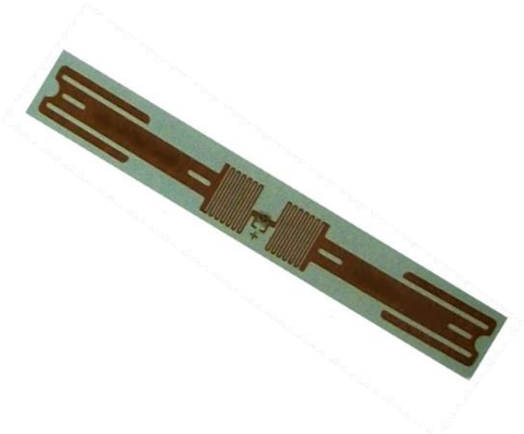
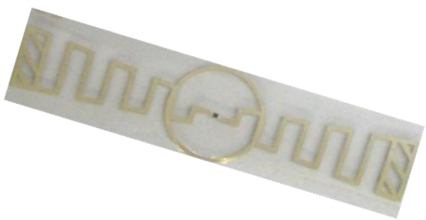
Fresh Re-Keying:

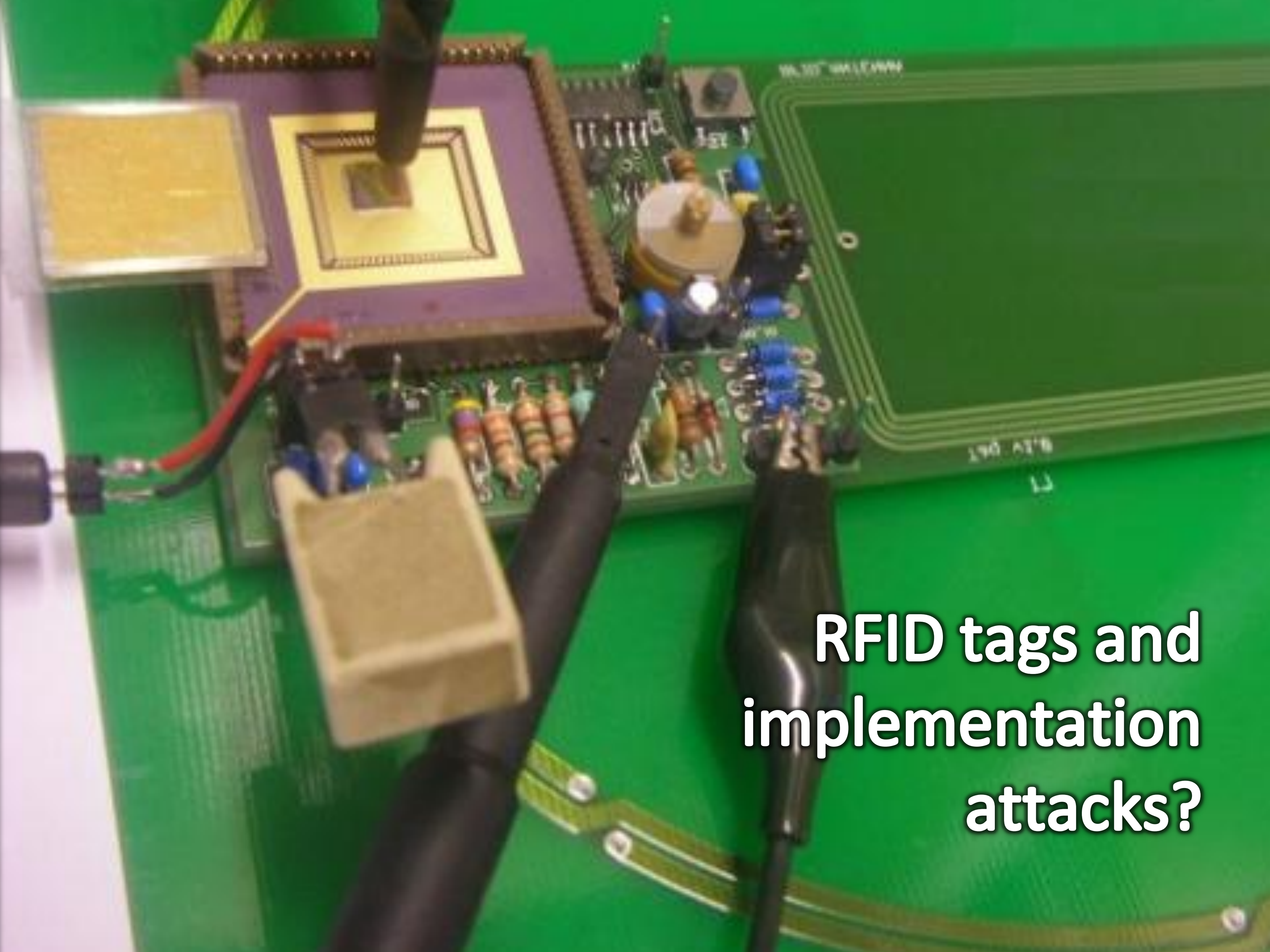
Security against Side-Channel and Fault Attacks for Low-Cost Devices





