Free Fault Leakages for Deep Exploitation: **Algebraic Persistent Fault Analysis on Lightweight Block Ciphers**

Fan Zhang, Tianxiang Feng, Zhiqi Li, Kui Ren and Xinjie Zhao

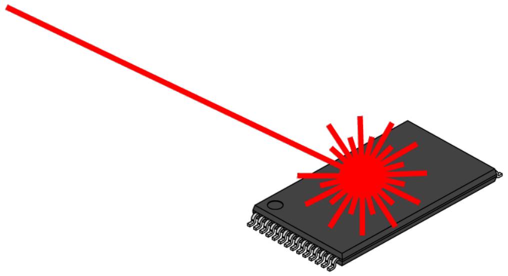
CHES 2022

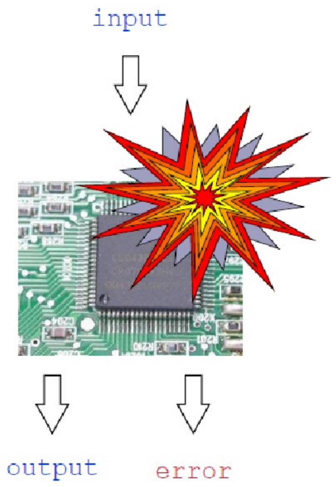
浙江大學
ZHEJIANG UNIVERSITY

OUTLINE

1.1 What are fault attacks

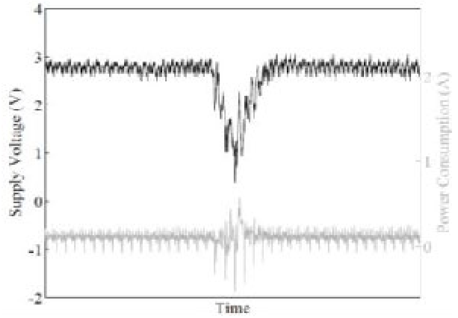
- Fault Attack (FA) first proposed by Boneh et al in 1996
- Active attacks against cryptographic implementations
- Two stages: fault injection and fault analysis

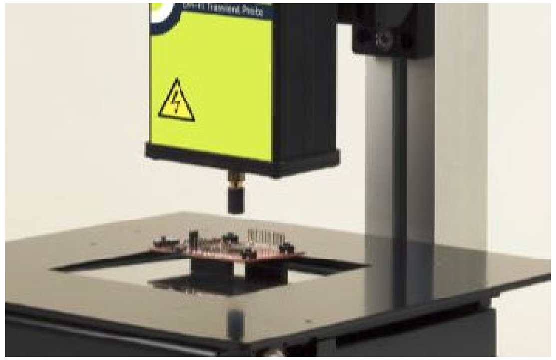




1.2 Fault Attack -- Fault Injection

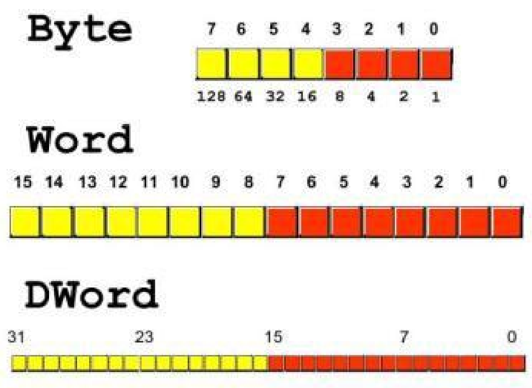
- Techniques
  - Clock Glitch
  - Voltage Spike
  - EM Pulse
  - Optical Laser
- Categories
  - Non-invasive
  - Semi-invasive
  - Invasive
- Mostly target some special intermediate
- Location and timing



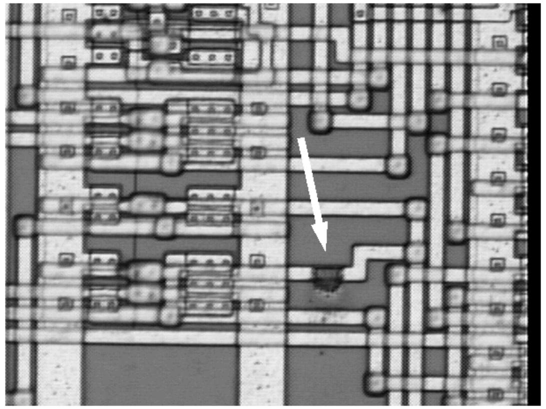


1.3 Fault model

- Granularity: how many bits are affected (fault width)
- Modification (fault type)
  - Stuck-at, e.g. zero or one
  - Flip
  - Random
- Control: on the fault location and on timing
  - Precise
- Duration or effect of the fault
  - Transient
  - Persistent
  - Permanent



