KOREA UNIVERSITY

Cryptographic Algorithm Laboratory

SDDT: An Operation Skip Attack Framework

for Bitslice Ciphers—Validated on PIPO

Dongwoo Kang [redr1102@korea.ac.kr]

2026/08/28

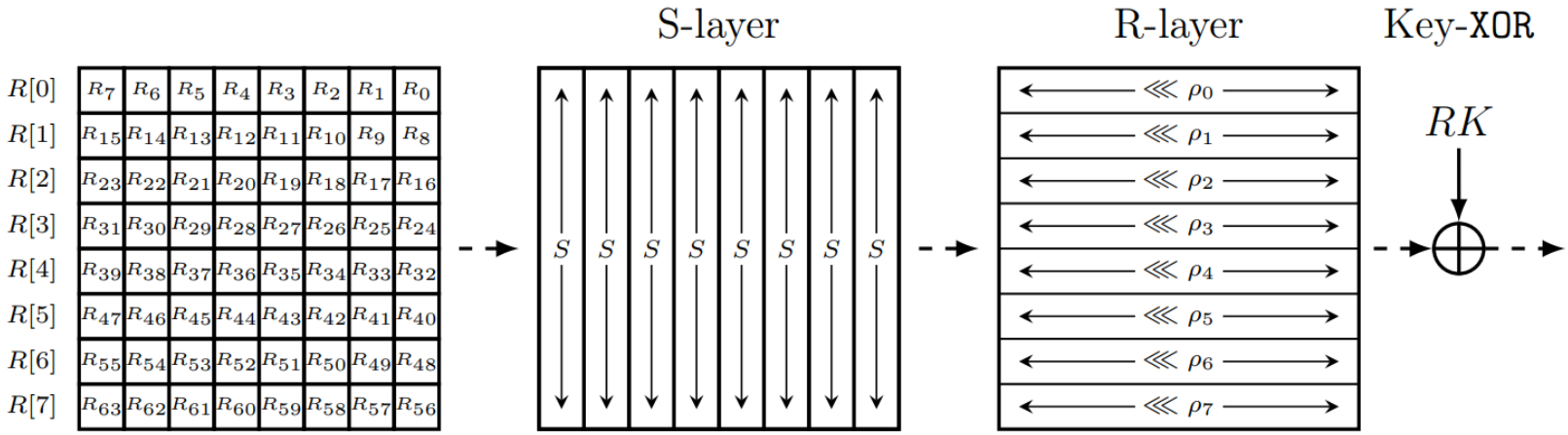
Contents

Introduction

- **Fault Attack**
  - Injecting faults (e.g. clock glitch, laser) into cryptographic operations may lead misbehavior
  - By analyzing this misbehavior, an adversary could recover secret key
- **Vulnerability of Bitslice Cipher**
  - If a fault is injected to the operations of S-Box, its misbehavior have some property
  - With these property, we propose an enhanced fault attack on PIPO

Attack Target: Bitslice Cipher PIPO

- **Settings**
  - 64-bit block, 128 or 256-bit key, 13 or 17 rounds respectively
- **Configuration**
  - S-layer: 8-bit S-box to each vertical column
  - R-layer: rotation to each parallel row



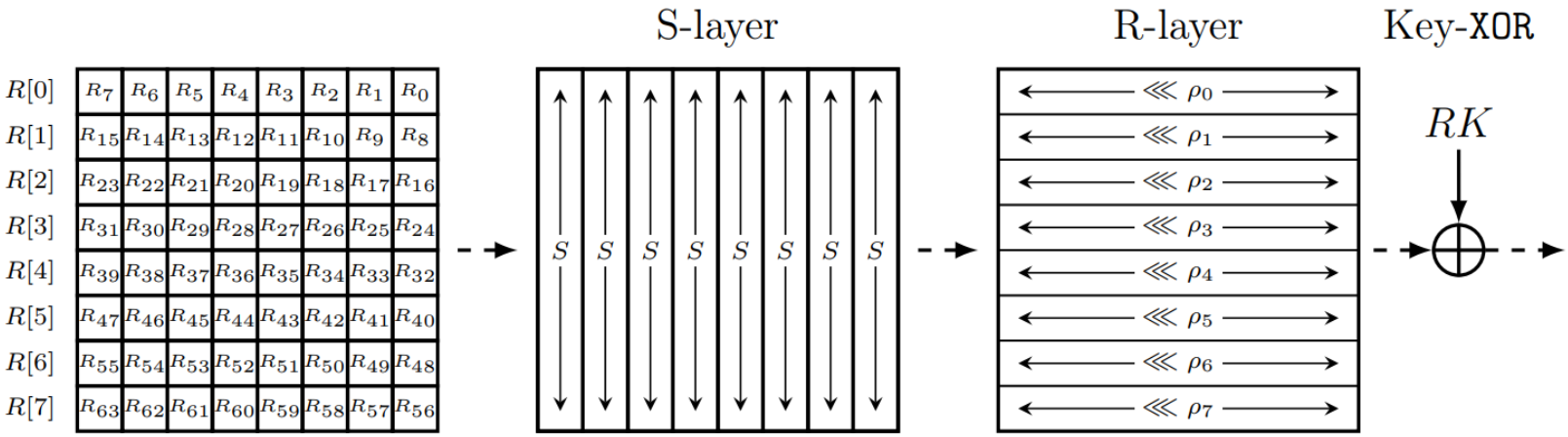
Round function of PIPO.