Marcel Medwed
François-Xavier Standaert
Johann Großschädl
Francesco Regazzoni

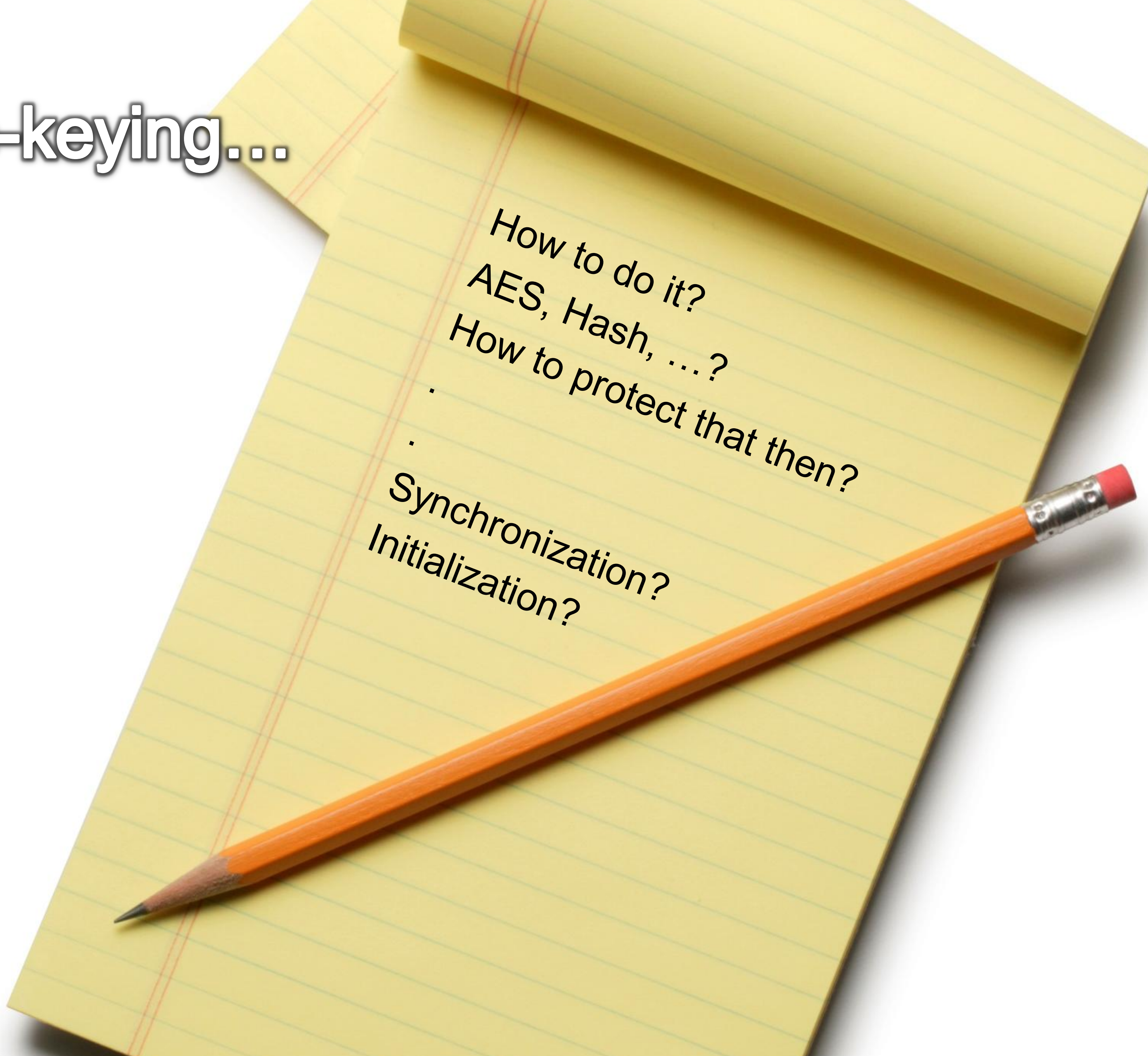
IAIK TU Graz
UCL Université catholique de Louvain
uni.lu UNIVERSITÉ DU LUXEMBOURG



RFID tags and implementation attacks?


Little costs
Low power
High performance

Re-keying…



How to do it?
AES, Hash, …?
How to protect that then?
.
.
Synchronization?
Initialization?