adopted from Josep Balasch in IACR Summer School 2015

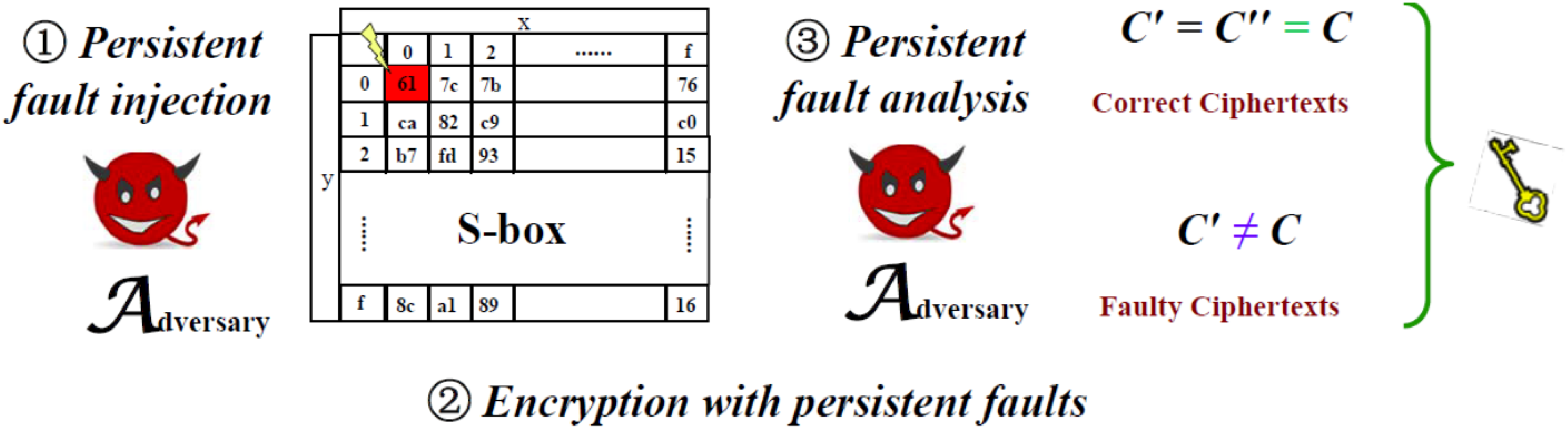


1.4 Persistent Fault Analysis (PFA)

➢ Fault Model

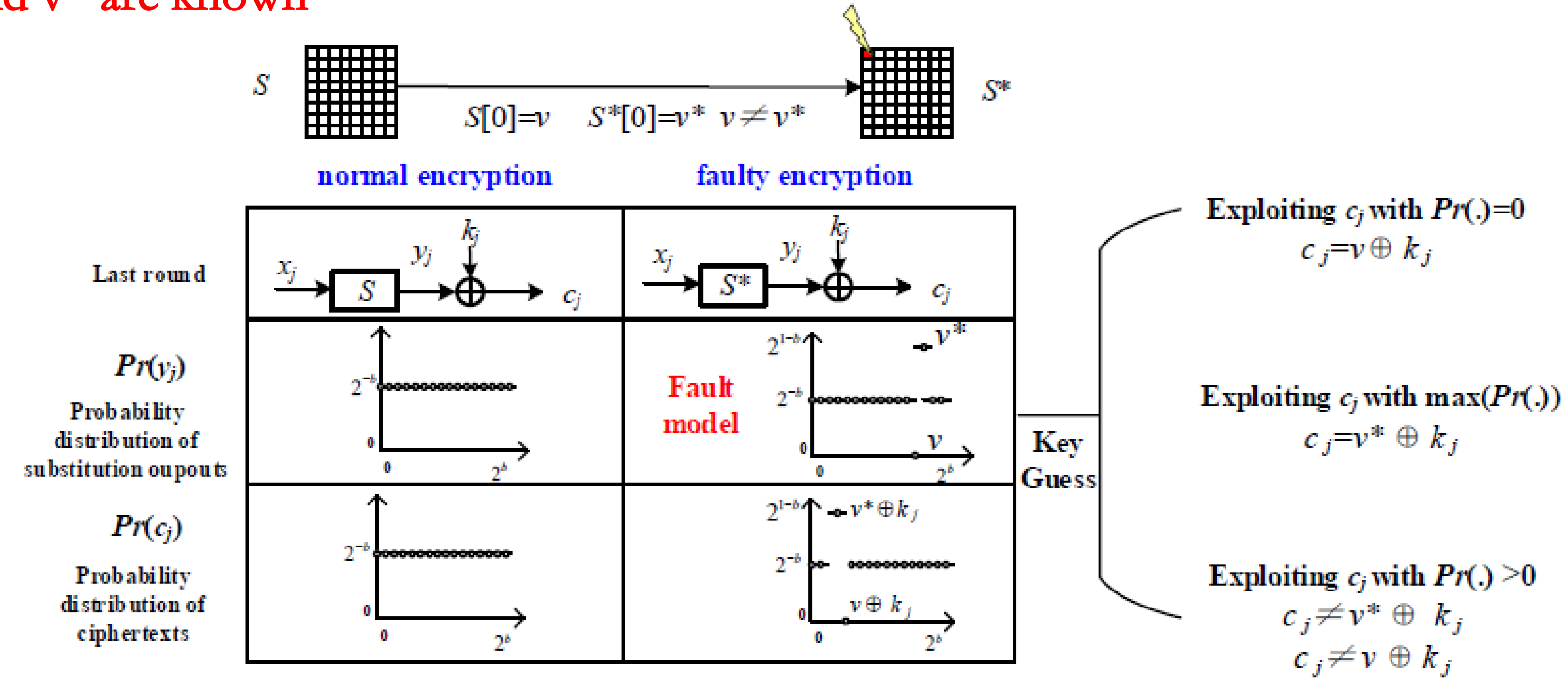
- The adversary can inject faults before the encryption of a block cipher
  - Typically, these faults alter a stored algorithm constant, e.g., S-box
- The injected faults are persistent
  - The affected constant stays faulty unless refreshed
  - All iterations are computed with the faulty constant
-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

