浙江大學信息与电子工程学院
College of Information Science & Electronic Engineering, Zhejiang University

Persistent Fault Analysis on Block Ciphers

Fan (Terry) Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin,
Wei He, Ruyi Ding, Samiya Qureshi and Kui Ren

Zhejiang University

CHES2018, Amsterdam, The Netherlands, 09/12/2018

HUAWEI

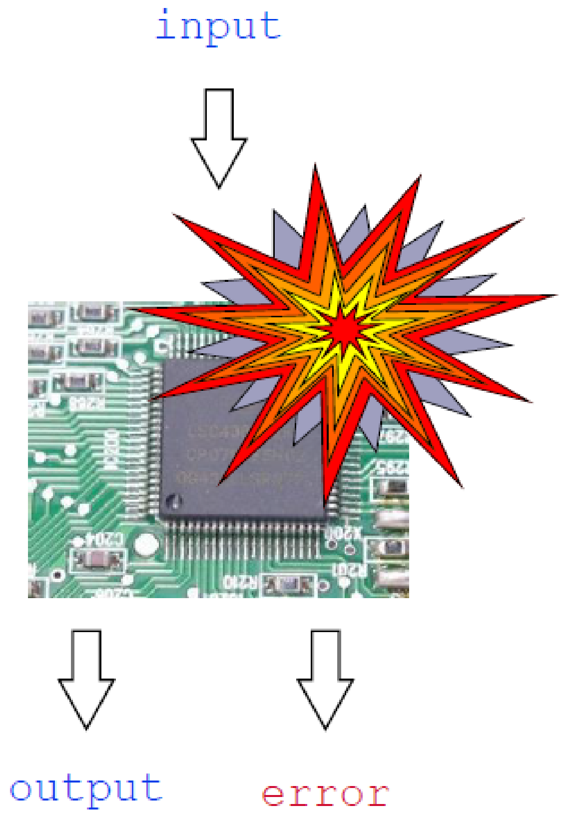
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

OUTLINE

1. Introduction

1.1 What are fault attacks

- Active attacks against cryptographic implementations
- FA (Fault Attack) first proposed by Boneh et al in 1996
- Two stages: Fault injection and Fault analysis

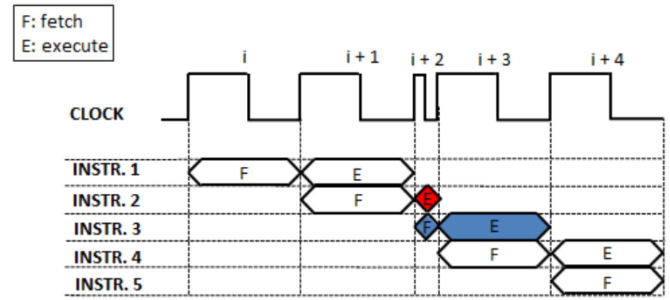


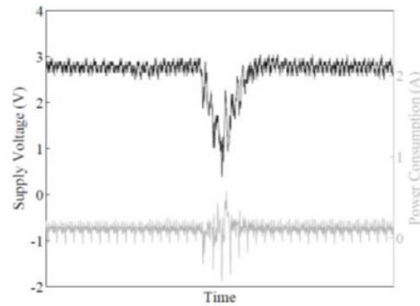
adopted from Josep Balasch in IACR Summer School 2015

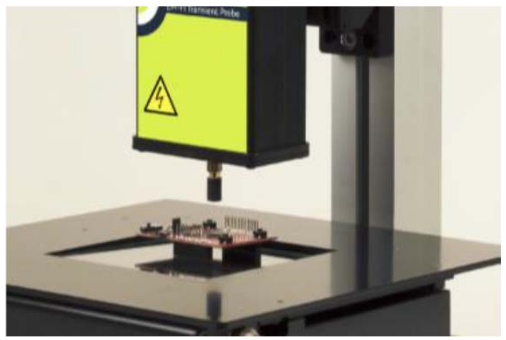
1. Introduction

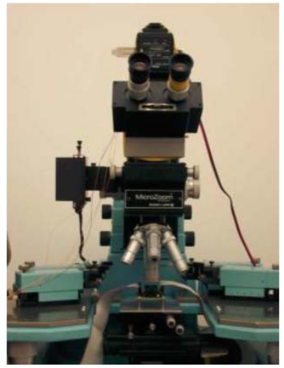
1.2 Fault injection (online)

- Categories
  - Non-invasive
  - Semi-invasive
  - Invasive
- Techniques
  - Clock Glitch
  - Voltage Spike
  - EM Pulse
  - Optical Laser
- Very popular form of non-invasive attacks









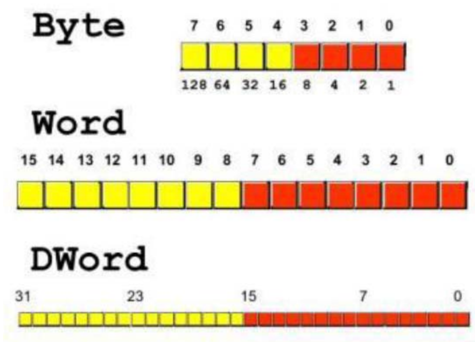
adopted from Josep Balasch in IACR Summer School 2015

1. Introduction

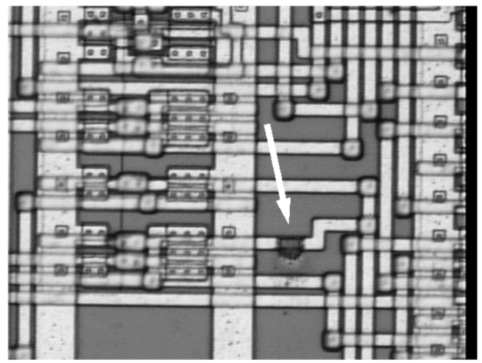
浙江大学 信息与电子工程学院
College of Information Science & Electronic Engineering, Zhejiang University