Fresh re-keying

Outline

- Implementation Attacks

- Fresh Re-keying
- Hardware Architecture
- Security Analysis

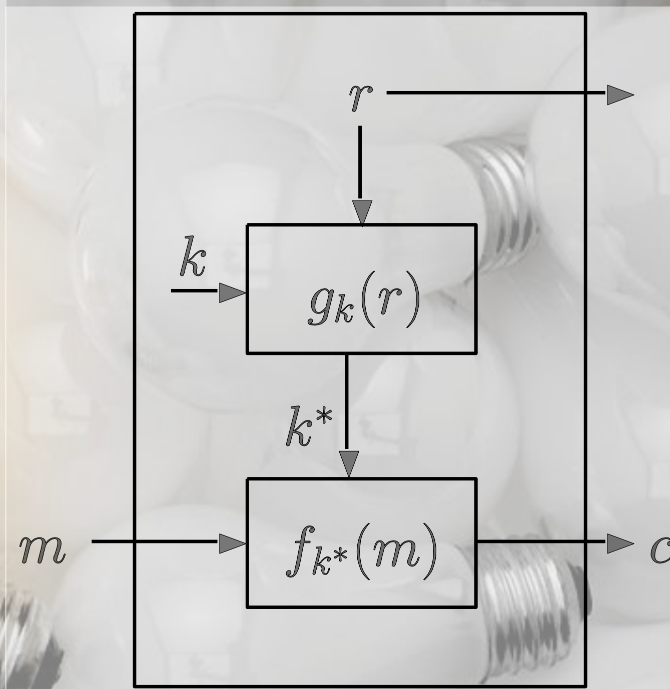
- Further research and Conclusions

Implementation Attacks

Attack	**Simple Power Analysis**	**Differential Power Analysis**	**Differential Fault Analysis**
# Invocations	One or few power traces	10s - 100s power traces	2+ encryptions under the same key and plaintext
Goals (In symmetric setup)	Extract Hamming weights of intermediate values	Exhaustively recover sub-keys	Reduce key entropy to allow exhaustive search
Uses…	Profiling and good knowledge about implementation	Divide-and-conquer approach and statistics	

Fresh Re-keying: The Basic Idea

- Input m → Output $\{c,r\}$
- f_{k*} is e.g. AES with session key
- $g_k(r)$ does the re-keying
- Just shift the problem to $g_k(r)$?
- Yes, but $g_k(r)$ will be easy to protect

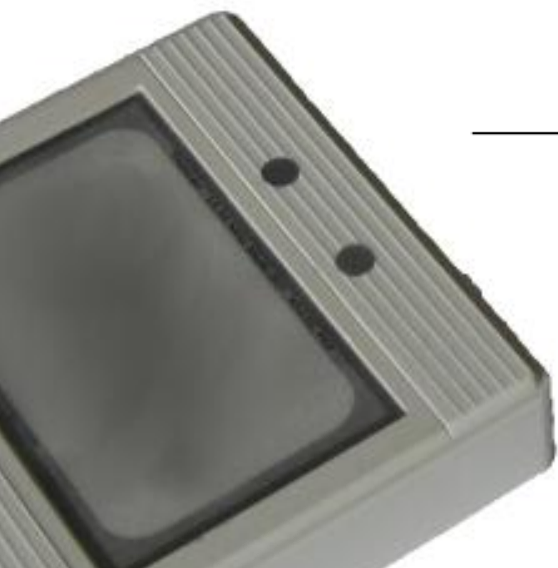


3-pass Mutual Authentication (ISO 9798-2)

$$R_B || \mathrm{Text}_1 \longrightarrow$$

$$\mathrm{Text}_3 || r_1 || r_2 || e_{k^*_{r_1}}(R_A || R_B || I_B || \mathrm{Text}_2) \longleftarrow$$

$$\mathrm{Text}_5 || e_{k^*_{r_2}}(R_B || R_A || \mathrm{Text}_4) \longrightarrow$$



Properties & Candidates

$$g_k(r)$$

- P1: Diffusion
- P2: No need for synchronization
- P3: No additional key material
- P4: Little hardware overhead
- P5: Easy to protect against SCA
- P6: Regularity

$$k^* = Hash_k(r) \qquad k^* = k \text{ xor } r$$

$$k^* = k * r \pmod{GF(2^8)[y]/y^{16}+1}$$

- Implementation Attacks
- Fresh Re-keying
- Hardware Architecture
  - Shuffling
  - Secure Logic
  - Blinding
  - Post synthesis results
- Security Analysis
- Further research and Conclusions

