Fault Analysis Study of IDEA

Christophe Clavier[1], Benedikt Gierlichs[2], Ingrid Verbauwhede[2]
[1] Gemalto Security Labs, La Ciotat, France
[2] K.U.Leuven, ESAT-COSIC, Belgium

04/09/08 | Session Code: CRYP-203

RSACONFERENCE2008

Outline

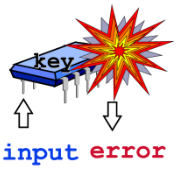
- Introduction
  - What are Physical Attacks?
  - What is Fault Analysis?
- Fault Analysis of IDEA
  - Summary of the IDEA block cipher
  - Fault Analysis Study of IDEA (software implementation)
  - Our 3-step Differential Fault Analysis
- Conclusions

Introduction to Physical Attacks

- Physical attacks ≠ Cryptanalysis
  (gray box, physics)   (black box, maths)
  - Physical Attacks: all means to threaten the security of a device exploiting physical properties and its behaviour
  - Passively observing and analysing:
    - The duration of operations (Timing Analysis)
    - The power consumption of a device (Power Analysis)
  - Actively perturbing the intended operation:
    - Analyse faulty outputs (Fault Analysis)

What is Fault Analysis?

- Exploits faulty behavior provoked by physical stress applied to the device
- Fault injection means:
  - Short and marked modification (glitch) of
    - Supply voltage
    - Clock signal
  - Intense illumination of the circuit surface
    - By white light (e.g. a camera flashlight)
    - By laser beam
  - Intense electromagnetic field
  - Environmental temperature



Fault Analysis Methods

- There exist several fault analysis techniques, choice depends on:
  - The fault model
  - The way inputs are chosen
  - The way outputs vary

- Frequently applied techniques:
  - Collision fault analysis (CFA)
  - Ineffective fault analysis (IFA)
  - Differential fault analysis (DFA)

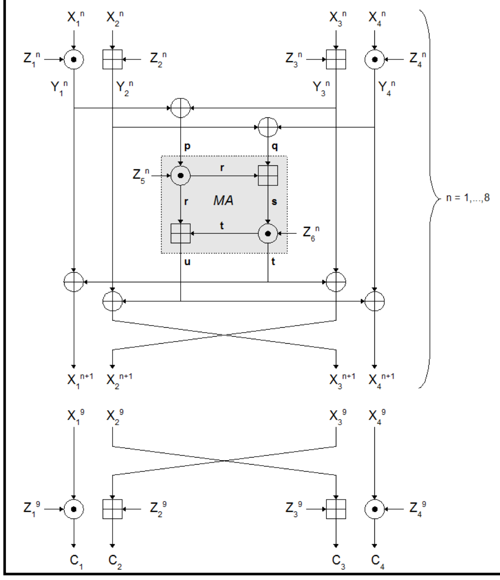
Summary of the IDEA block cipher

- IDEA is a 8.5 –round block cipher encrypting 64-bit blocks using a 128-bit key
- Introduced by Lai & Massey in 1991
- Available in crypto libraries (PGP, SSH, OpenSSL), used in embedded devices in GSM and Pay-TV
- Applies operations on three algebraic groups
- Difficult to cryptanalyse, even on reduced rounds
  - Best known result: Biham et al. FSE ’07
    6 rounds, 2^{64}-2^{52} plaintext/ciphertext pairs, $2^{126.8}$ encryptions

Physical attacks on IDEA

- Interesting to study, but almost no literature on the subject
- Differential Power Analysis:
  - Lemke et al. CHES ’04: DPA on multiplication and addition mod 2^{16}
- Fault analysis: no published result
- Our contribution:
  - A study of IDEA‘s vulnerability to
    - Collision Fault Analysis
    - Ineffective Fault Analysis
    - Differential Fault Analysis