Attack Target: Bitslice Cipher PIPO

- Bitslice S-box
  - Instead of evaluating S-box with LUT,
  - apply logical operations on each registers
  - ① Constant-time, ② Hardware-friendly

```
void sbox(u8 *X)
{
        u8 T[3] = { 0, };
        //(MSB: x[7], LSB: x[0])
        // Input: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
        //S5_1
        X[5] ^= (X[7] & X[6]); (1)
        X[4] ^= (X[3] & X[5]); (2)
        X[7] ^= X[4]; (3)
        X[6] ^= X[3]; (4)
        X[3] ^= (X[4] | X[5]); (5)
        X[5] ^= X[7]; (6)
        X[4] ^= (X[5] & X[6]); (7)
```

Some part of PIPO S-box implementation.



Round function of PIPO.

Attack Tool

- **Main Idea**
  - If an adversary skip an operation during bitslice S-box evaluation,
  - What happens to the output difference?
- **Skip-induced Difference Distribution Table (SDDT)**
  - Brute-force for all possible skip indices, and for all 8-bit input $x \in \mathbb{F}_2^8$

Algorithm 3 Generation of SDDT

Require: An n-bit S-box $S(x)$
Require: Set of skip faults $\{E_1, \ldots, E_N\}$
Ensure: SDDT of size $N \times 2^n$ ▷ N denotes the number of operations of S-box

```
SDDT[N][2^n] ← 0                                  ▷ Initialize matrix with zeros
for each operation index i from 1 to N do         ▷ Iterate over all operation indices
    for x = 0 to 2^n − 1 do                       ▷ Iterate over all possible x
        Δy ← S(x) ⊕ S_{E_i}(x)
        SDDT[i][Δy] ← SDDT[i][Δy] + 1             ▷ Increment the counter for Δy
    end for
end for
return SDDT
```

Attack Tool

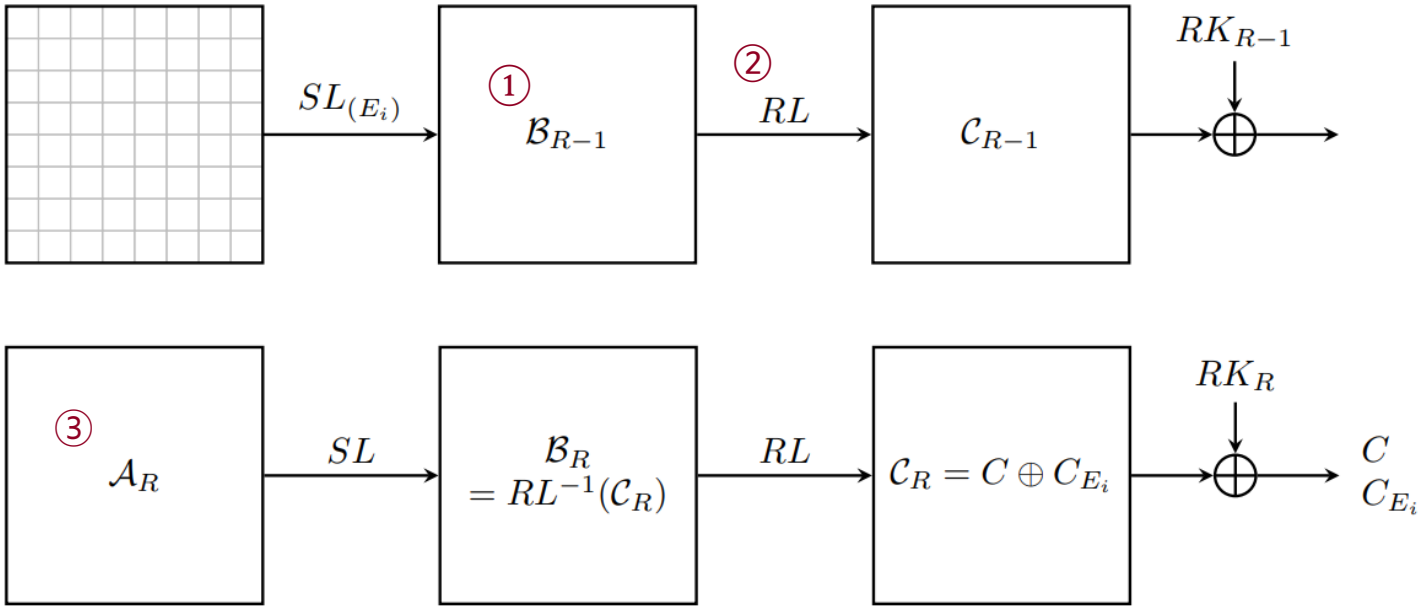
- SDDT of PIPO
  - For instance, if an adversary skip the 11th operation, 0x46, 0x5E, 0xC6, 0xDE would appear with equal probability