Secure Logic Styles & Shuffling

r_2	r_1	r_0

k_2	k_1	k_0

r_2k_0	r_1k_0	r_0k_0
r_1k_1	r_0k_1	r_2k_1
r_0k_2	r_2k_2	r_1k_2

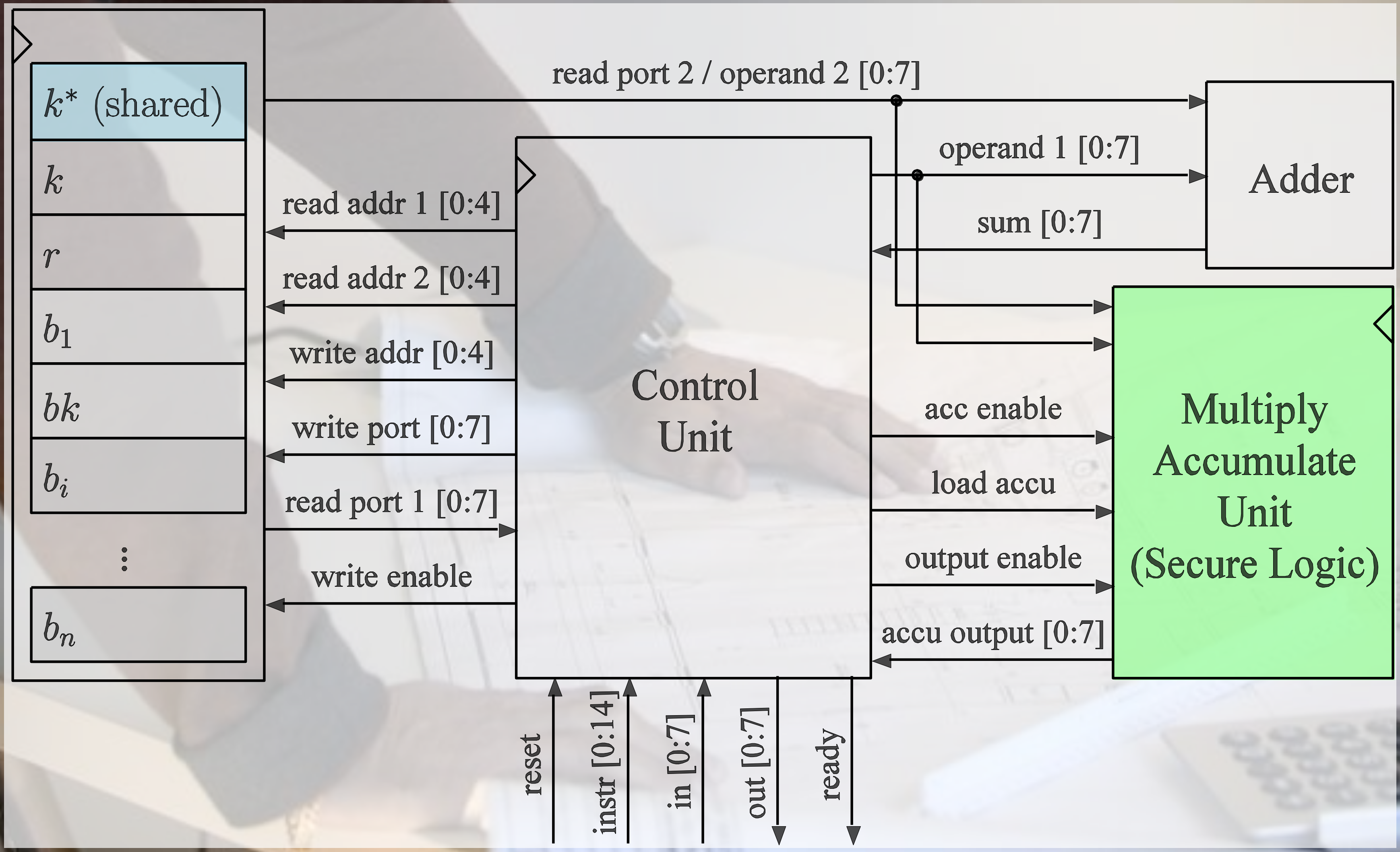
k_2^*	k_1^*	k_0^*