IDEA – The algorithm



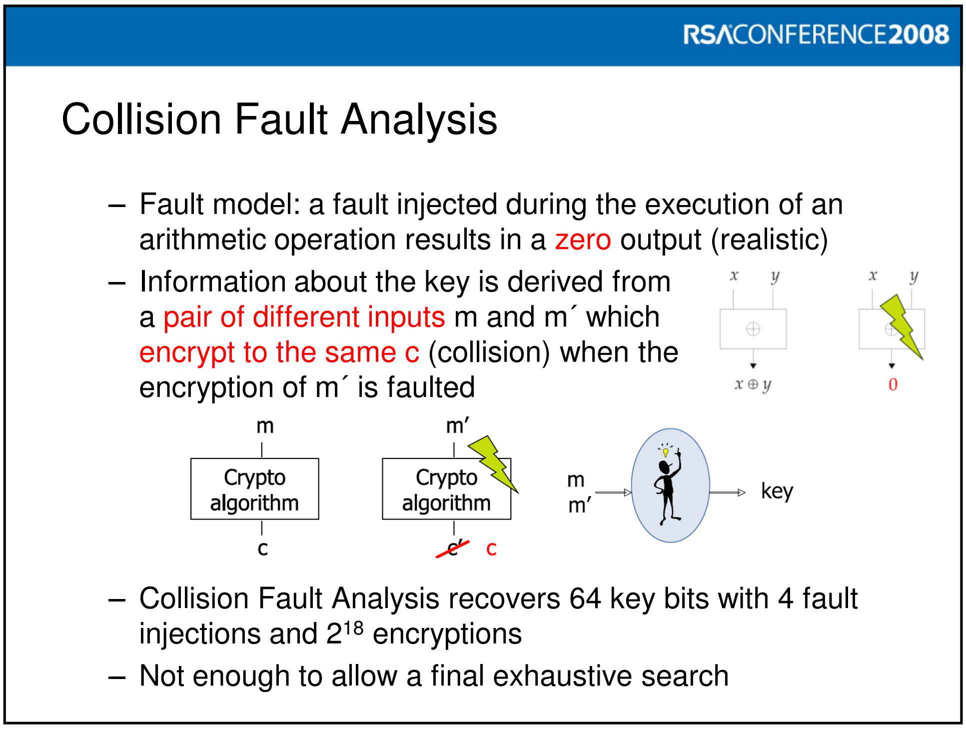
⊕ Bitwise XOR

⊞ Addition mod. 2^{16}

⊙ Multiplication mod. $2^{16}+1$

All operations are on 16 bit operands

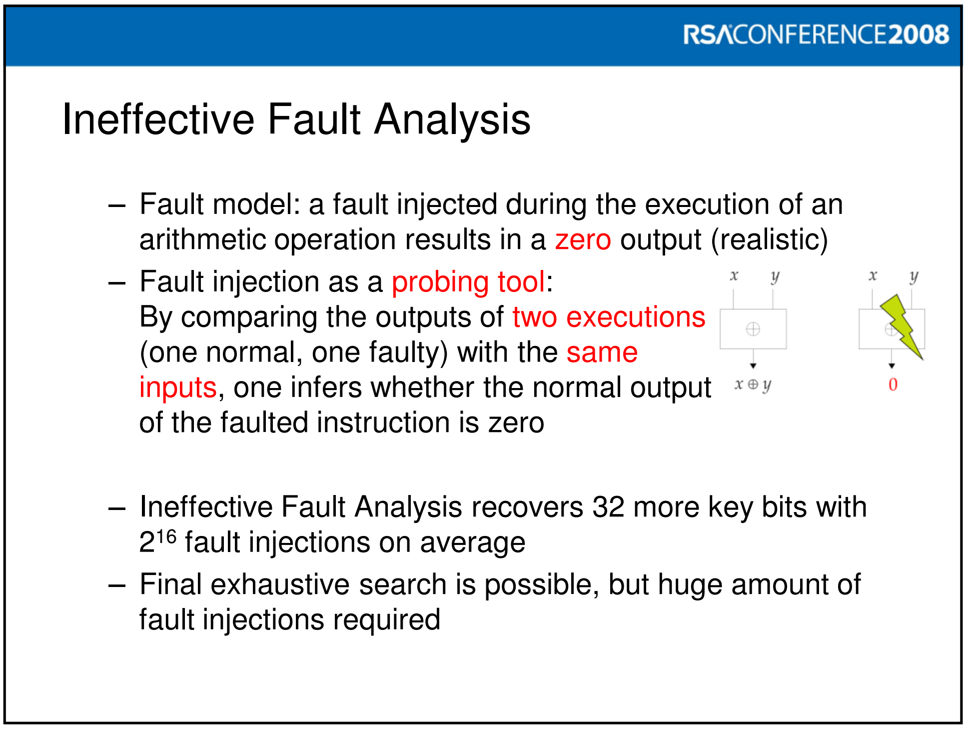
Key schedule simply selects ranges of keybits



