TOHOKU UNIVERSITY

GSIS
Graduate School of Information Sciences,
TOHOKU University

April 15, 2014 COSADE2014

A Multiple-fault Injection Attack by Adaptive Timing Control under Black-box Conditions and a Countermeasure

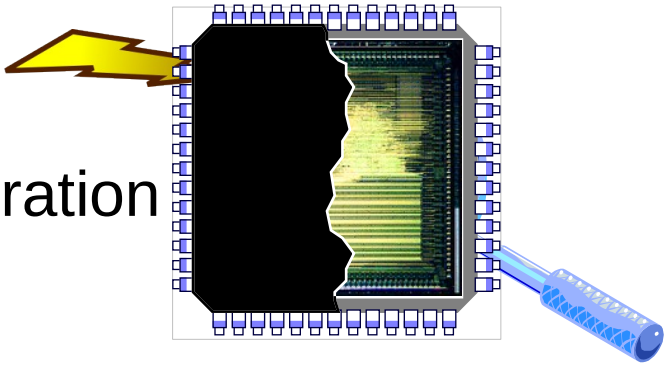
Sho Endo1, Naofumi Homma1, Yu-ichi Hayashi1,
Junko Takahashi2, Hitoshi Fuji2 and Takafumi Aoki1

1Tohoku University, Japan

2NTT Secure Platform Laboratories, Japan

GSIS, TOHOKU UNIVERSITY

Fault injection attacks against microcontrollers



- Fault Injection attacks
  - Injects faults in cryptographic operation
  - Obtain a secret key from faulty ciphertexts or other information
- Countermeasures against the attacks by software
  - Fault detection by recalculation
  - Adding random delay before encryption
- Multiple fault injection attacks in microcontrollers
  - Involves Multiple fault injections into single cryptographi c operation

Multiple fault injection attacks

- Experiments against RSA software
  - Injects faults into both encryption and recalculation
  - Power glitches [Kim 2007]
  - Laser shots [Trichina 2010]
  - Skips branch instruction in recalculation routine
- Conventional attacks were performed in a white-box setti ng
  - Execution timing of critical instructions are known
  - Black-box condition (execution timing is not known) w as not considered in literature

Goal of this work

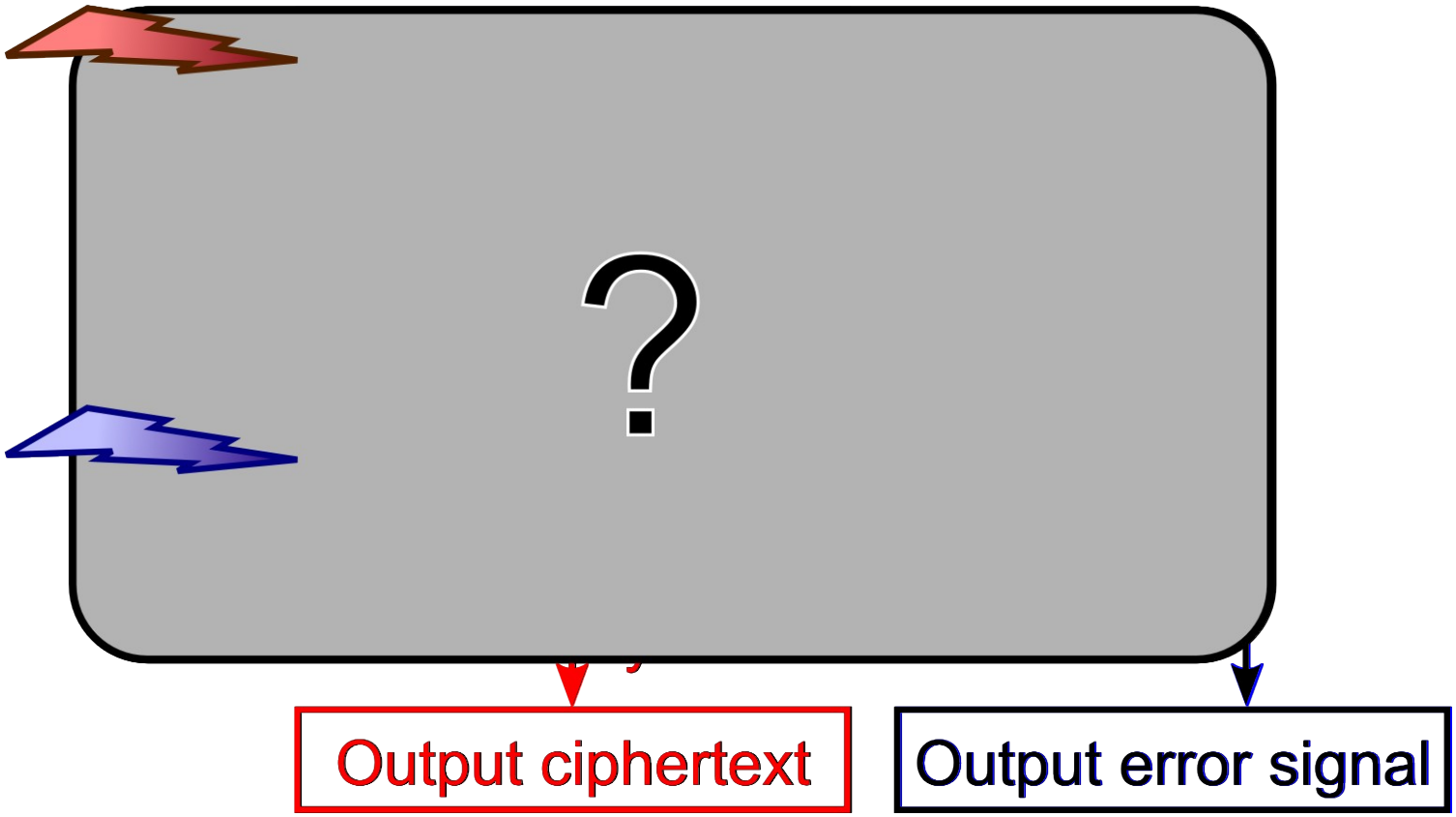
Investigating multiple-fault injection attack in black-box setting and countermeasure