Op #	Output Differences ΔY_i (Count)
1	0x00(192), 0x22(4), 0x25(8), 0x26(2), 0x36(2), 0x3A(4), 0x64(2), 0x6C(2), 0xAE(2), 0xBD(8), 0xBE(2), 0xE0(4), 0xE3(4), 0xEB(4), 0xF3(4), 0xF4(2), 0xF8(4), 0xFB(4), 0xFC(2)
2	0x00(192), 0x17(8), 0x37(8), 0x8B(8), 0x97(8), 0x9B(8), 0xA3(4), 0xAB(4), 0xAF(8), 0xB3(4), 0xBB(4)
3	0x00(128), 0x0B(16), 0x17(16), 0x1B(16), 0x23(8), 0x2B(8), 0x33(8), 0x3B(8), 0x3F(16), 0x97(16), 0xA7(16)
4	0x00(128), 0x0C(16), 0x1C(16), 0x24(16), 0x34(16), 0x8C(16), 0x9C(16), 0xAC(16), 0xBC(16)
5	0x00(64), 0x42(48), 0x5A(48), 0xC2(48), 0xDA(48)
6	0x00(128), 0x0E(8), 0x12(32), 0x1E(8), 0x22(16), 0x26(8), 0x36(8), 0x3A(16), 0x8E(8), 0x9E(8), 0xAE(8), 0xBE(8)
7	0x00(192), 0xA0(16), 0xA8(16), 0xB0(16), 0xB8(16)
8	0x00(192), 0x43(8), 0x47(8), 0x4F(8), 0x5B(8), 0xC3(8), 0xC7(8), 0xCF(8), 0xDB(8)
9	0x00(64), 0x23(8), 0x2B(8), 0x2F(8), 0x33(8), 0x3B(8), 0x3F(8), 0xA2(32), 0xA7(8), 0xAA(32), 0xB2(32), 0xB7(8), 0xBA(32)
10	0x00(64), 0x05(48), 0x85(48), 0x89(48), 0x99(48)
11	0x46(64), 0x5E(64), 0xC6(64), 0xDE(64)
12	0x00(128), 0x05(32), 0x09(32), 0x19(32), 0x85(32)
13	0x00(128), 0x42(32), 0x5A(32), 0xC2(32), 0xDA(32)
14	0x00(128), 0xA0(32), 0xA8(32), 0xB0(32), 0xB8(32)
15	0x00(128), 0x04(32), 0x08(32), 0x18(32), 0x84(32)
16	0x00(128), 0x02(32), 0x1A(32), 0x82(32), 0x9A(32)
17	0x00(128), 0x80(32), 0x88(32), 0x90(32), 0x98(32)
18	0x00(192), 0x0C(16), 0x1C(16), 0x8C(16), 0x9C(16)
19	0x00(128), 0x04(64), 0x84(64)
20	0x00(64), 0x08(128), 0x18(64)
21	0x00(128), 0x02(64), 0x82(64)
22	0x00(64), 0x10(192)
23	0x00(192), 0x80(64)

Full SDDT of PIPO S-box.

Attack Scenario

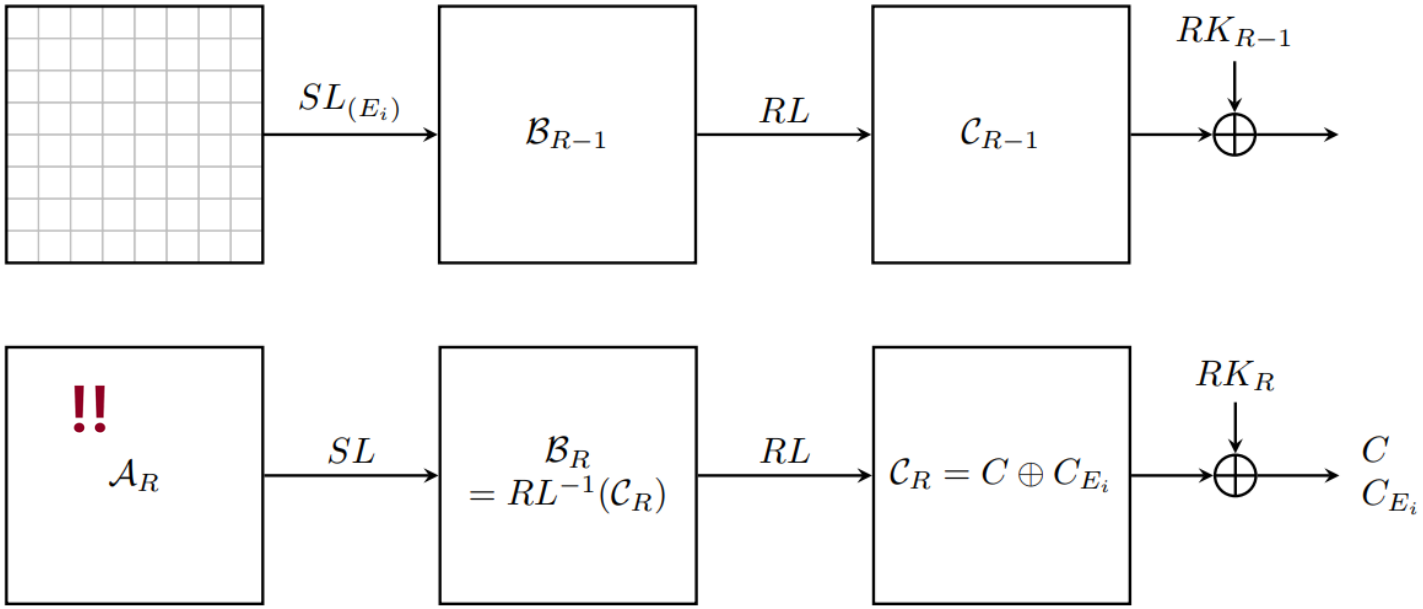
- Fault Propagation
  - Skipping fault is injected to $R-1$ round S-box
  - $\mathcal{A}, \mathcal{B}, \mathcal{C}$ denote 64-bit differences of each locations
  - ① $\mathcal{B}_{R-1}$ arises from SDDT, ② propagates via R-layer, ③ occurs at $\mathcal{A}_R$
  - ⇒ Difference pattern of $\mathcal{A}_R$