Blinding

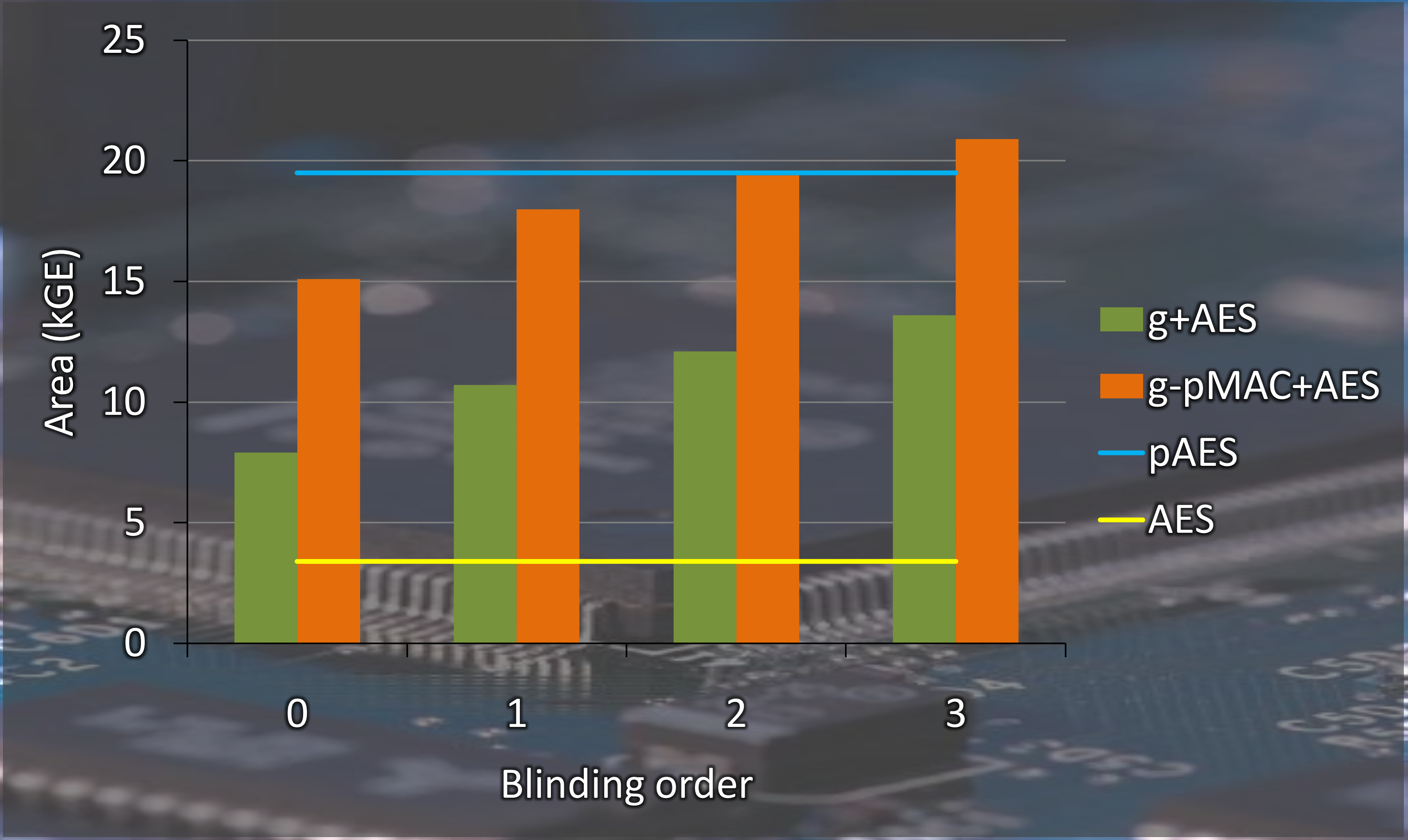
- Use randomized, redundant representation of data
- Addition and multiplication are distributive
- $k^* = k * r$