1.3 Fault model

- Granularity: how many bits are affected (aka fault width)
- Modification (aka fault type)
  - Stuck-at, e.g. zero or one
  - Flip
  - Random
- Control: on the fault location and on timing
  - Precise
  - Loose
  - None
- Duration of the fault
  - Transient
  - Permanent



adopted from Josep Balasch in IACR Summer School 2015

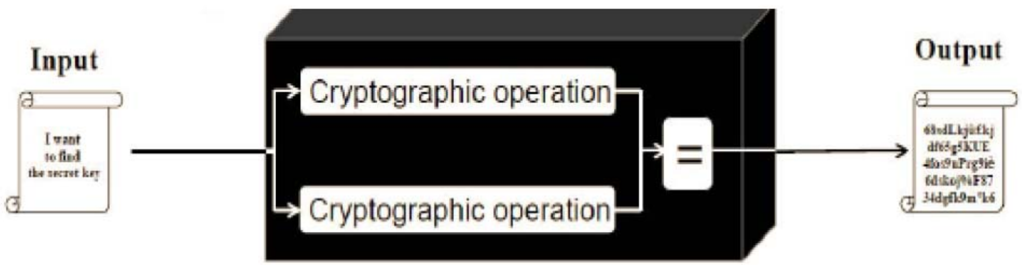


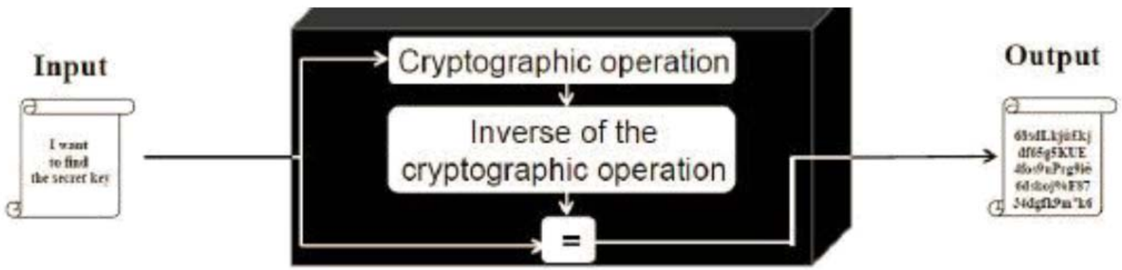
Persistent

浙江大学 信息与电子工程学院
College of Information Science & Electronic Engineering, Zhejiang University

1.4 Countermeasures

- Hardening hardware
  - Hide sensitive parts of the chip
  - Add filters and/or security sensors
- Hardening computations
  - Information redundancy (Addition of parities, linear codes, Ring embeddings, Infective computations)
  - Hiding countermeasures
  - Branchless implementations
  - Parallel execution or inverse execution





adopted from Josep Balasch in IACR Summer School 2015

1.5 Disadvantages of previous works

- Very tight time synchronization on the round calculation and the injection timing
- Very complicated analysis due to the random value and the fault propagation
- May not work if there are countermeasures against fault attacks

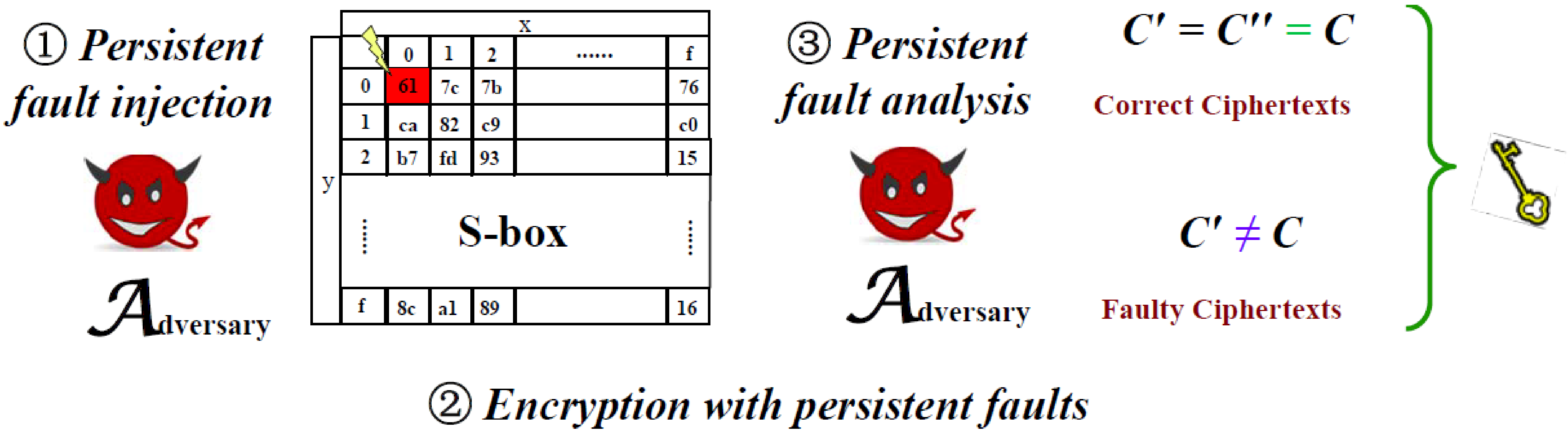
OUTLINE

2. Persistent Fault Attack

2.1 Fault model of PFA

- The adversary can inject faults before the encryption of a block cipher
  - Typically, these faults alter a stored algorithm constant
- The injected faults are persistent
  - The affected constant stays faulty unless refreshed
  - All iterations are computed with the faulty constant
- The adversary is capable of collecting multiple ciphertext outputs
  - A watchdog counter on detected faults is considered out of scope