Last two rounds of PIPO.

Attack Scenario

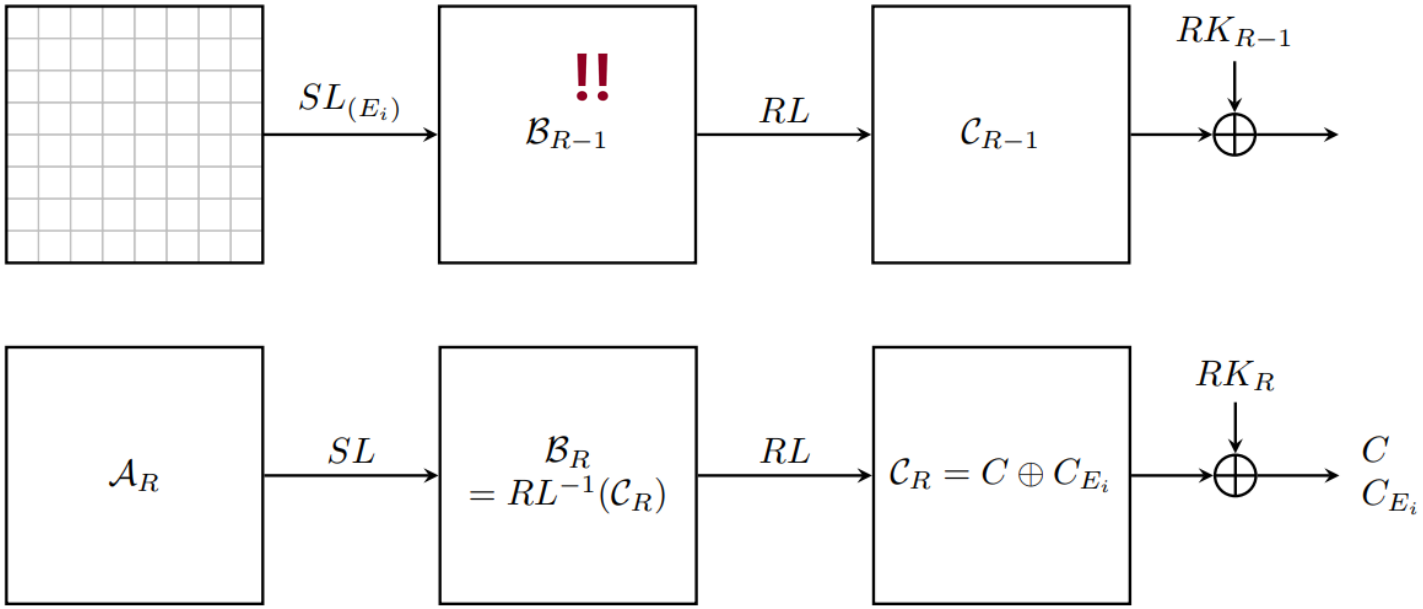
- Guessing RK_R Key Byte Filtering
  - With guessing 8-bit of RK_R, an adversary could recover vertical 8-bit of $\mathcal{A}_R$
  - If the recovered 8-bit is not matching with difference pattern, guessed key would be wrong.



Last two rounds of PIPO.

Attack Scenario

- Guessing RK_{R-1} Key Byte Filtering
  - After recovering RK_R, one round decrypt and filter RK_{R-1}
  - Regardless of RK_{R-1}, 64-bit difference $\mathcal{C}_{R-1}$ and also $\mathcal{B}_{R-1}$ are fixed



Last two rounds of PIPO.