- Scanning appropriate fault injection timing
  - Controlling fault injection timing adaptively according to the output of microcontrollers
  - Attack can be applied without knowledge of program
- An experiment of attack against AES with recalculation
  - Demonstrates that we can obtain faulty ciphertext for DFA
- Proposal of a countermeasure

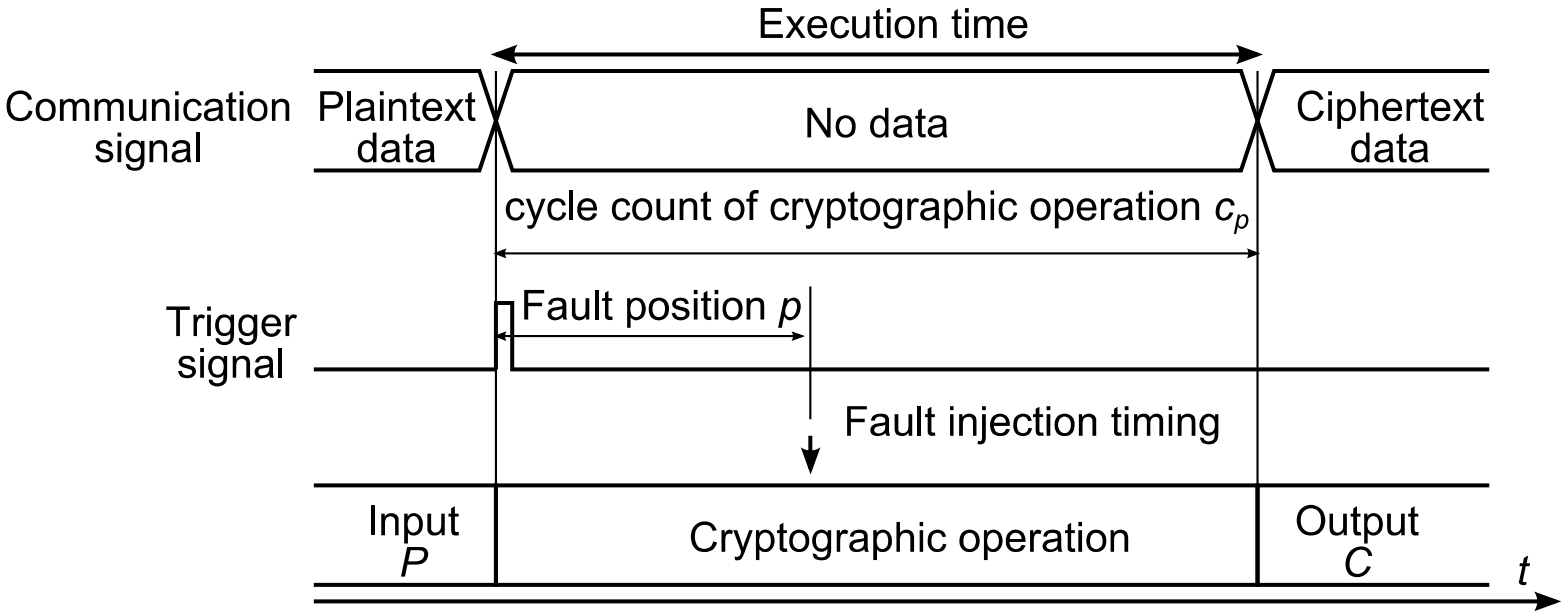
Outline

- Background
- Concept of the proposed attack
- Scanning algorithm
- Experiment of proposed attack against AES program wit h recalculation
- Countermeasure against the proposed attack
- Conclusion and future works