1.5 Core Idea of Persistent Fault Attack



1.6 PFA on AES

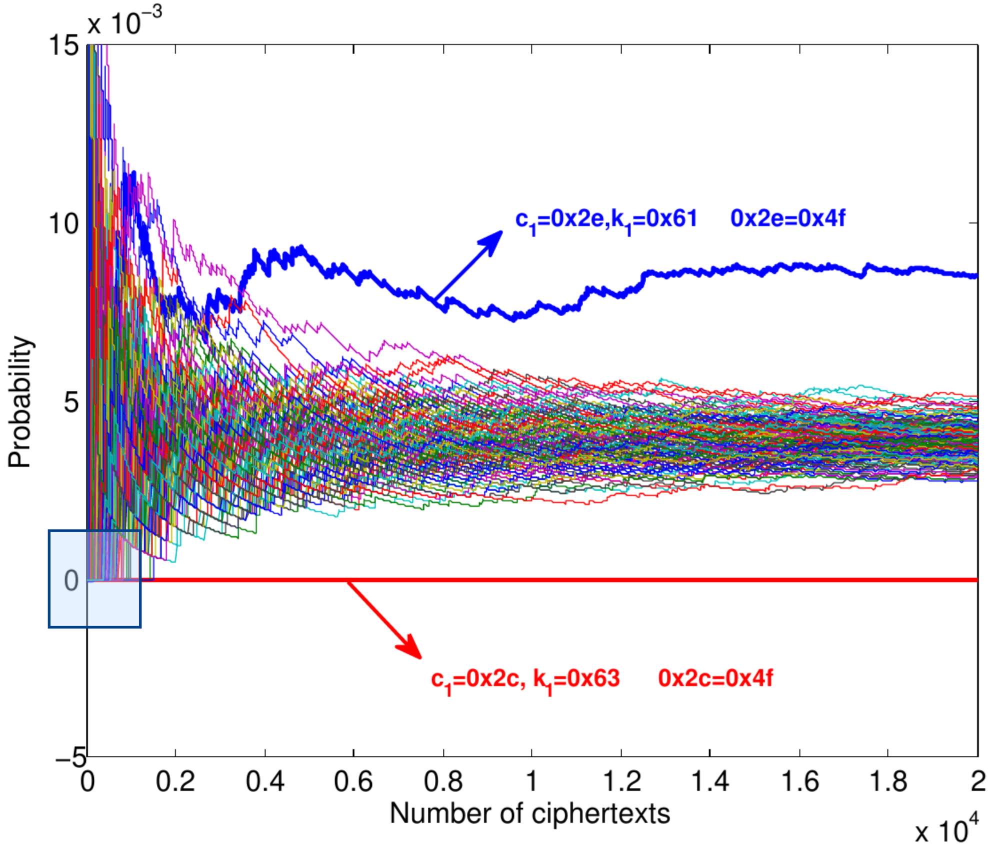
- A statistical analysis on the last round, exploiting three types of fault leakages
- v and v* are known



1.6 PFA on AES

Example:

- Through statistical method, we can clearly see two distinct curves
  - A blue curve represents the byte with higher frequency
  - A red curve represents the byte with lower frequency
- Both of them can be used to recover the key byte