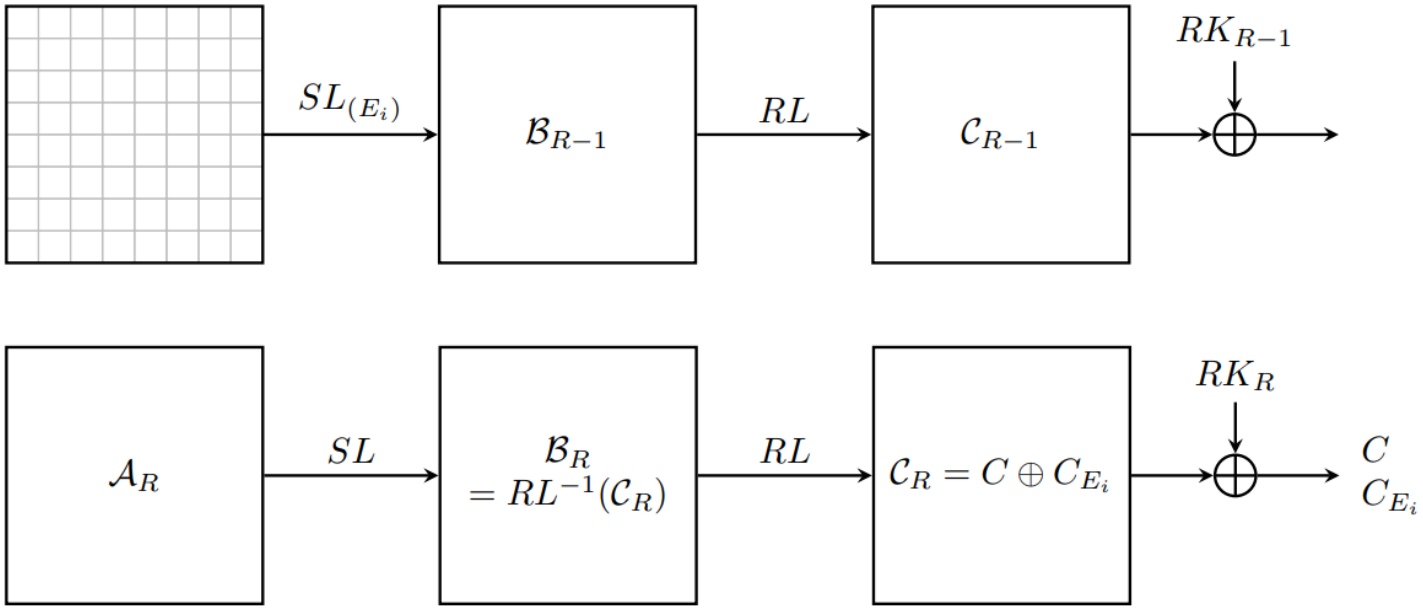
Attack Scenario

- Guessing RK_{R-1} Key Byte Filtering
  - With SDDT, an adversary could filter 8-bit value to generate the difference pattern
  - 11th operation for instance, only 64 values could make 0x46, and others so.
  - By filtering the internal value, the adversary filter RK_{R-1}

Op #	Output Differences ΔY_i (Count)
1	0x00(192), 0x22(4), 0x25(8), 0x26(2), 0x36(2), 0x3A(4), 0x64(2), 0x6C(2), 0xAE(2), 0xBD(8), 0xBE(2), 0xE0(4), 0xE3(4), 0xEB(4), 0xF3(4), 0xF4(2), 0xF8(4), 0xFB(4), 0xFC(2)
2	0x00(192), 0x17(8), 0x37(8), 0x8B(8), 0x97(8), 0x9B(8), 0xA3(4), 0xAB(4), 0xAF(8), 0xB3(4), 0xBB(4)
3	0x00(128), 0x0B(16), 0x17(16), 0x1B(16), 0x23(8), 0x2B(8), 0x33(8), 0x3B(8), 0x3F(16), 0x97(16), 0xA7(16)
4	0x00(128), 0x0C(16), 0x1C(16), 0x24(16), 0x34(16), 0x8C(16), 0x9C(16), 0xAC(16), 0xBC(16)
5	0x00(64), 0x42(48), 0x5A(48), 0xC2(48), 0xDA(48)
6	0x00(128), 0x0E(8), 0x12(32), 0x1E(8), 0x22(16), 0x26(8), 0x36(8), 0x3A(16), 0x8E(8), 0x9E(8), 0xAE(8), 0xBE(8)
7	0x00(192), 0xA0(16), 0xA8(16), 0xB0(16), 0xB8(16)
8	0x00(192), 0x43(8), 0x47(8), 0x4F(8), 0x5B(8), 0xC3(8), 0xC7(8), 0xCF(8), 0xDB(8)
9	0x00(64), 0x23(8), 0x2B(8), 0x2F(8), 0x33(8), 0x3B(8), 0x3F(8), 0xA2(32), 0xA7(8), 0xAA(32), 0xB2(32), 0xB7(8), 0xBA(32)
10	0x00(64), 0x05(48), 0x85(48), 0x89(48), 0x99(48)
11	0x46(64), 0x5E(64), 0xC6(64), 0xDE(64)
12	0x00(128), 0x05(32), 0x09(32), 0x19(32), 0x85(32)
13	0x00(128), 0x42(32), 0x5A(32), 0xC2(32), 0xDA(32)
14	0x00(128), 0xA0(32), 0xA8(32), 0xB0(32), 0xB8(32)
15	0x00(128), 0x04(32), 0x08(32), 0x18(32), 0x84(32)
16	0x00(128), 0x02(32), 0x1A(32), 0x82(32), 0x9A(32)
17	0x00(128), 0x80(32), 0x88(32), 0x90(32), 0x98(32)
18	0x00(192), 0x0C(16), 0x1C(16), 0x8C(16), 0x9C(16)
19	0x00(128), 0x04(64), 0x84(64)
20	0x00(64), 0x08(128), 0x18(64)
21	0x00(128), 0x02(64), 0x82(64)
22	0x00(64), 0x10(192)
23	0x00(192), 0x80(64)