Multiple fault injection attack against recalculation



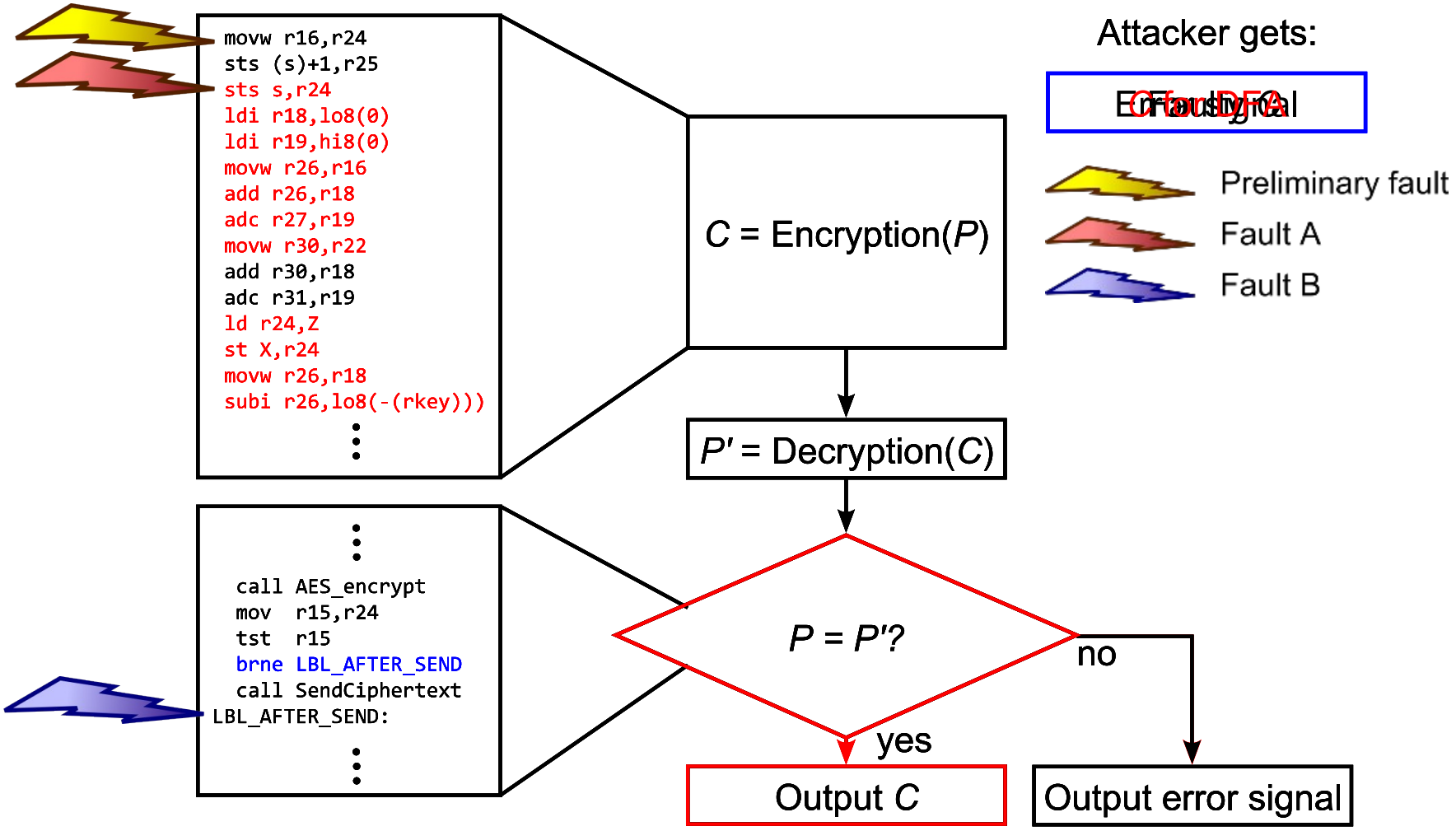
Assumption of our attack



Countermeasure by recalculation is present

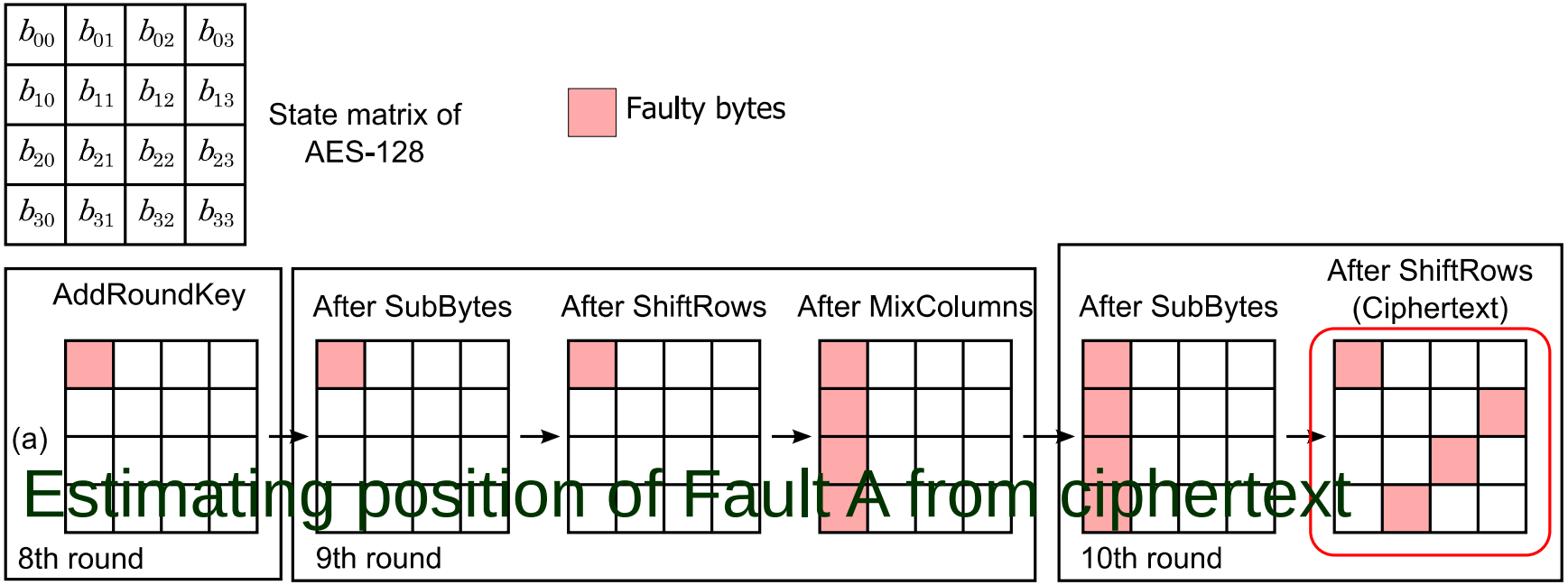
We can observe start and end timings of cryptographic operation through communication signal