Collision Fault Analysis

- Fault model: a fault injected during the execution of an arithmetic operation results in a zero output (realistic)
- Information about the key is derived from a pair of different inputs m and m´ which encrypt to the same c (collision) when the encryption of m´ is faulted

- Collision Fault Analysis recovers 64 key bits with 4 fault injections and 2^{18} encryptions
- Not enough to allow a final exhaustive search



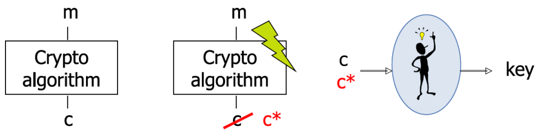
Ineffective Fault Analysis

- Fault model: a fault injected during the execution of an arithmetic operation results in a zero output (realistic)
- Fault injection as a probing tool:
  By comparing the outputs of two executions (one normal, one faulty) with the same inputs, one infers whether the normal output of the faulted instruction is zero

- Ineffective Fault Analysis recovers 32 more key bits with 2^{16} fault injections on average
- Final exhaustive search is possible, but huge amount of fault injections required

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

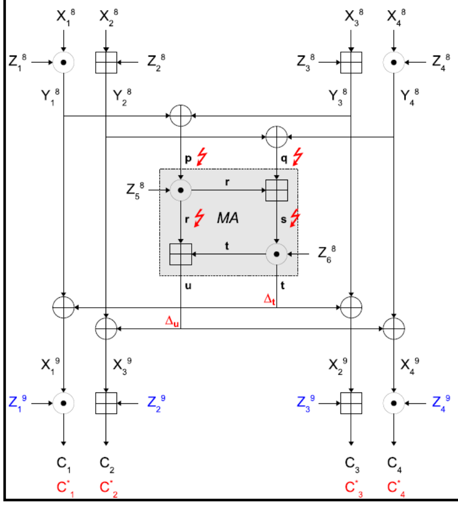


- No particular assumption about the fault‘s effect, random fault model
- Fault injection time does not need to be very precise
- Differential Fault Analysis on IDEA requires three steps to recover 93 key bits with a few fault injections

---

Differential Fault Analysis on IDEA – Step 1

- Finding the subkeys of the output transformation Z_1^9 to Z_4^9



A fault corrupts the value of either p, q, r or s in the last round $\rightarrow$ Δ_u, Δ_t

$$\begin{aligned} X_1^9 \oplus X^{*9}_1 &= X_2^9 \oplus X^{*9}_2 &= \Delta_t \\ X_3^9 \oplus X^{*9}_3 &= X_4^9 \oplus X^{*9}_4 &= \Delta_u \end{aligned}$$

Each pair (C, C^*) reduces the key space

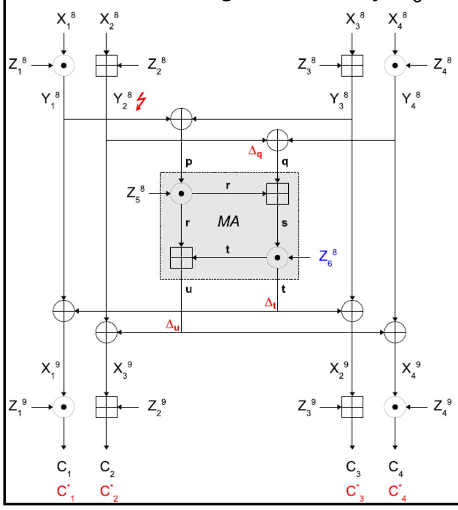
e.g.: any guess on (Z_1^9, Z_3^9) must verify:

$$(C_1 \odot (Z_1^9)^{-1}) \oplus (C_1^* \odot (Z_1^9)^{-1}) = (C_3 \boxminus Z_3^9) \oplus (C_3^* \boxminus Z_3^9)$$