Attack Scenario

- Target Operation Selection
  - If zero-difference occurs at some column of $\mathcal{A}_R$, an adversary cannot filter wrong key
  - 11th operation is the only not to make zero-difference



Last two rounds of PIPO.

Attack Scenario

- **Fault Propagation from the Target Operation**
  - As mentioned, skipping 11th operation generates only 4 patterns — 0x46, 0x5E, 0xC6, 0xDE
  - By analyzing propagation through RL, 5 bits are strictly restricted
  - If recovered value does not suffice this restriction, guessed key would be wrong

Possible Input Differences				RL	Output difference
0x46	0x5E	0xC6	0xDE		
0	0	1	1	→	?
1	1	1	1	→	1
0	0	0	0	→	0
0	1	0	1	→	?
0	1	0	1	→	?
1	1	1	1	→	1
1	1	1	1	→	1
0	0	0	0	→	0

Difference propagation through RL

Attack Scenario

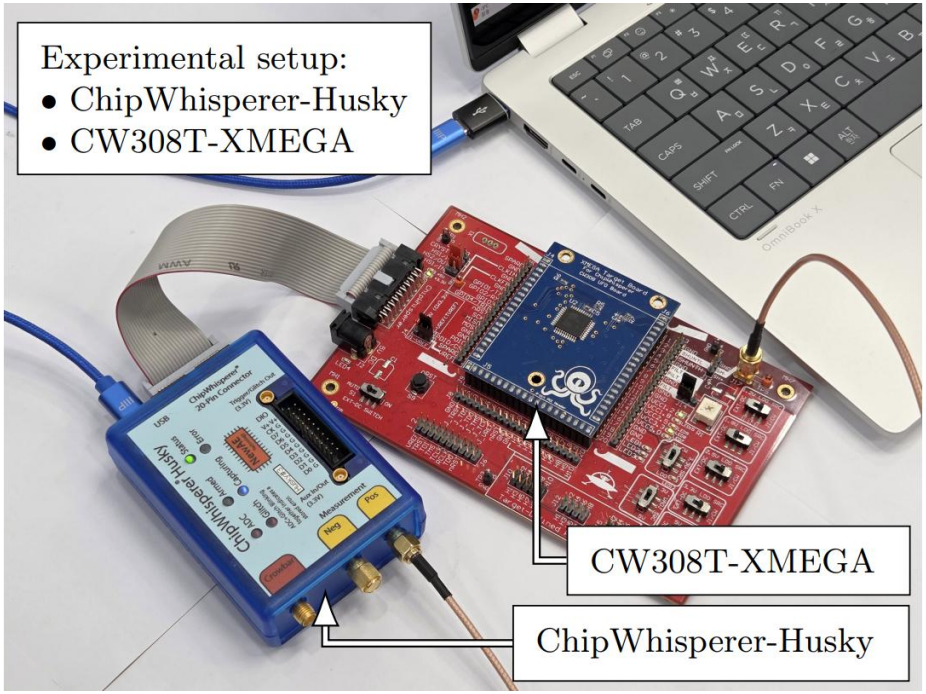
- **Achieving Robustness against Invalid Fault Injections**
  - In real world, targeting the specific operation precisely would be difficult
  - Two strategies to acheive robustness
- **Fault Profiling**
  - With 8 possible input patterns, the output pattern is restricted

$$\mathbb{D}_{\text{impossible}} = \{\texttt{0x00}, \texttt{0x04}, \texttt{0x09}, \texttt{0x0D}, \texttt{0x19}, \\ \texttt{0x1D}, \texttt{0x82}, \texttt{0x86}, \texttt{0xCA}, \texttt{0xFC}\}.$$