Scanning fault timing



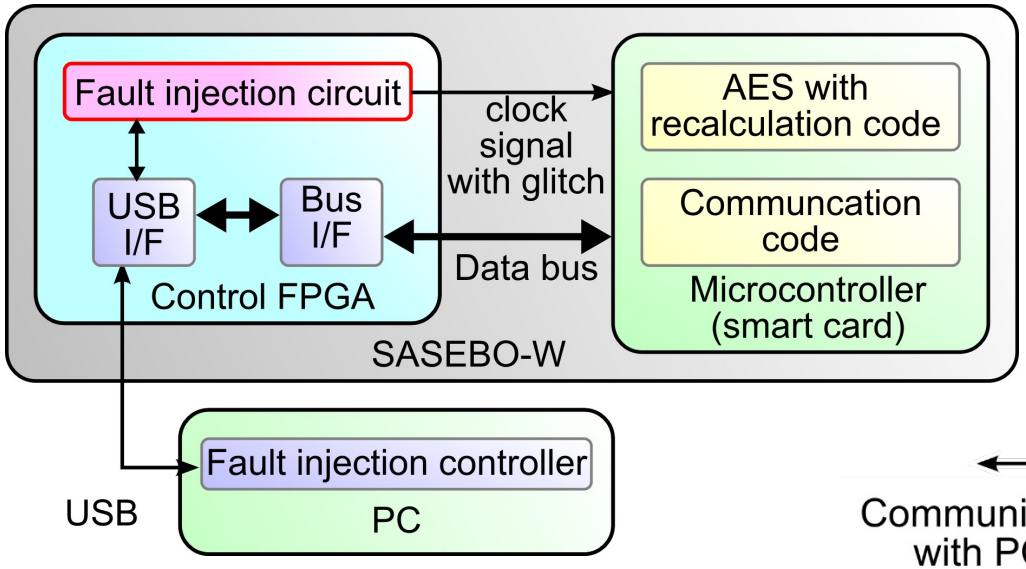
Obtaining faulty ciphertext for DFA

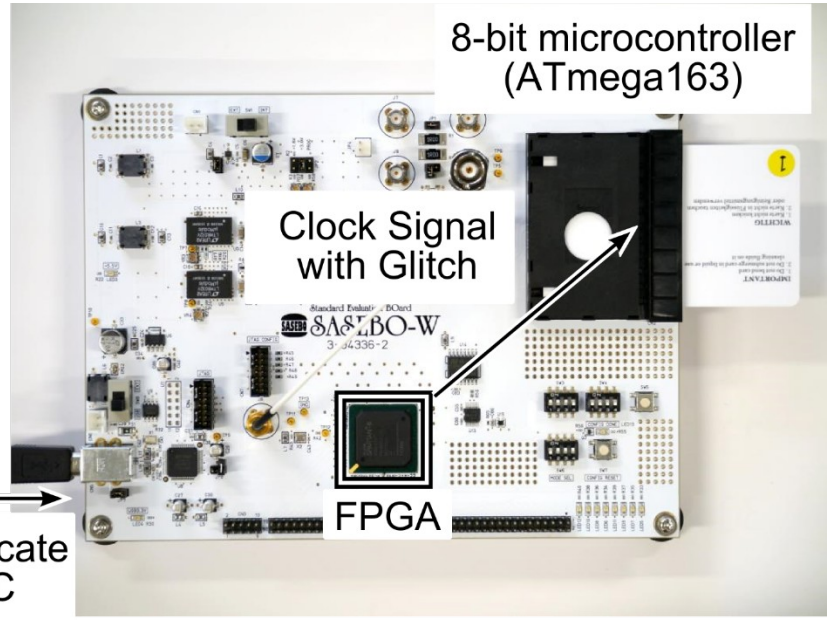
Example of faulty ciphertext



Experiment

- Experimental setup





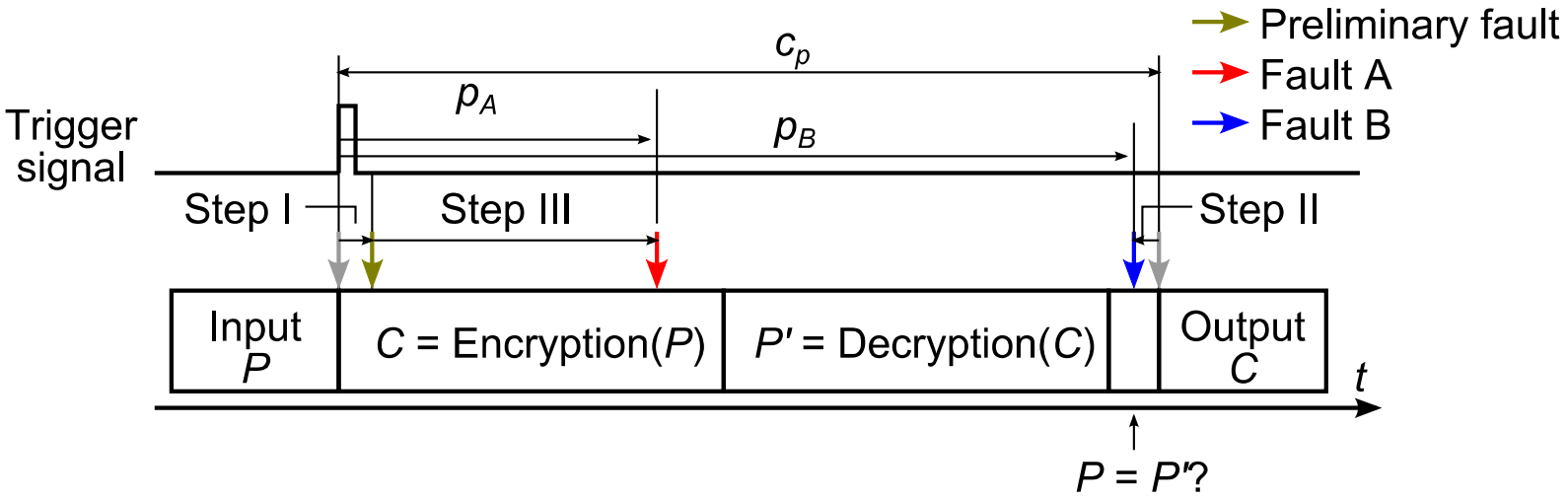
Experimental conditions

- Conditions

Cryptographic algorithm	**128-bit AES with recalculation**
Microcontroller	**AVR ATmega163 (8-bit)**
Compiler	**gcc 4.3.3 -Os**
FPGA	**Xilinx XC6SLX150**
Clock frequency of microcontroller	**3.6 MHz**
Plaintext	**(00112233445566778899aabbccddeeff)16**
Key	**(000102030405060708090a0b0c0d0e0f)16**