  $= (k+b) * r + b * r$
- Allows arbitrary blinding order

Effects of Countermeasures on the Architecture



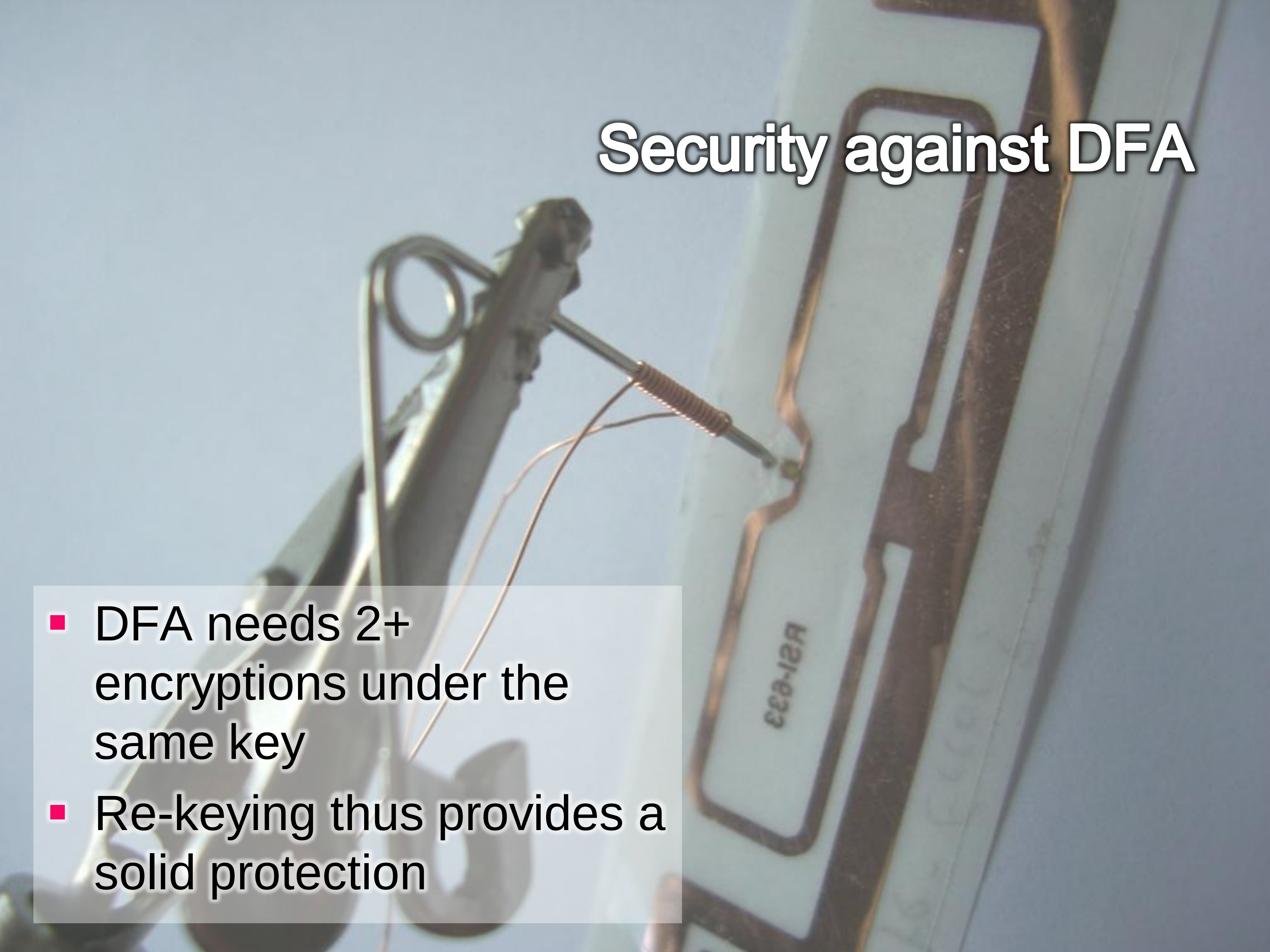
Post-Synthesis Results



- Implementation Attacks
- Fresh Re-keying
- Hardware Architecture
- Security Analysis
  - Choice of k
  - Security against DFA
  - Component-wise Security (SPA and DPA)
  - Security of the Complete Scheme (D&C)
- Further research and Conclusions

Choice of k

- Not every ring element is a unit
- Choosing a multiple of (y+1) leads to a reduced session-key space
- Accounts for a loss of entropy of 0.0056 bits out of 128