  - If one of these patterns occurs, it would be wrong faulty cipher, and reject
- **Voting Scheme**
  - Not strictly deny the filtered key, vote to the ‘not filtered’ keys

Experimental Results

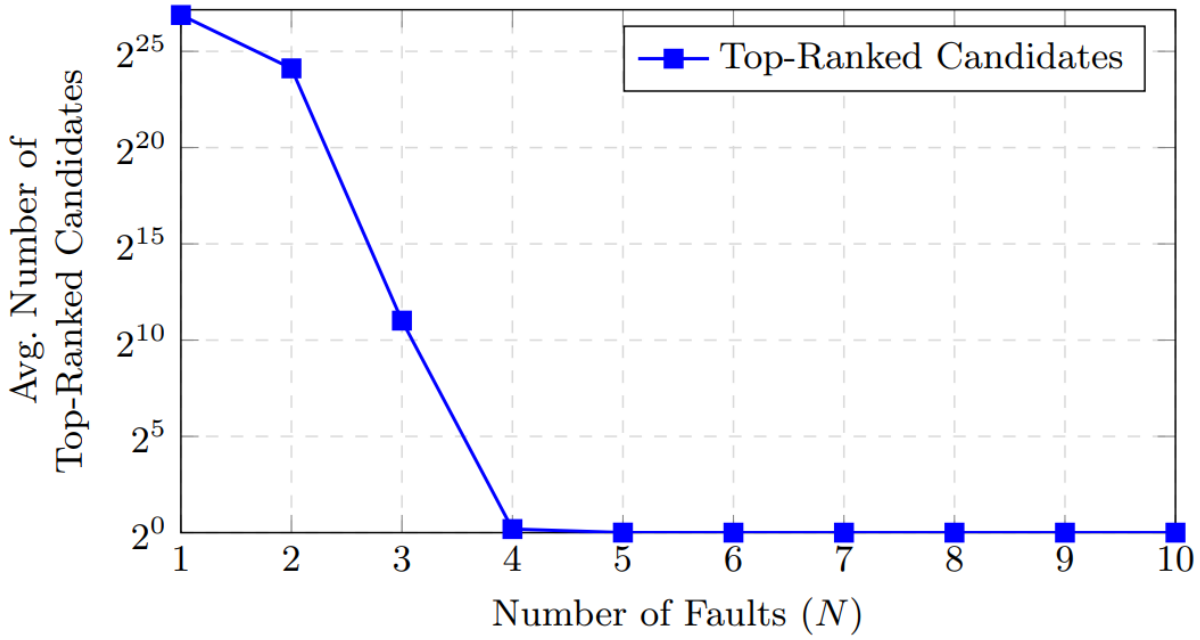
- Setup
  - Utilized ChipWhisperer-Husky and 8-bit MCU board
  - Reference C implementation



Experimental Results

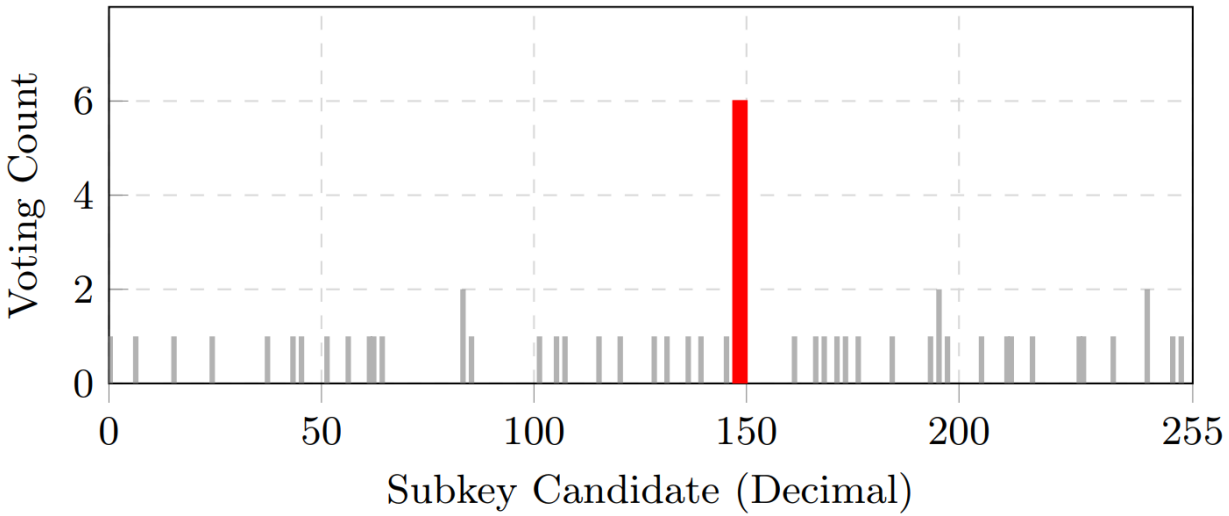
- RK_R Guessing Entropy over Number of Faults
  - With these parameters, we achieved 98.85% of fault injection success rate
  - Unique key recovery at $N = 6$