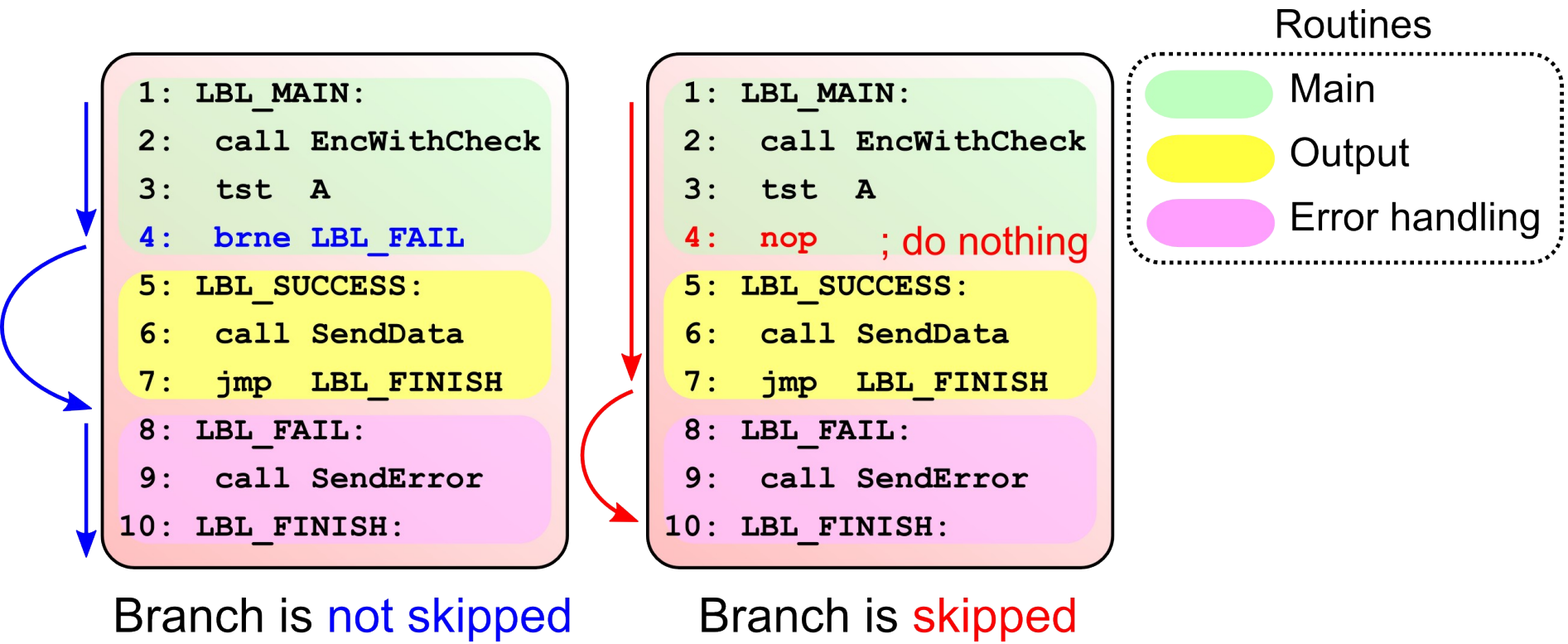
  - Can be exploited by Piret's DFA

Number of trials in our attack



Steps	Fault position		# of trials
	Start	End	
Step I (Scanning preliminary fault)	*pp*=0	*pp*=3	4