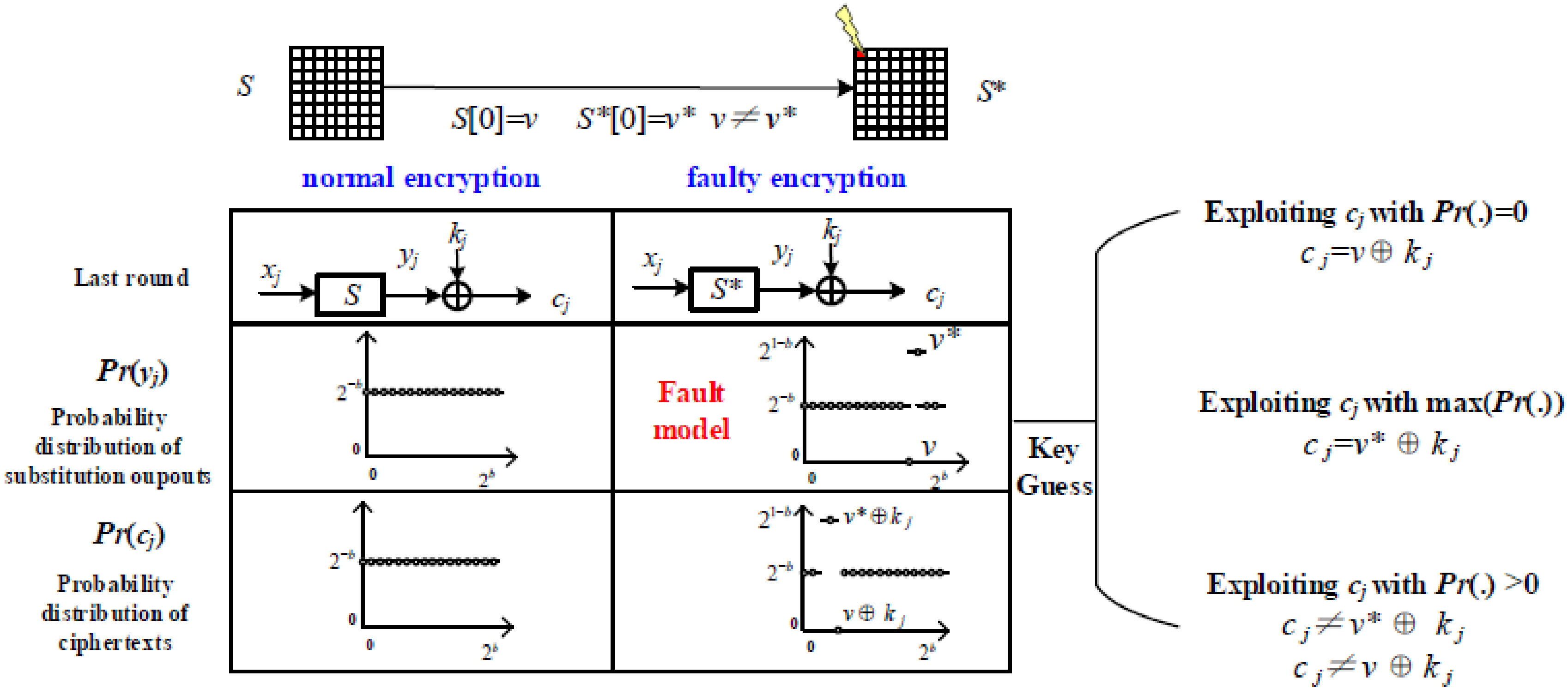
2.2 Core idea of Persistent Fault Attack



Three Stages

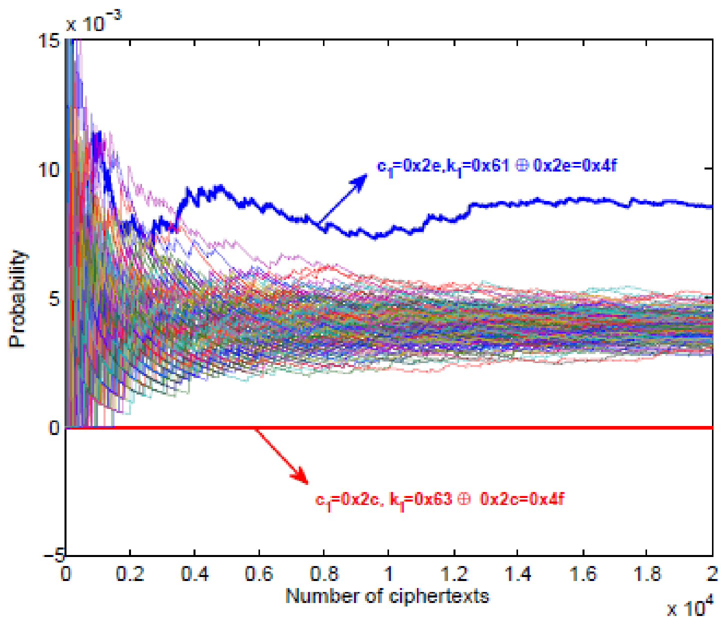
2.3 Overview of Persistent Fault Analysis (PFA)

- A statistical analysis on the last round, exploiting three types of fault leakages
- *v* and *v** are known

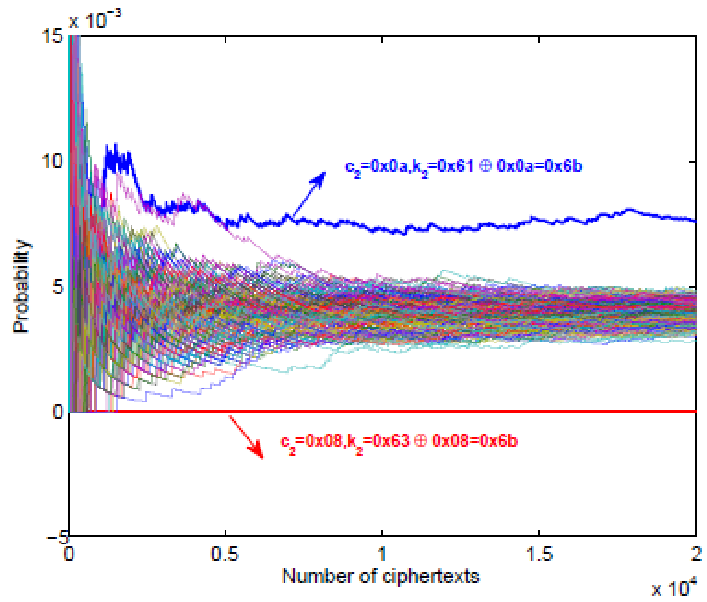


2.3 Illustration of analysis result

- Counts the number of appearances of possible values for the specific byte in ciphertexts



(a) Extract k_1 using the distribution of c_1



(b) Extract k_2 using the distribution of c_2

2.5 Comparison with other fault analysis

(1) The attack is not differential in nature and thus the control over the plaintext is not required.

(2) The adversary does not necessarily need live synchronization

(3) The fault model remains relaxed

(4) PFA can also be applied in multiple fault setting

(5) PFA can bypass some redundancy based countermeasures

(1) It needs higher number of ciphertexts as compared to DFA

(2) Persistent faults can be detected by some built-in health test mechanism.