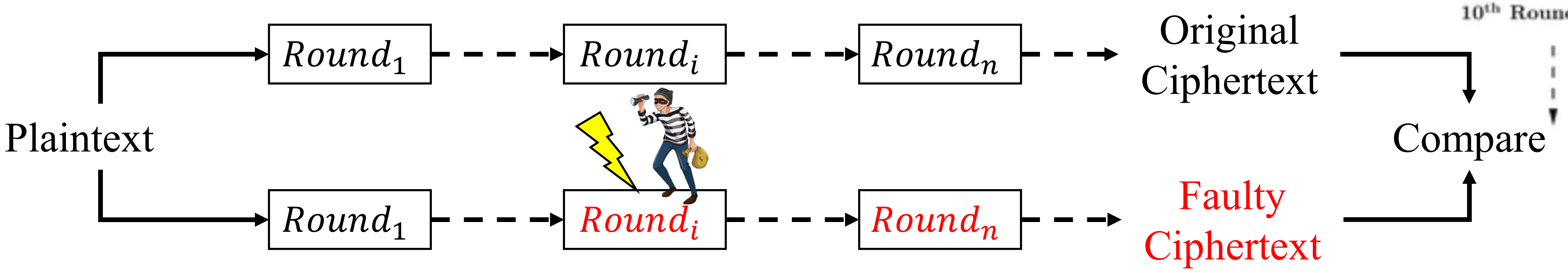


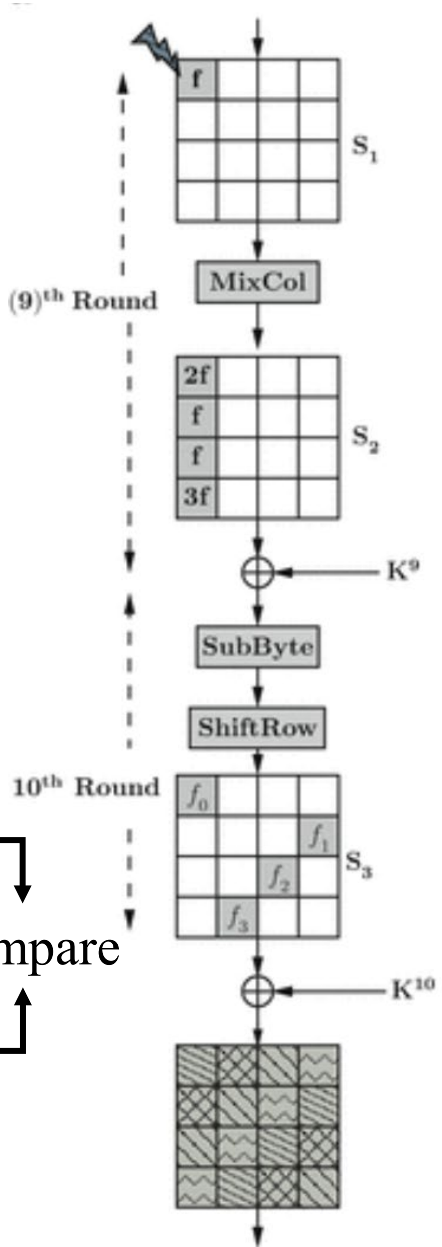
(a) Extract k_1 using the distribution of c_1

OUTLINE

2.1 Motivation and Ideas

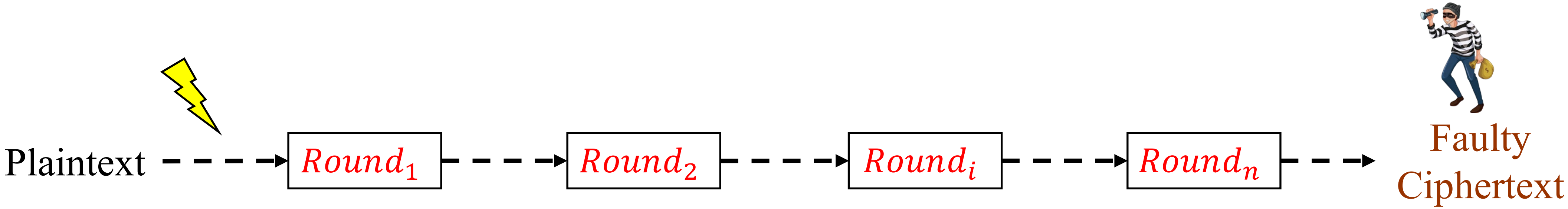
- Differential Fault Analysis（DFA）
  - After injecting the fault, the fault starts to propagate at the fault injection location
  - One fault injection for one exploitation of fault leakages
  - The deeper the fault injection goes, the more complex the analysis is





2.1 Motivation and Ideas

- Persistent Fault Analysis （PFA）
  - The fault is injected before encryption
  - One fault injection for the multiple exploitations of fault leakages
  - PFA (**CHES 2018**) only uses the last round of fault leakages