Step II (Scanning Fault B)	*pB*=30530(=*cp*)	*pB*=30527	4
Step III (Scanning Fault A)	*pA*=3	*pA*=9996	9994
Total			10002

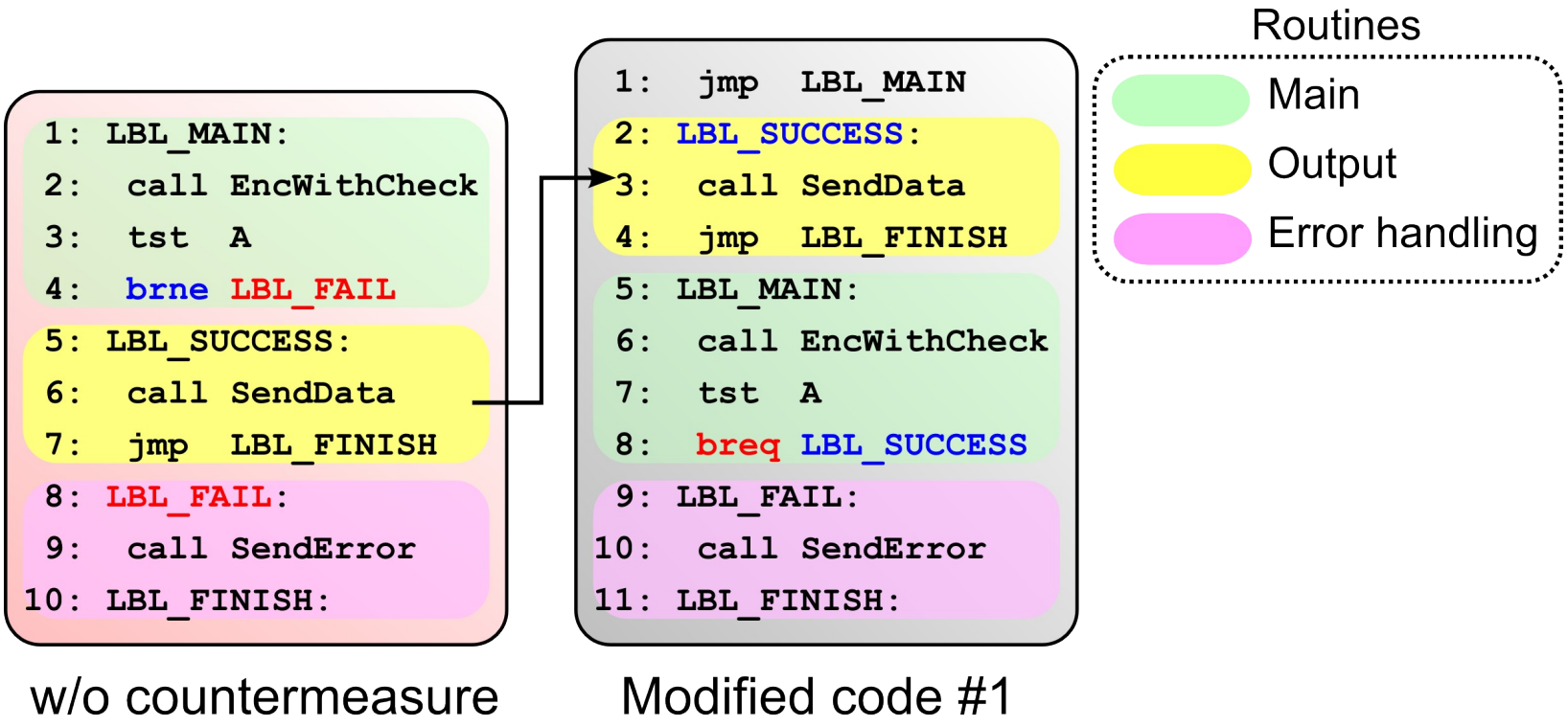
Instruction that was skipped in the experiment