OUTLINE

1. Introduction
2. Persistent Fault Attack
3. Persistent Fault Analysis on AES-128
4. PFA on Countermeasures against Fault Analysis
5. Case Study – Rowhammer-based PFA on T-box
6. Conclusion and Future Work

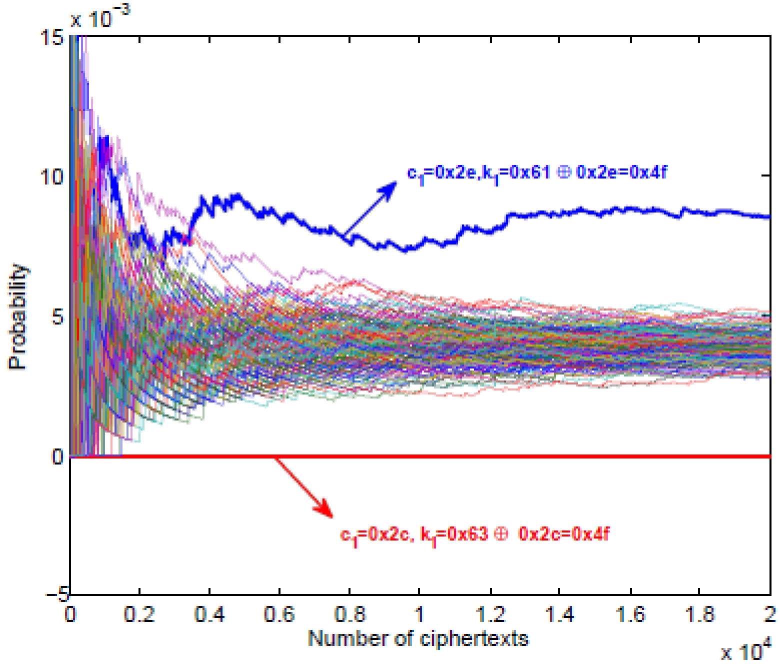
3.1 AES implementations

- S-box Implementation
- T-box Implementation

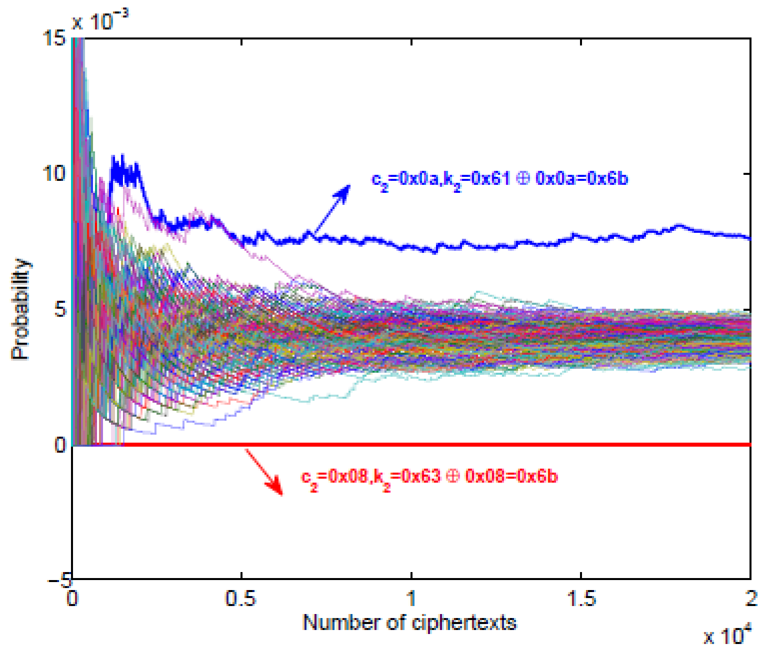
Table 1: Different implementations of AES-128 encryptions.

Type	Lookups in each round	Table size	Notes
I1	R_{1-10}: S	S:256B	Typical S-box implementation
I2	R_{1-10}: T_0, T_1, T_2, T_3	T_i:1KB	Typical T-box implementation
I3	R_{1-9}: T_0, T_1, T_2, T_3 R_{10}: T'_0, T'_1, T'_2, T'_3	T_i:1KB T'_i:1KB	Code can be found in rijndeal-amd64.S in the library Libgcrypt 1.6.3

3.2 PFA on vulnerable S-box implementation



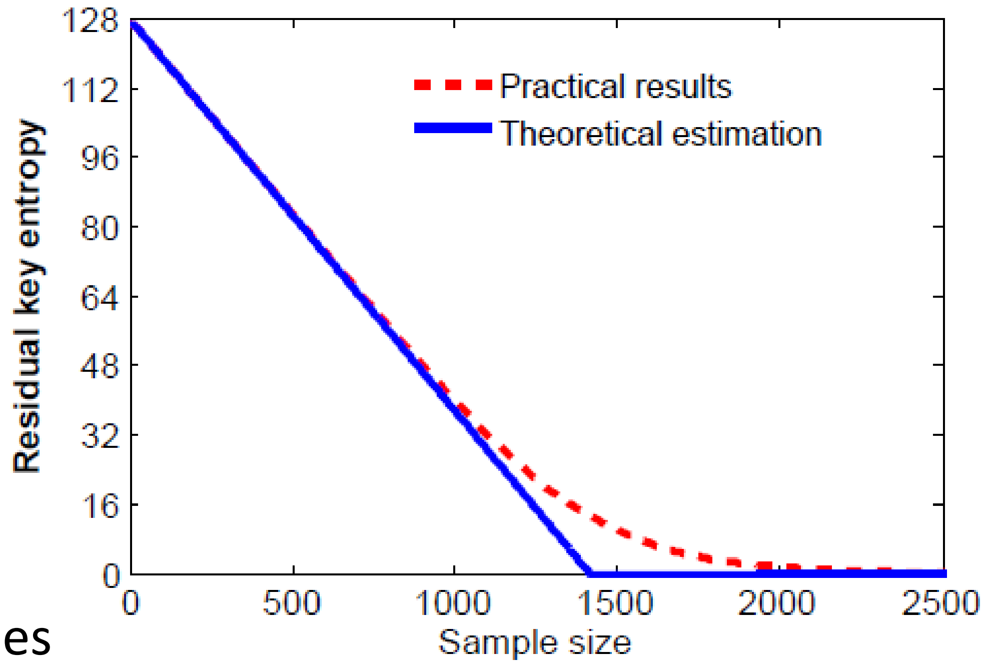
(a) Extract k_1 using the distribution of c_1