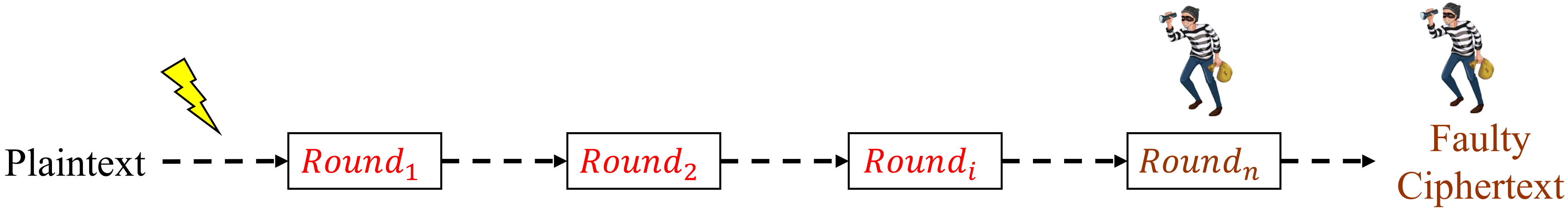


2.1 Motivation and Ideas

The constraint equations of EPFA

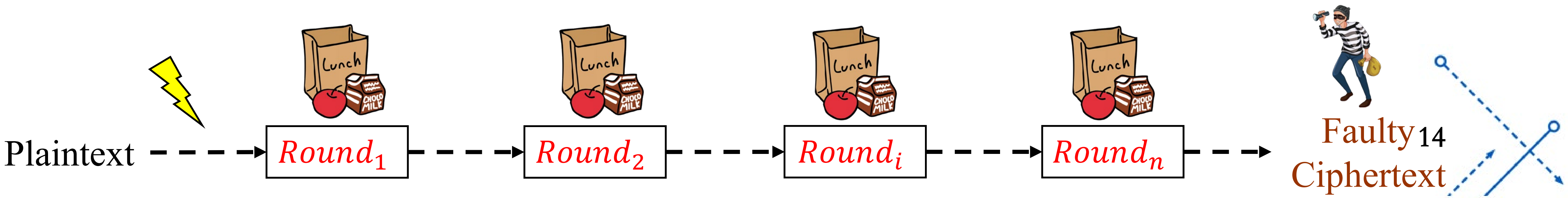
$$
\begin{aligned}
\mathsf{S}'(I_9^{(1)}) =& (\mathsf{S}^{-1}(c_1 \oplus k_1) \oplus (k_1 \oplus \mathsf{S}(k_{14} \oplus k_{10}) \oplus h_{10}) \cdot 14) \\
& \oplus (\mathsf{S}^{-1}(c_{14} \oplus k_{14}) \oplus (k_2 \oplus \mathsf{S}(k_{15} \oplus k_{11})) \cdot 11) \\
& \oplus (\mathsf{S}^{-1}(c_{11} \oplus k_{11}) \oplus (k_3 \oplus \mathsf{S}(k_{16} \oplus k_{12})) \cdot 13) \\
& \oplus (\mathsf{S}^{-1}(c_8 \oplus k_8) \oplus (k_4 \oplus \mathsf{S}(k_{13} \oplus k_9)) \cdot 9), \quad (9)
\end{aligned}
$$

- Enhanced Persistent Fault Analysis （EPFA）
  - The fault model of EPFA is the same as that of PFA
  - EPFA uses the last two rounds of fault leakages
  - The constraints constructed by EPFA depend on the block cipher itself



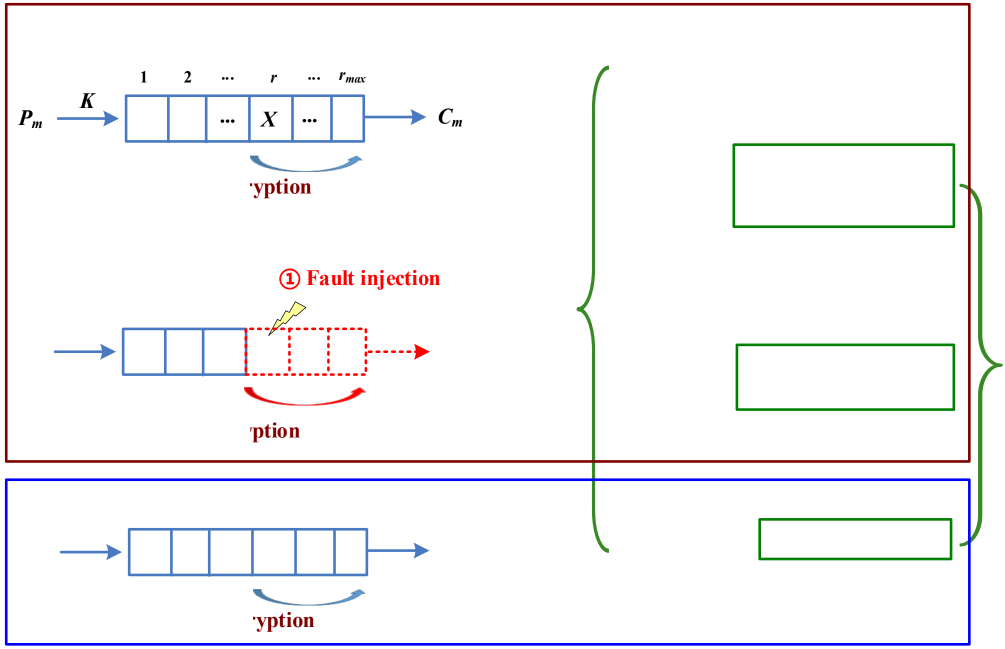
2.1 Motivation and Ideas

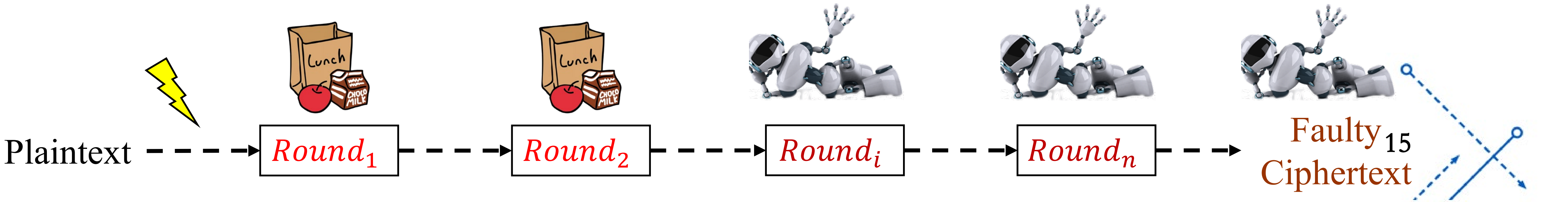
- There **is** a lunch （Free Fault Leakages）
  - Persistent fault leakages exist in **each round**
  - The deeper fault leakages can be used directly without additional fault injection
- There **is** no such thing as a free lunch
  - It is difficult to manually exploit fault leakage from deeper rounds.
  - For EPFA, the deeper the rounds are used, the more constraints are manually constructed, and the complexity increases exponentially
  - **How can we easily taste the free lunch?**