Application of proposed attack

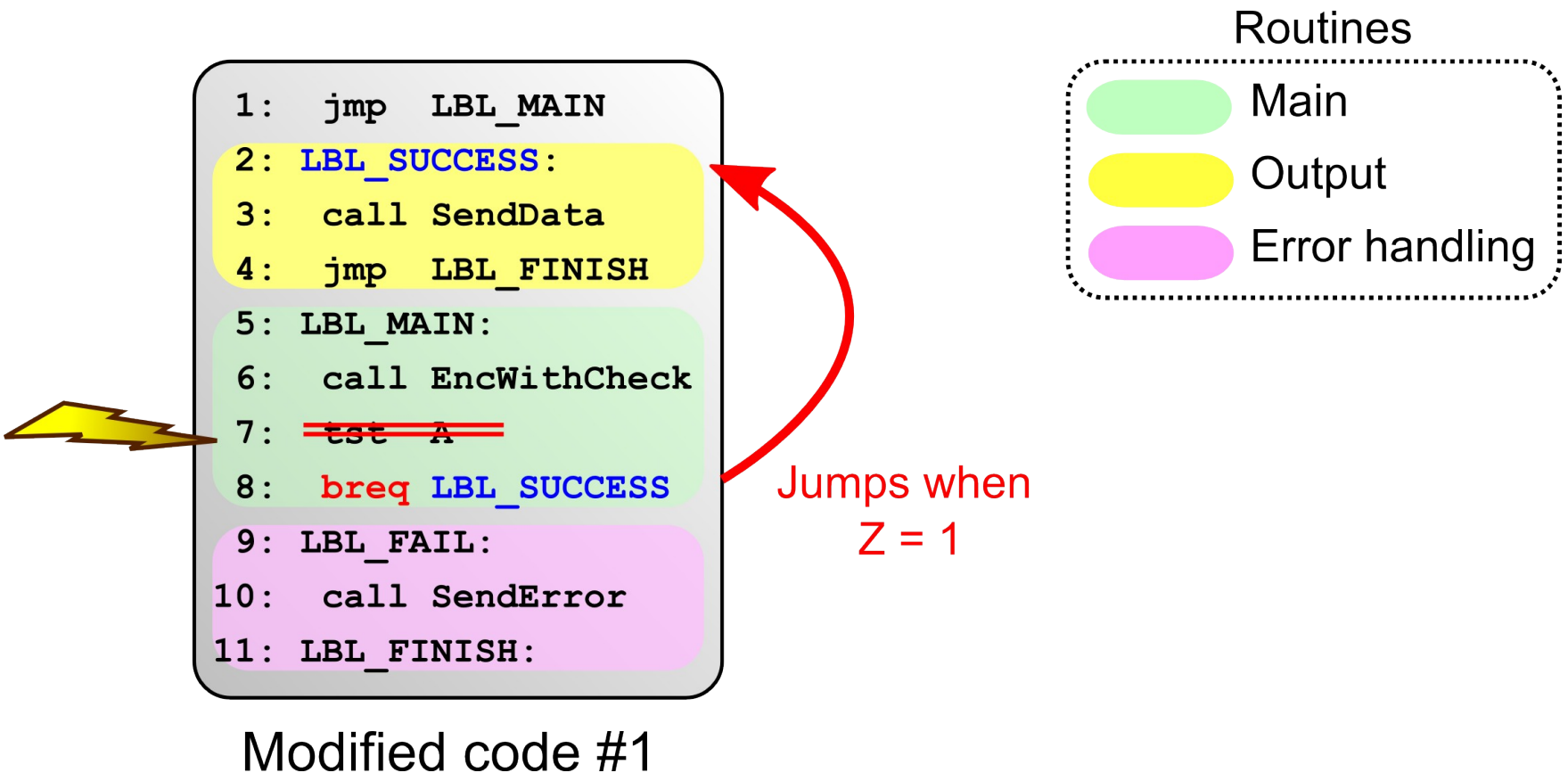
- Attacks against conventional countermeasures for fault injection
  - Duplication of instructions can be defeated by injecting faults into all the duplicated instructions
  - Random delay before the encryption can be defeated by skipping random number generation code
- Proposed countermeasure
  - Rearrange instructions of main function so that faulty ciphertext is not output when critical instructions are skipped