62 bits of $(Z_1^9, \ldots, Z_4^9)$ are recovered with approximately 5 faults

Differential Fault Analysis on IDEA – Step 2

– Finding the subkey $Z_6{}^8$



A fault corrupts the value of Y_2^8 (or Y_4^8) in the last round → $\Delta_q, \Delta_u, \Delta_t$

From Z_1^9 to Z_4^9, one derives Δ_u, Δ_t and Δ_q

$TR_2 = \{(t,r) : (r \boxplus t) \oplus (r \boxplus (t \oplus \Delta_t)) = \Delta_u\}$

Any guess on Z_6^8 is eliminated if there exists no $(t,r) \in TR_2$ with:

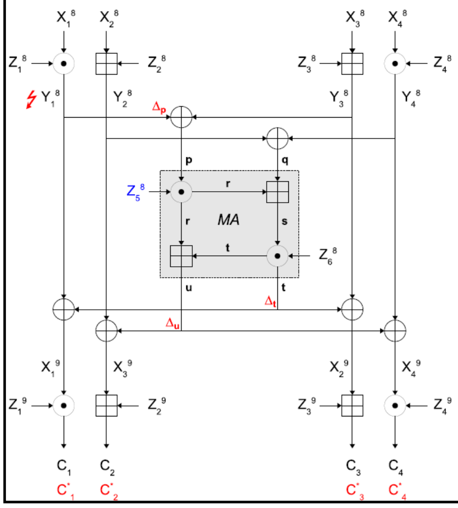
$$\begin{aligned} s &= t \odot (Z_6^8)^{-1} \\ s^* &= (t \oplus \Delta_t) \odot (Z_6^8)^{-1} \\ \Delta_q &= (s \boxminus r) \oplus (s^* \boxminus r) \end{aligned}$$

The correct value of the 16-bit subkey Z_6^8 is identified with approximately 5 to 10 faults

---

Differential Fault Analysis on IDEA – Step 3

– Finding the subkey $Z_5{}^8$



A fault corrupts the value of Y_1^8 (or Y_3^8) in the last round → $\Delta_p, \Delta_u, \Delta_t$

From Z_1^9 to Z_4^9, one derives Δ_u, Δ_t and Δ_p

Computes $TR_3 = \{(t,r) : \Delta_q = 0\}$, with:

$$\begin{aligned} t^* &= t \oplus \Delta_t \\ r^* &= ((r \boxplus t) \oplus \Delta_u) \boxminus t^* \\ s &= t \odot (Z_6^8)^{-1} \\ s^* &= t^* \odot (Z_6^8)^{-1} \\ \Delta_q &= (s \boxminus r) \oplus (s^* \boxminus r^*) \end{aligned}$$

Any guess on Z_5^8 is eliminated if there exists no $(t,r) \in TR_3$ with:

$$\Delta_p = (r \odot (Z_5^8)^{-1}) \oplus (r^* \odot (Z_5^8)^{-1})$$

The correct value of the 16-bit subkey Z_5^8 is identified with approximately only 3 faults

Differential Fault Analysis of IDEA

- After the three steps:
  - 93 out of 128 key bits have been recovered
  - The key can be determined by exhaustive search over the remaining 35 bits
- A trick allows to further reduce the number of fault injections required: faults for steps 2 and 3 are useful for step 1
- DFA on IDEA is practical: considers the very general random fault model
- DFA on IDEA is efficient: it is possible to reveal the key with as few as 10 faults

---

Conclusions

- We presented a study of several fault analysis techniques applied to IDEA (in software)
- Collision Fault Analysis does not recover enough key bits to pose a real threat
- Ineffective Fault Analysis finds more key bits, but requires a huge number of faults
- Differential Fault Analysis recovers 93 out of 128 key bits with as few as 10 faults
- Fault attacks against IDEA are practical and efficient, need for secure implementations

Thank you for your attention!

Questions?