Parameter	Value
ext_offset	13
offset	50
width	150
repeat	1



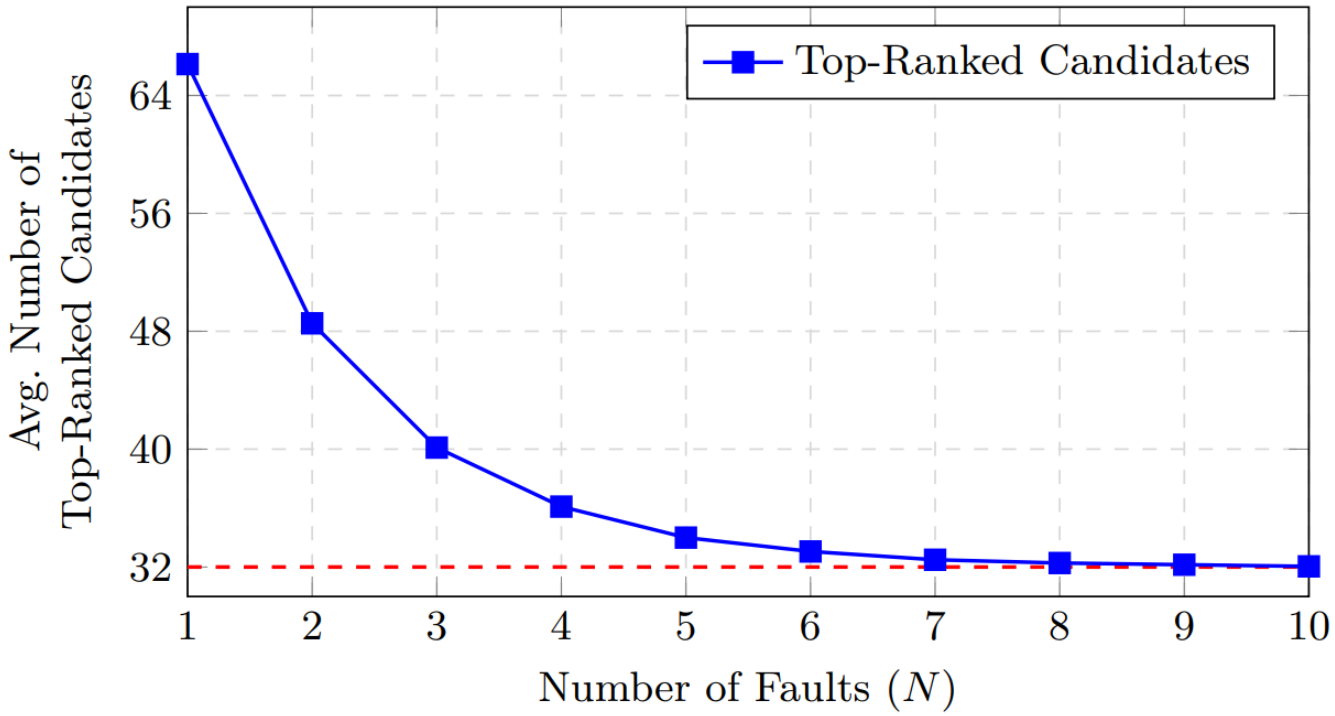
Experimental Results

- Voting Score Results
  - True key is 0x95 (149_{10})
  - Leveraging 6 faults, correct key is uniquely identified with maximum score
  - Score of wrong keys shrinks



Experimental Results

- RK_{R-1} Guessing Entropy over Number of Faults
  - Bounded on $32 = 2^5$ key space
  - Recall that only 3 bits differs on the $\mathcal{B}_{R-1}$ pattern, so only 3 bits of RK_{R-1} column could be determined



Conclusion & Discussions

- Fault Attack on Bitslice Cipher PIPO
  - Utilizing the output difference pattern to recover secret key
  - Deeper round to fault injection, fewer number of faults than previous works

Target Cipher	Reference	Fault Model	Fault Loc.	# Faults
PIPO-64/128	[24]	Bit-flip	Round 12 and 13	64
PIPO-64/256	[24]	Bit-flip	Round 16 and 17	128
PIPO-64/128	[23]	Random Byte	Round 12 and 13	≤ 32
PIPO-64/128	[34]	Random Byte	Round 13 and Key schedule	$4^{\dagger}$
PIPO-64/128	**Ours**	**Operation Skip**	**Round 11 and 12**	$\mathbf{\leq 10}$
PIPO-64/256	**Ours**	**Operation Skip**	**Round 15 and 16**	$\mathbf{\leq 20}$

Conclusion & Discussions

- **Countermeasures**
  - Inserting random delay to prevent accurate fault injection
  - Masking
- **SIFA-like Attack**
  - On fault detection, an adversary cannot acquire faulty ciphertexts
  - But using zero-difference (ineffective fault) like SIFA, would be still vulnerable
  - (Future Work)

Q&A

Thanks

KOREA UNIVERSITY

Cryptographic Algorithm Laboratory