(b) Extract k_2 using the distribution of c_2

3.3 Practical result v.s. Theoretical estimation

- $\varphi_t(n)$ is calculated by the equation, coupon collector's problem.
- $\varphi(n)$ is calculated by the code
- $\varphi(n)$ is close to $\varphi_t(n)$
  - $\phi_t(n) \leq 16$ when $n \approx 1240$
  - $\phi(n) \leq 16$ when $n \approx 1360$
  - $\phi_t(n) \leq 1$ when $n \geq 1405$
  - $\phi(n) \leq 1$ when $n \geq 2148$
- The full key attacks are conducted ξ=1000 times
  - $1678 \leq N_f \leq 3504$
  - $N_f \approx 2281$ on average

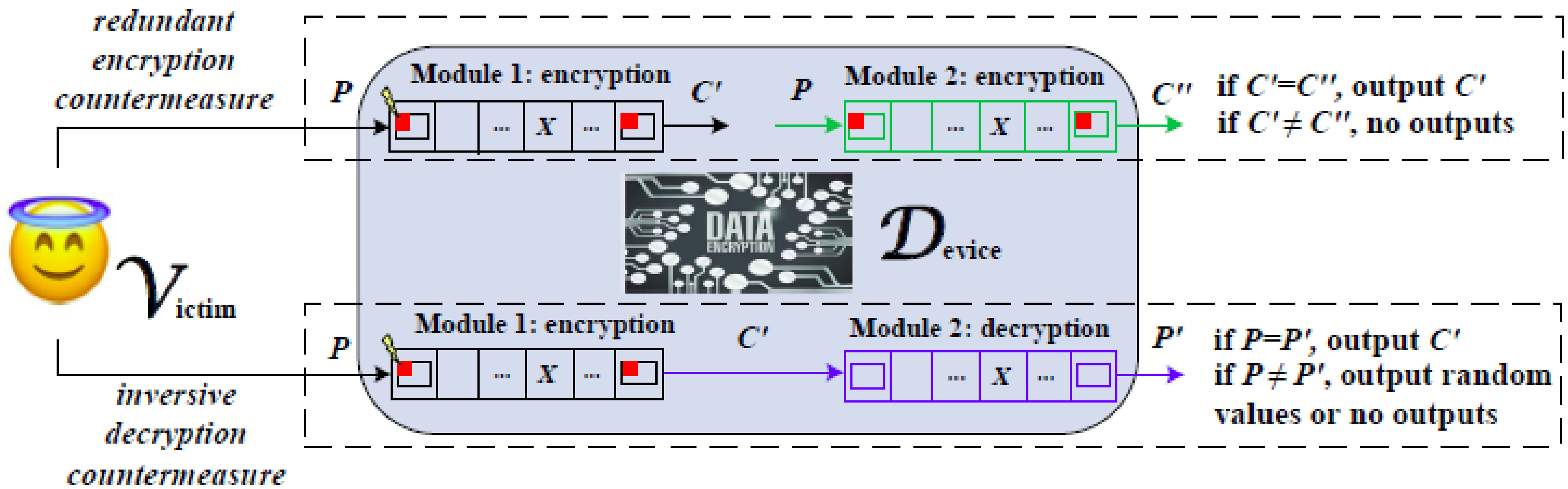


OUTLINE

4. PFA on Countermeasures against FA

4.1 Dual Modular Redundancy (DMR)

- Time redundancy v.s. Space redundancy
- Two modules: Module 1 and Modules 2
  - Redundant Encryption based DMR (REDMR)
  - Inversive Decryption based DMR (IDDMR)
- PFA is naturally against REDMR



4.2 Three types based on the reaction

- NCO: No ciphertext output
- ZVO: Zero value output
- RCO: Random ciphertext output

- REDMR
  - If both the modules use shared memory, *i.e.*, common lookup tables
  - All three countermeasures will fail

- IDDMR
  - A stronger countermeasure (two different lookup tables)

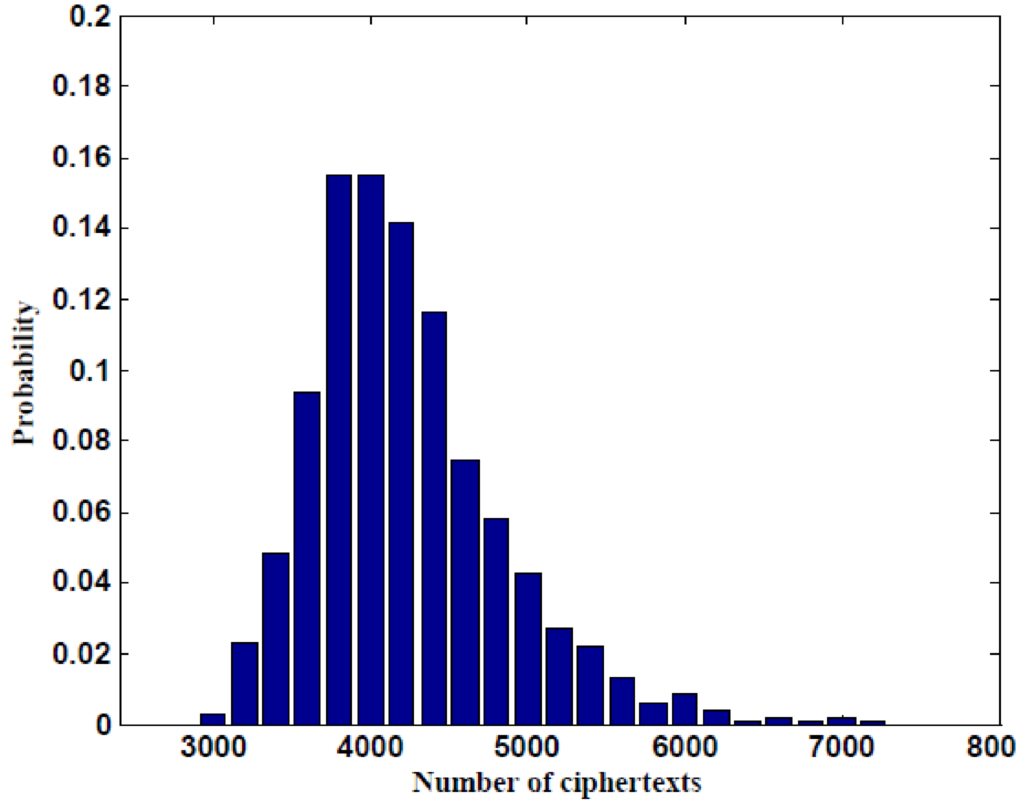
4. PFA on Countermeasures against FA

4.3 PFA on S-box (I1) with NCO/ZVO

- *p*, the probability that one plaintext can bypass IDDMR

$$p = (1 - \frac{1}{256})^{160} \approx 0.5346$$

- Only $p*N$ ciphertexts can be used in attacks
- The adversary requires N/p ≈ 1.8706*N encryptions (equivalent to REDMR)
- ξ=1000
- $3042 \leq N_f \leq 7141$
- $N_f \approx 4234$ on average
- If $n \geq 7200$, the success rate is 100%