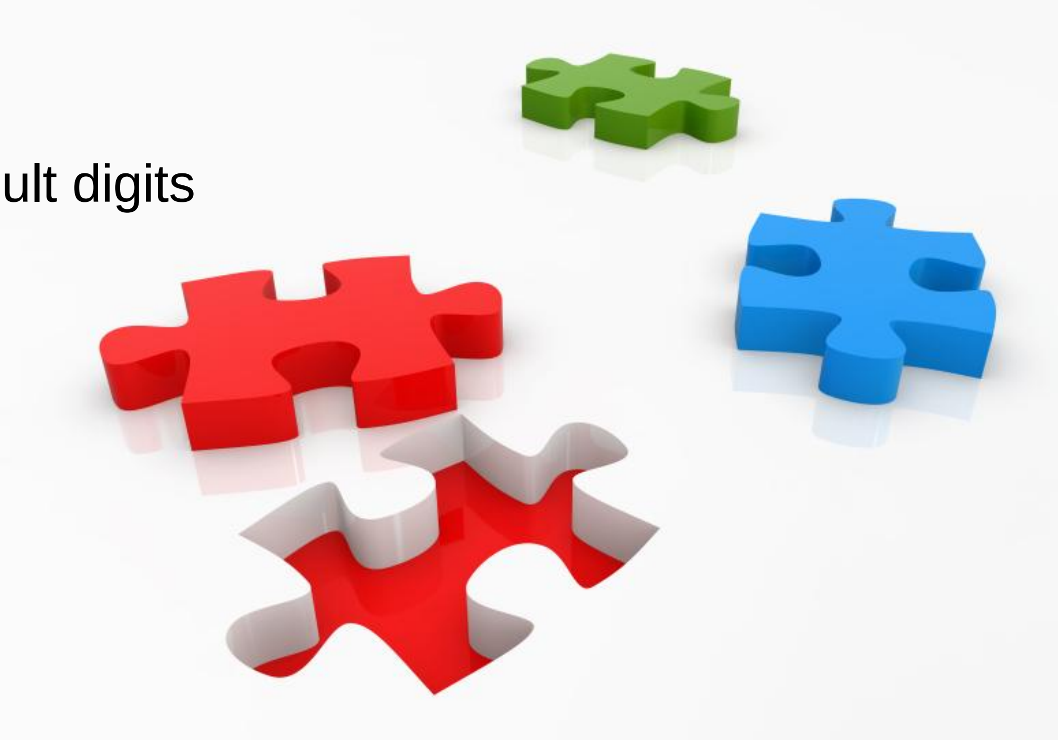


Security against DFA

- DFA needs 2+ encryptions under the same key
- Re-keying thus provides a solid protection

Component-wise Security

- SPA and DPA against g
  - Blinding
  - Shuffling
  - Secure Logic
  - An adversary might get Hamming weights of result digits with unknown indices
- SPA on AES
  - Shuffling

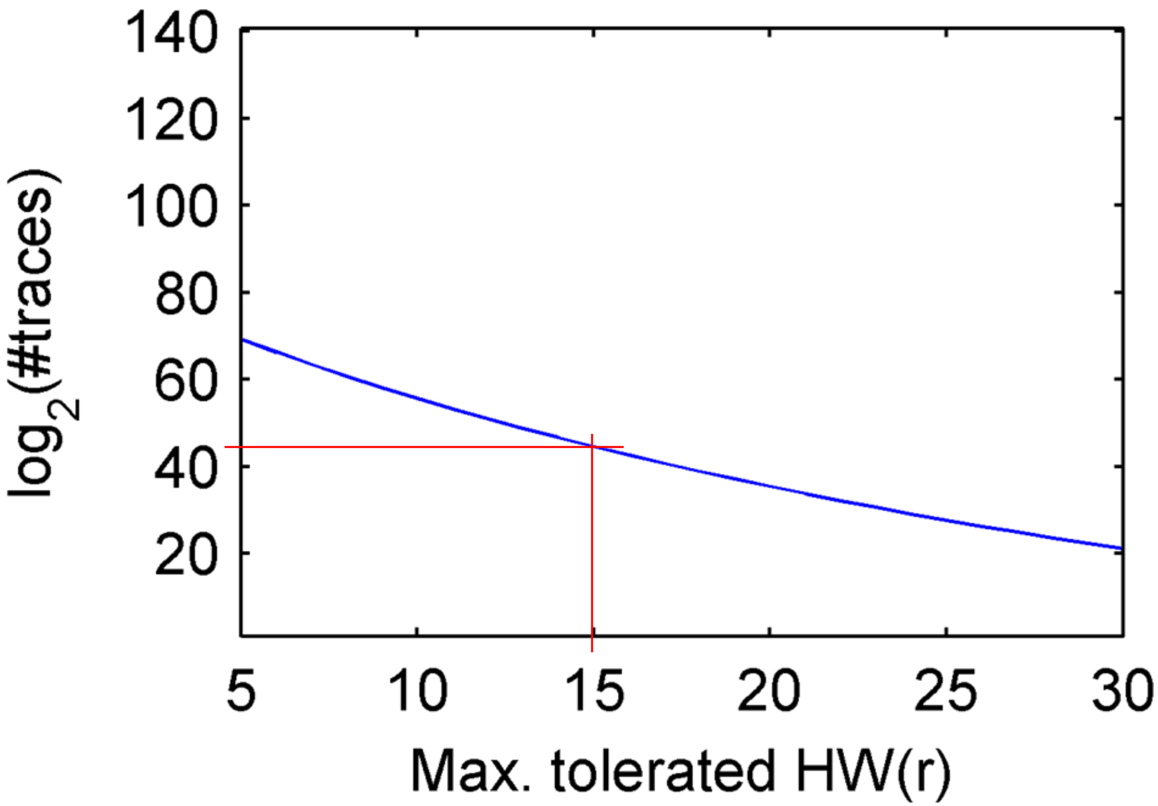


Security of the Complete Scheme

- One bit of k^* depends on HW(r) bits of k
- $\Pr[\mathrm{HW}(r) \leq X] = \sum_{i=0}^{X} \frac{\binom{n}{i}}{2^n}$
- #bits for hypothesis usually >1
- #traces for attack usually >1