2.1 Motivation and Ideas

- From DFA to AFA
- Algebraic Fault Analysis（AFA）
  - Introducing algebraic analysis to differential fault analysis
  - solving complex fault propagation paths by algebraic solvers
- Combining **algebraic analysis** with PFA can ease the difficulty of exploiting free leakages （√）
  - Algebraic Persistent Fault Analysis（APFA）





OUTLINE

1/ Background ——Persistent Fault Analysis (PFA)

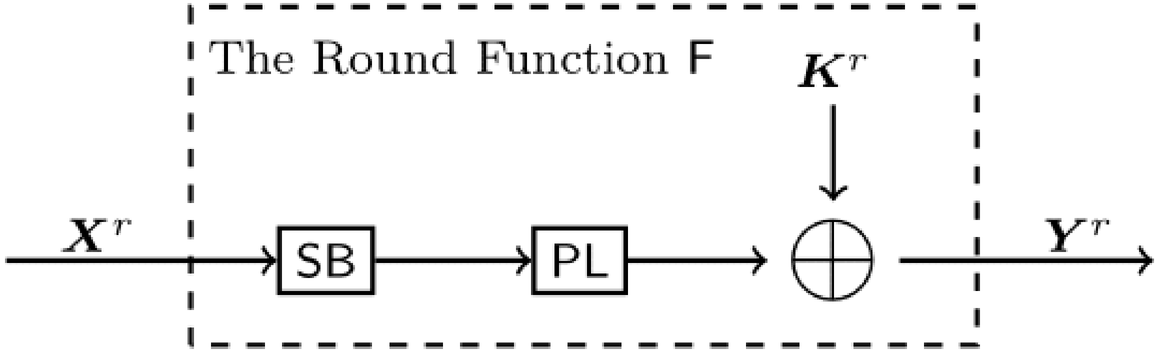
2/ Motivation ——Multiple Rounds of Fault Leakages

3/ Method ——Algebraic Persistent Fault Analysis（APFA）

4/ Application ——Applied to Various Block Ciphers

3.1 SPN Block Cipher

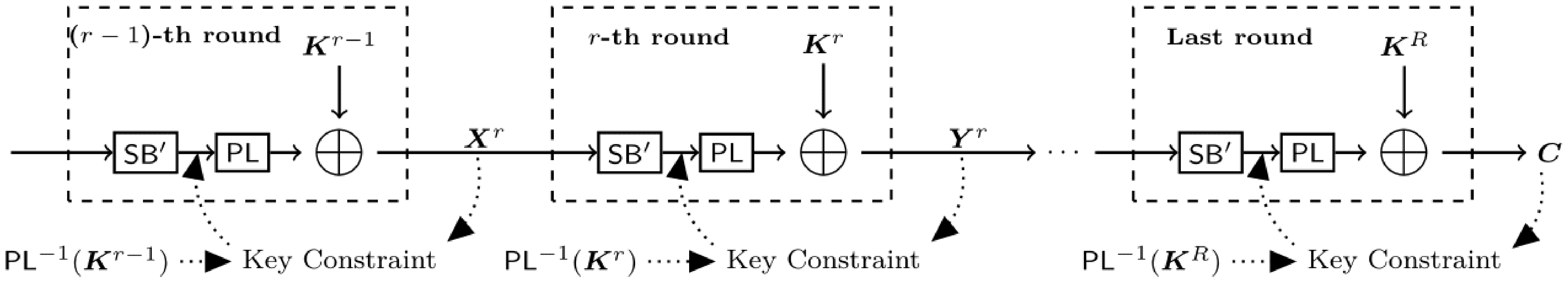
- SPN block ciphers contain three operations of the round function:
  - Substitution layer (SB) , which substitutes the value according to the S-box
  - Permutation layer (PL), which has a special mapping relationship between input and output
  - Addition, the input is XORed with the round key (AK) or constant (AC).