Countermeasure for the skip of branch instruction



- Output routine was moved to the address less than that of encrypt ion
- Branch condition was flipped

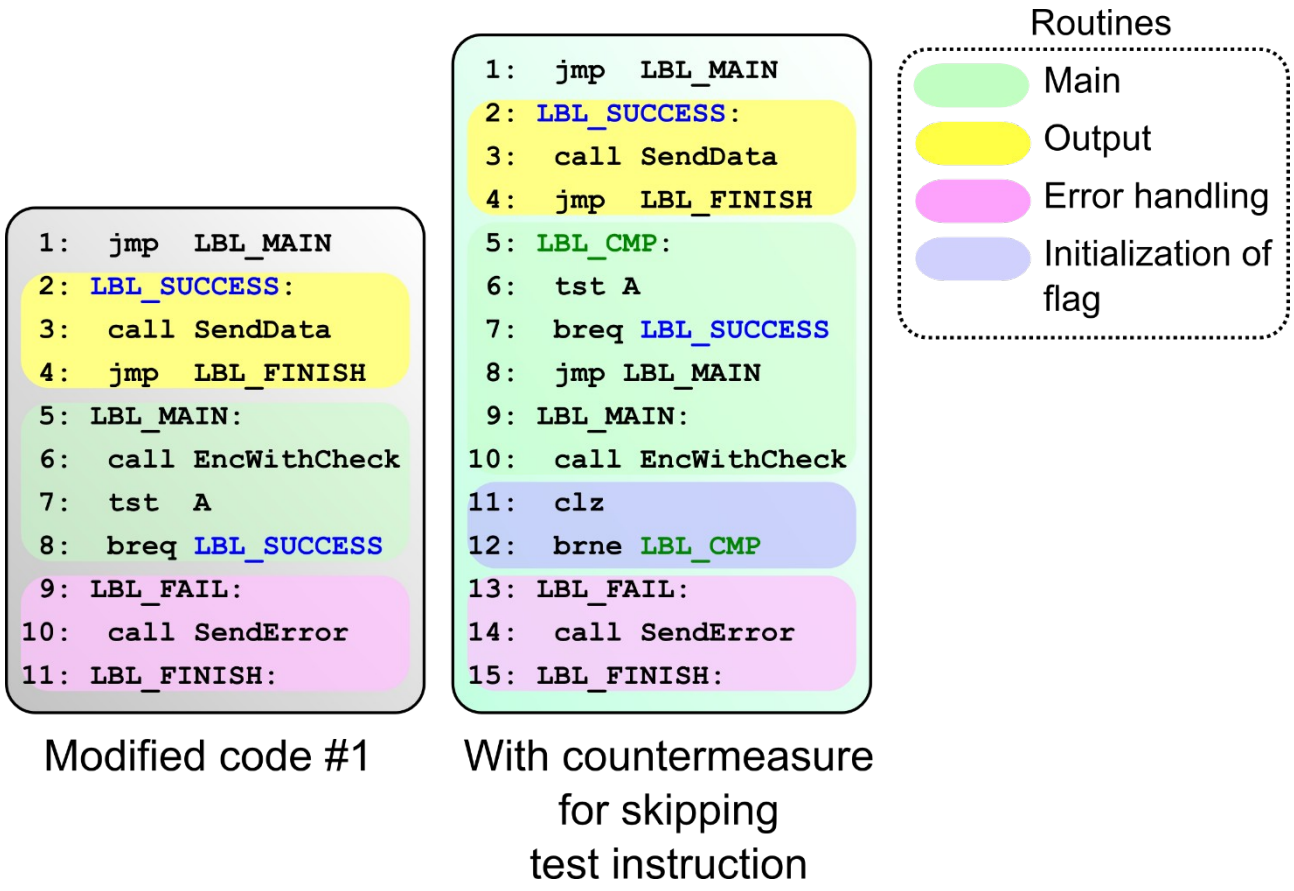
Attack on test (TST) instruction



Modified code #1

- Program may jumps to Line 2 when Line 7 was skipped and Z = 1

Proposed countermeasure



- Initialize Zero (Z) flag before executing test instruction

Conclusion and future works

- Proposal of scanning method to find appropriate fault positi on
  - Tuning the fault position adaptively according to output
- Experiment against AES program with recalculation
  - Successfully obtained faulty ciphertext
- Proposal of countermeasure against proposed attack
- Future works
  - Experiment on microcontrollers with other architectures
  - Implementation of compiler applies the proposed counte rmeasure automatically

Thank you!
Any questions?

Screenshot during fault injection