4. PFA on Countermeasures against FA

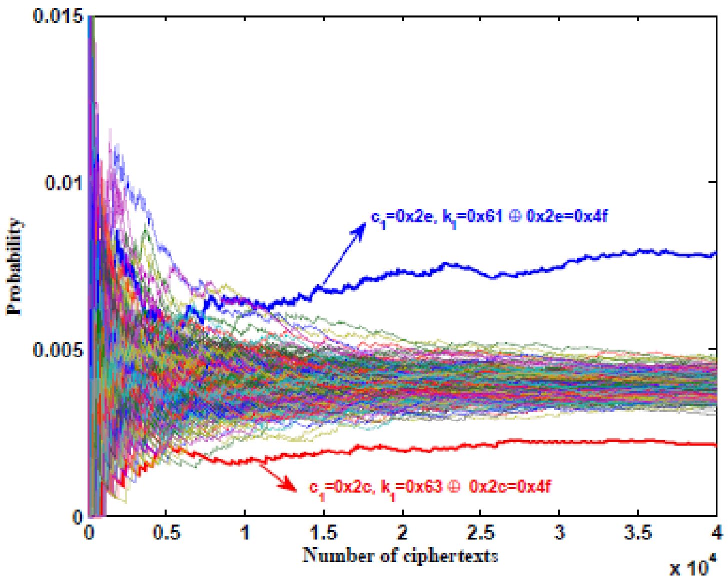
4.4 PFA on S-box (I1) with RCO

- No impossible values, however, the slight probability difference can still be detected

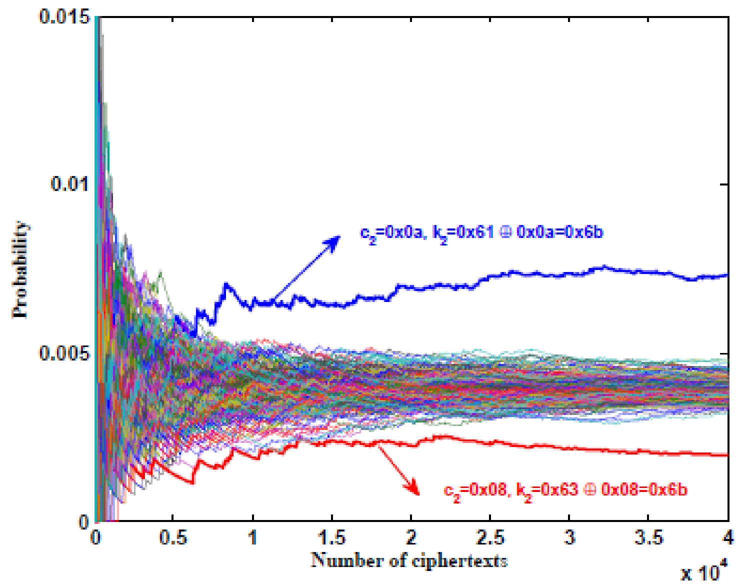
$$Pr(y = v) = 0 \times p + \frac{1}{256} \times (1 - p) = \frac{0.4654}{256}$$

$$Pr(y = v^*) = \frac{2}{256} \times p + \frac{1}{256} \times (1 - p) = \frac{1.5346}{256}$$

$$Pr(y \neq v \wedge y \neq v^*) = \frac{1}{256} \times p + \frac{1}{256} \times (1 - p) = \frac{1}{256}$$



(a) Extract k_1 using the distribution of c_1



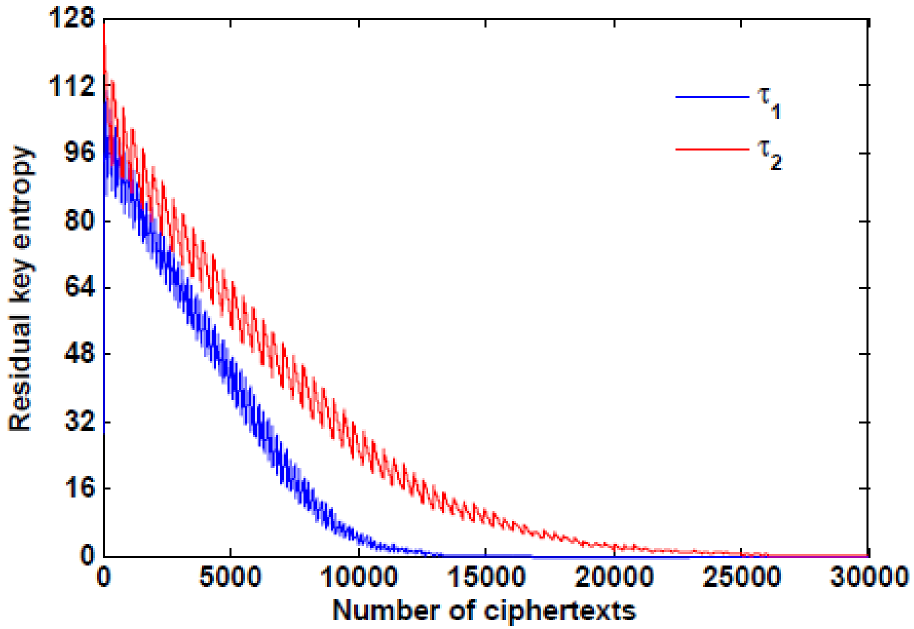
(b) Extract k_2 using the distribution of c_2