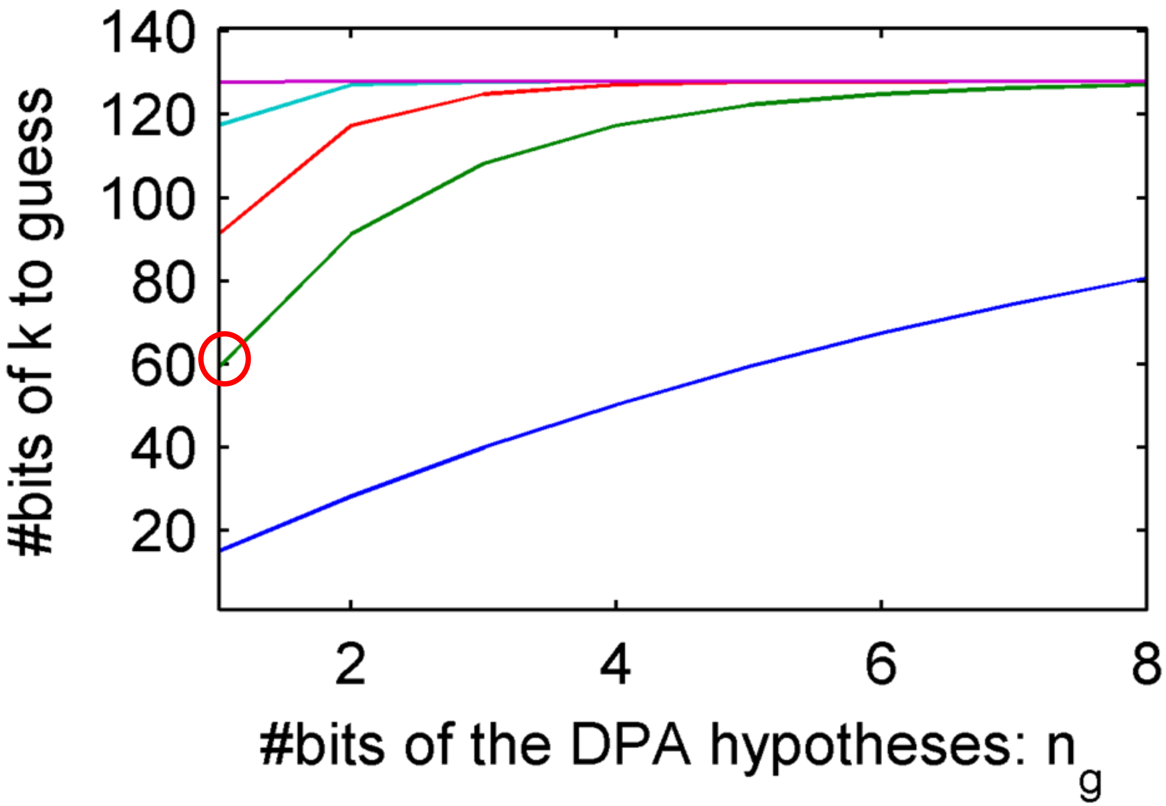
- #bits in total → $\left(1 - \left(\frac{n-X}{n}\right)^{n_t \cdot n_g}\right) \cdot 128$

An Example



- Observe traces with HW(r) less equal 15
- Need to record ~$n_{t^*}2^{44}$ traces

An Example



- Observe traces with HW(r) less equal 15
- Need to record ~$n_t * 2^{44}$ traces
- Set $n_t = 5$ and $n_g = 1$ → 2^{60} Hypotheses

- Implementation Attacks
- Fresh Re-keying
- Hardware Architecture
- Security Analysis
- Further research and Conclusions
  - Algebraic Side-Channel Attacks
  - The best Choice for g
  - Two parties

Algebraic Side-Channel Attacks

- g has a simple structure
- Thus ASCA is likely to apply
- Shuffling thwarts basic ASCA
- Topic is recent, needs further investigation

The best Choice for *g*

- We picked g since it fulfills the minimum requirements
- There might be better choices
- Randomness extractors?

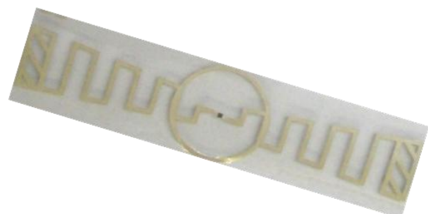
Protecting Two Parties

- How to extend the scheme to two parties
  - Restrict the choice of r
  - Does coding theory help?

Conclusions

- Fresh re-keying separates the system in an SCA target and a cryptanalysis target
- SCA target generates session key, is small and is easy to protect
- Complete solution is more efficient than previous proposals (area and security)
- Only one party can be protected
- Lots of further research…

Fresh Re-Keying:
Security against Side-Channel and Fault Attacks for Low-Cost Devices


Marcel Medwed
François-Xavier Standaert
Johann Großschädl
Francesco Regazzoni

IAIK TU Graz
UCL Université catholique de Louvain
uni.lu Université du Luxembourg