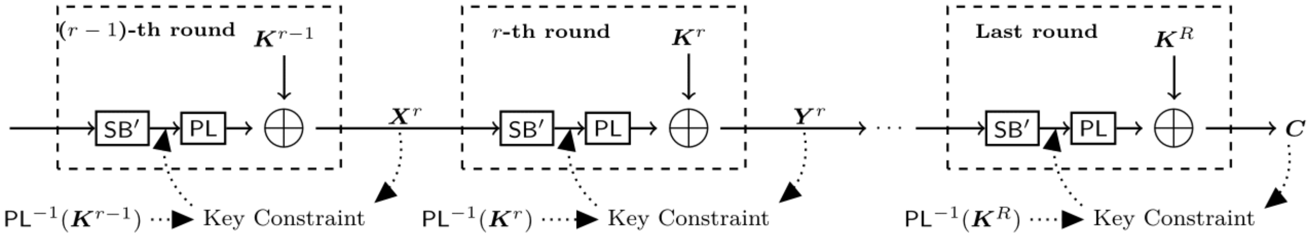
3.2 Core Idea

➤ SPN block cipher round function analysis:

- The form of fault leakages is same for each round
- The fault leakages of each round are only related to the round key $\boldsymbol{K}^r$ and the intermediate variable $\boldsymbol{Y}^r$



3.2 Core Idea



- A general method of fault leakage exploitation can be deduced:
  - The output of the faulty S-box will **not be equal to** the original value V (Line 1)
  - The intermediate variable $\boldsymbol{Y}^r$ after AK contains the fault leakage (Line 2)
  - Performing an inverse permutation operation on $\boldsymbol{Y}^r$ can exploit fault leakage (Line 3)

$$\left.\begin{array}{r}\mathsf{S}'[X_i^r] \neq V \\ \boldsymbol{Y}^r = \mathsf{PL}(\mathsf{SB}'(\boldsymbol{X}^r)) \oplus \boldsymbol{K}^r \\ \mathsf{PL}^{-1}(\boldsymbol{Y}^r) = \mathsf{SB}'(\boldsymbol{X}^r) \oplus \mathsf{PL}^{-1}(\boldsymbol{K}^r)\end{array}\right\} \Longrightarrow \tilde{K}_i^r \neq \tilde{Y}_i^r \oplus V, \quad 0 \leq i < \frac{n}{w}$$

- The above formula only includes constant V and ***r*-th round** variables $\tilde{K}_i^r$ and $\tilde{Y}_i^r$, which are not related with other round
  - The difference between different rounds is only the index of variables in the algebraic system

3.2 The algebraic representation of fault leakages

➢ How to convert $\widetilde{K}_i^r \neq \widetilde{Y}_i^r \oplus V$ to algebraic representation?

- The nature of XORed operation:

$$A \neq B \rightarrow A \oplus B \neq 0$$

- Introducing an intermediate variable D, there is

$$D = \widetilde{K}_i^r \oplus \widetilde{Y}_i^r \oplus V, \quad D \neq 0$$

- d_i is the i-th bit of D, there **must be** an element in d_i that is 1

Remove $\widetilde{Y}_i^r \oplus V$ from the key search space

$$d_i + \tilde{y}_i + \tilde{k}_i + v_i = 0 \ , \ 0 \leq i < w$$

$$d_0 \vee d_1 \vee \cdots \vee d_{w-1} = 1$$

3.3 APFA

Algorithm 1: APFA on block ciphers using multiple rounds of fault leakages.

input : $\mathbb{C}, l, f, N_r$
output : $\boldsymbol{K}$

$V = \mathsf{S}[l]$; // Get the original value of the S.
$\mathsf{S}'[l] = \mathsf{S}[l] \oplus f$;
for $r = 1; r \leq R; r{+}{+}$ **do**
  GenKSR($r, \boldsymbol{K}^r$) ; // Generate the equations for the round key.
end
for $r = 1; r \leq R; r{+}{+}$ **do**
  $\tilde{\boldsymbol{K}}^r$=GenInvPL($\boldsymbol{K}^r$) ; // Generate the equations for $\mathsf{PL}^{-1}(\boldsymbol{K}^r)$.
end
for $\boldsymbol{C} \in \mathbb{C}$ **do**
  for $r = R - N_r; r \leq R; r{+}{+}$ **do**
    GenSB($\boldsymbol{X}^r$);
    GenPL($\boldsymbol{X}^r$);
    GenAK($\boldsymbol{X}^r, \boldsymbol{K}^r$);
    $\tilde{\boldsymbol{X}}^r$=GenInvPL($\boldsymbol{X}^r$);
    GenConst($\tilde{\boldsymbol{X}}^r, \tilde{\boldsymbol{K}}^r, V$) ; // Add constraints to the round key.
  end
  $\boldsymbol{X}^R = \boldsymbol{C}$; // $\boldsymbol{X}^R$ is assigned with $\boldsymbol{C}$.
end
$\boldsymbol{K}$=RunAPFA()

浙江大學 ZHEJIANG UNIVERSITY

OUTLINE

1/ Background ——Persistent Fault Analysis (PFA)