4.5 PFA on AES-128 with RCO using thresholds

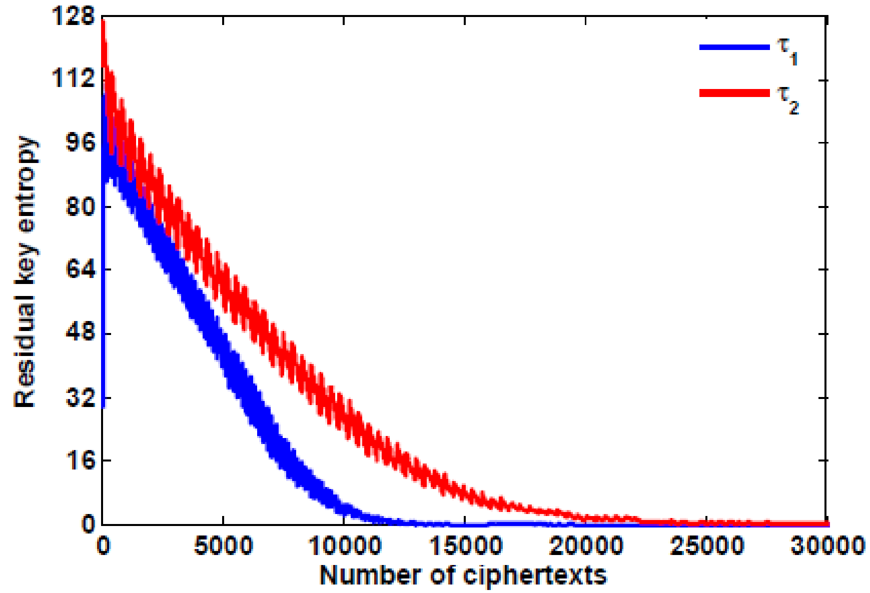
- Two thresholds to differentiate the abnormal cases
- Apply PFA on S-box (I1) and T-box (I2) implementation

$$\tau_1 = 0.9 \times \frac{1.5346}{256}$$

$$\tau_2 = 1.1 \times \frac{0.4654}{256}$$



(a) S-box implementation with RCO



(b) T-box implementation with RCO

OUTLINE

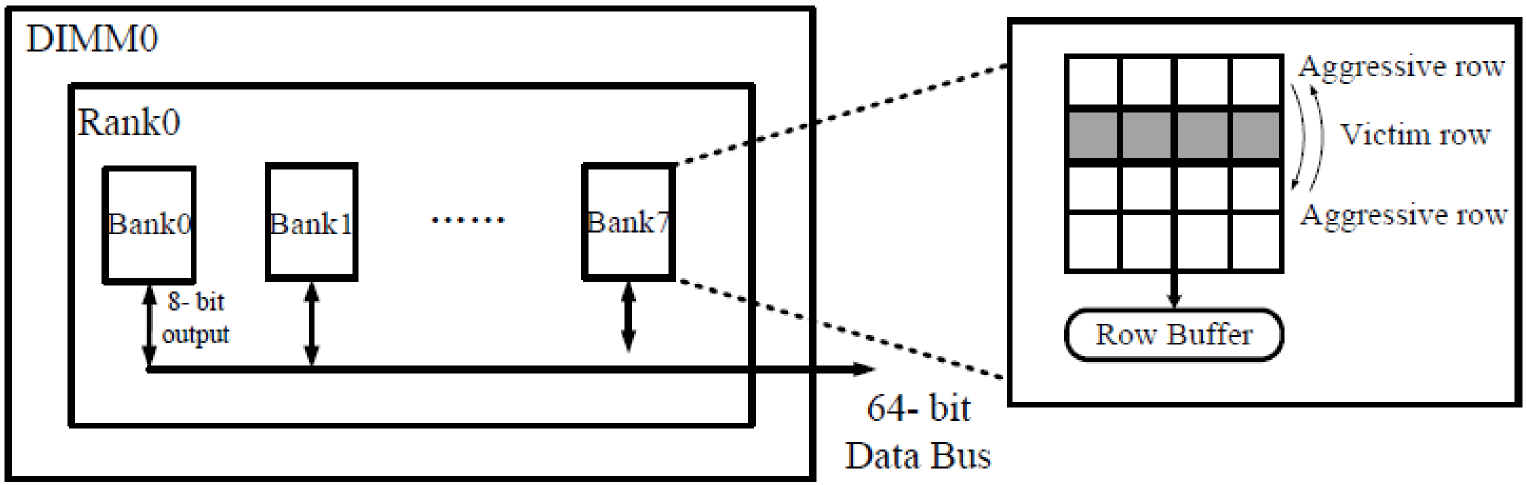
5.1 Background of Rowhammer techniques and shared libraries

Rowhamer vulnerability

- Appeared in 2014
- Frequent DRAM access leads to disturbance errors
- Hardware intrinsic, difficult to prevent
- Can be triggered from software (js, network)
- Can gain the privileges of ring0 without authorizations

Shared library

- Multiple processes shared one lib
- Dynamic load
- Read only at ring3 (user mode)
- Libgcrypt, OpenSSL, Crypto++, etc



5.3 Setup of our Rowhammer experiments

- Lenovo ThinkPad x230 laptop
  - Intel(R) Core(TM) i5-3320M at 2.60GHz
  - two Samsung DDR3 modules, 2GB at 1333MHz
  - Linux OS is Ubuntu 12.04 LTS, kernel version of 3.2.0-79 generic
- Libgcrypt v1.6.3
  - Compiled as shared library
  - GCC 4.6.3, No optimization
- T-box implementation (I3)
  - AES T-table T0 starts at the offset 000d6710h
  - T_0 is followed by the corresponding element of T'_0

```
000d6710h: C6 63 63 A5 63 00 00 00 F8 7C 7C 84 7C 00 00 00 ;
000d6720h: EE 77 77 99 77 00 00 00 F6 7B 7B 8D 7B 00 00 00 ;
000d6730h: FF F2 F2 0D F2 00 00 00 D6 6B 6B BD 6B 00 00 00 ;
000d6740h: DE 6F 6F B1 6F 00 00 00 91 C5 C5 54 C5 00 00 00 ;
000d6750h: 60 30 30 50 30 00 00 00 02 01 01 03 01 00 00 00 ;
000d6760h: CE 67 67 A9 67 00 00 00 56 2B 2B 7D 2B 00 00 00 ;
000d6770h: E7 FE FE 19 FE 00 00 00 B5 D7 D7 62 D7 00 00 00 ;
000d6780h: 4D AB AB E6 AB 00 00 00 EC 76 76 9A 76 00 00 00 ;
```

5. Case Study: Rowhammer-based PFA

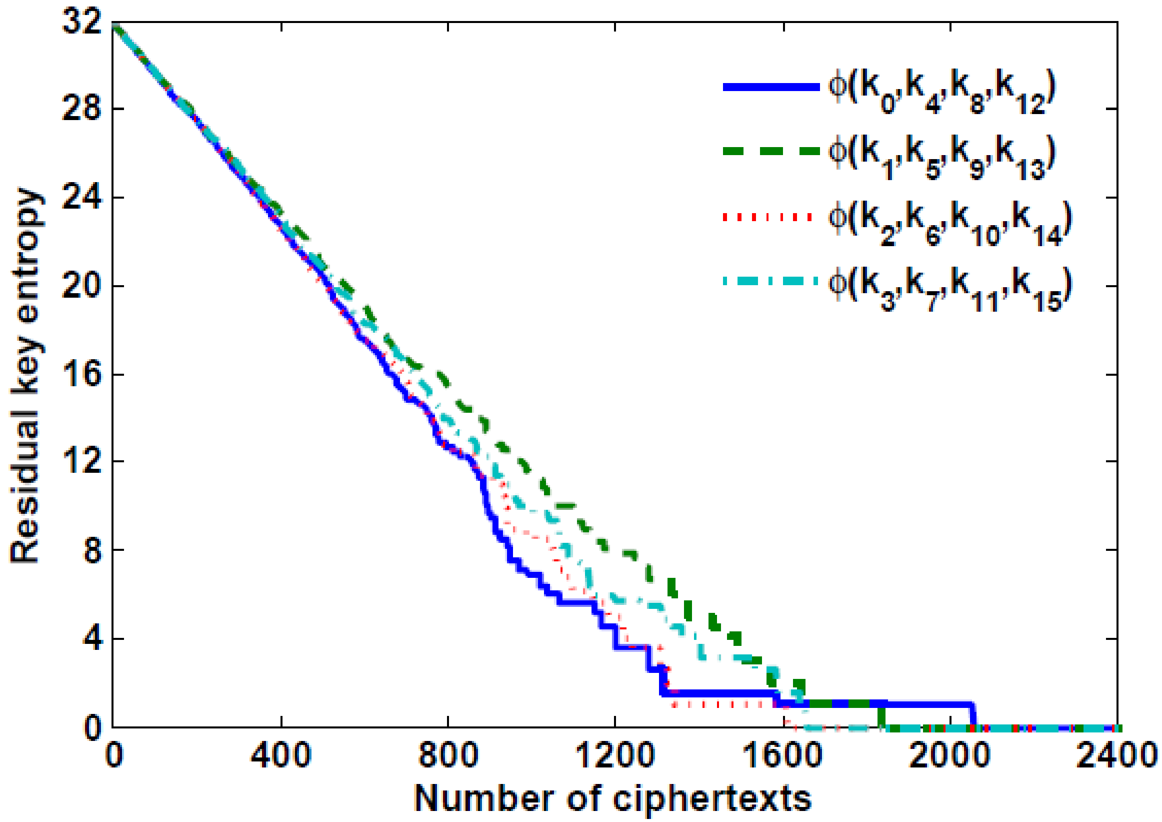
5.4 Results of Hammering

- Successfully inject one bit to any of T'_0, T'_1, T'_2, T'_3
  - Occur 5,4,6,5 times to T'0, T'1, T'2, T'3, in 90.80, 57.75, 49.83, 59.6 minutes respectively
- Ranging from 3 up to 230 minutes for the first 20 experiments
  - Facilitated with profiling
- It takes about 461 and 1367 minutes
  - Without profiling

ID	Attack time(min)	Location of flip	Data before injection	Data after injection
1	30	$T'_0[235]$	e900 0000	a900 0000
2	38	$T'_1[208]$	0070 0000	0050 0000
3	53	$T'_2[100]$	0000 4300	0000 4100
4	81	$T'_3[67]$	0000 001a	0000 0018
5	230	$T'_0[18]$	c900 0000	c800 0000
6	102	$T'_1[131]$	00ec 0000	00cc 0000
7	77	$T'_2[172]$	0000 9100	0000 9000
8	3	$T'_3[34]$	0000 0093	0000 0091
9	104	$T'_0[230]$	8e00 0000	8600 0000
10	49	$T'_2[126]$	0000 f300	0000 7300
11	86	$T'_3[101]$	0000 004d	0000 004c
12	75	$T'_3[55]$	0000 009a	0000 001a
13	17	$T'_2[221]$	0000 c100	0000 8100
14	44	$T'_1[67]$	001a 0000	0018 0000
15	53	$T'_3[147]$	0000 00dc	0000 00d8
16	5	$T'_0[108]$	0000 0050	0000 0010
17	41	$T'_2[252]$	0000 0f00	0000 0b00
18	62	$T'_2[140]$	0000 6400	0000 4400
19	47	$T'_1[13]$	00d7 0000	0097 0000
20	85	$T'_0[168]$	c200 0000	8200 0000
1	461(w/o profiling)	$T'_3[75]$	0000 00b3	0000 00f3
2	1367(w/o profiling)	$T'_1[163]$	000a 0000	0002 0000

5.5 Results of Analysis

- REDMR
- One injection can recover four bytes.
  - 4000 ciphertexts are collected
- At least four injections are required
- 8200 ciphertexts are required to recover the full key
  - 2050 ciphertexts per row



OUTLINE

6.1 Conclusion

- We propose persistent fault analysis
  - A novel attack on general block ciphers
  - Can defeat mainstream countermeasures against fault attacks
  - Can be used in different fault attacks with persistence
  - Different implementations
  - Different analysis strategies

- We conduct several evaluations
  - The attack is practically conducted in a shared library setting to target AES-128 in cryptographic library Libgcrypt

6.2 Future work

- More formal proofs on the theoretical estimation based on probabilities
  - Analog to Coupon Collector’s Problem
- Revisit the case for key scheduling
- Countermeasures design (Counter or health check)
- And more

Thank you very much!

Q and A

5.2 Overview of Rowhammer-based PFA on shared libraries

- Four steps
  - Profiling (optional)
  - Allocation
  - Positioning
  - Hammering
- Implication
  - One bit in the table will be flipped
  - Hammering one random bit in any T'_0 is possible
  - The bit flip is persistent for both encryptions and all rounds and viewed as faulty for careless users