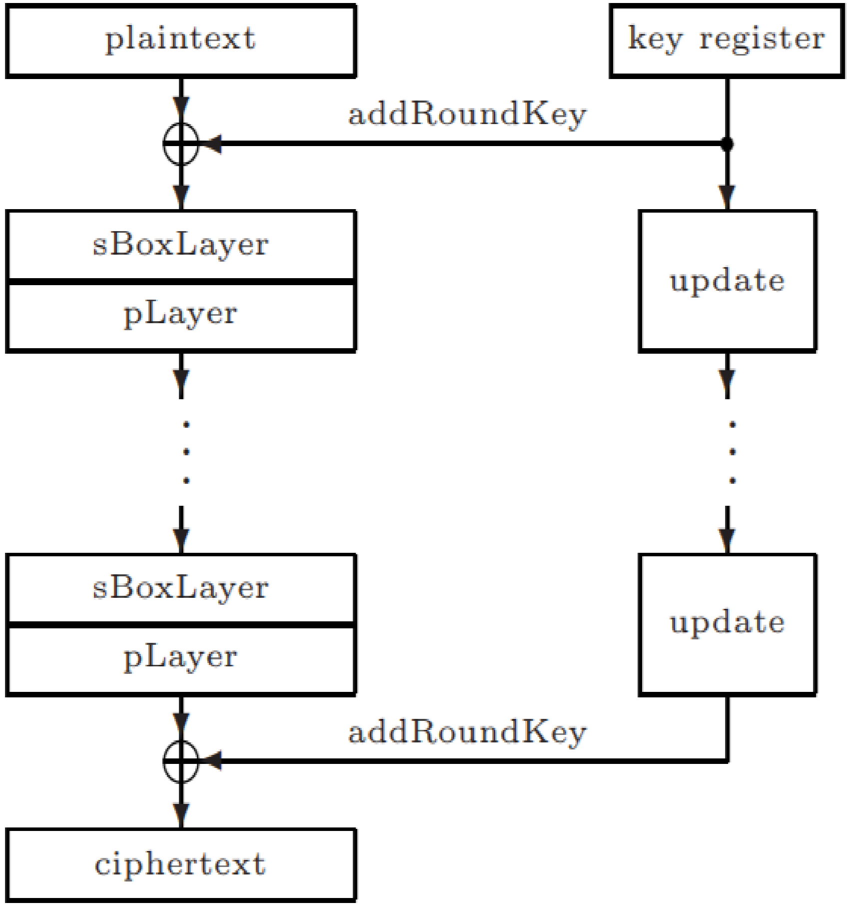
2/ Motivation ——Multiple Rounds of Fault Leakages

3/ Method ——Algebraic Persistent Fault Analysis （APFA）

4/ Application ——Applied to Various Block Ciphers

4.1 Application on PRESENT

- Lightweight SPN block cipher
- 4-bit SBox (16 elements)
- 80/128-bit key size, 64-bit block size



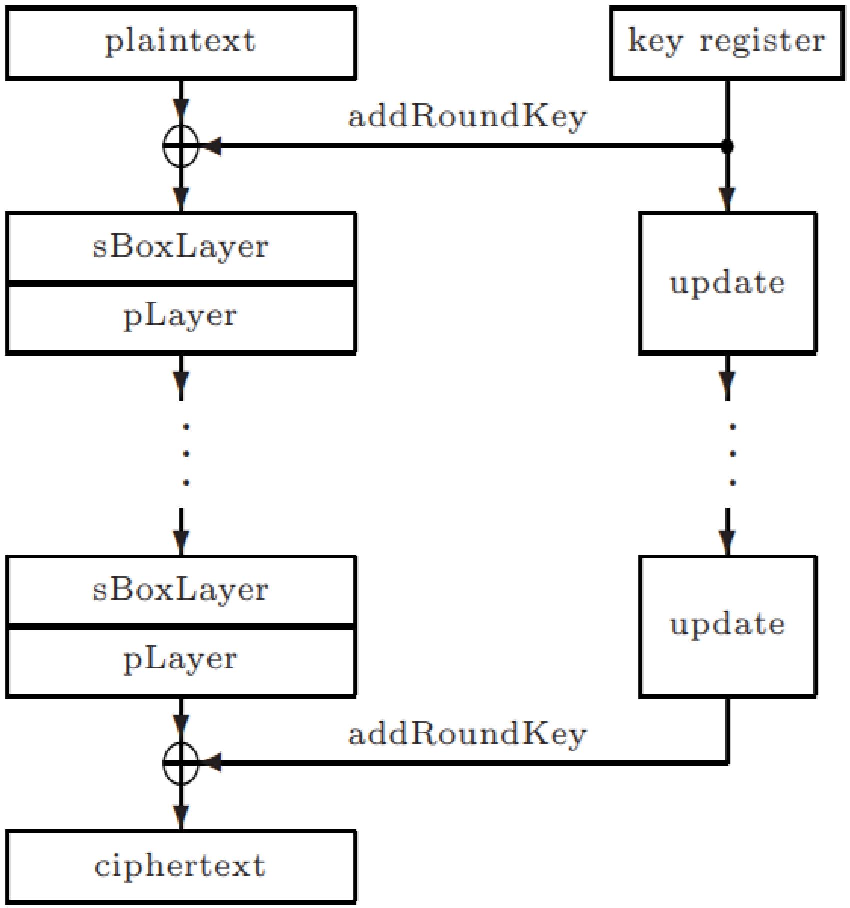
4.1 Application on PRESENT

➢ addRoundKey (AK)

- The input XORed with the round key, denoted as AK, can be representation as:

$$x_i + k_i + y_i = 0, \qquad 0 \leq i < n$$

where x_i, y_i and k_i are one bit of the input, the output and the round key, respectively.

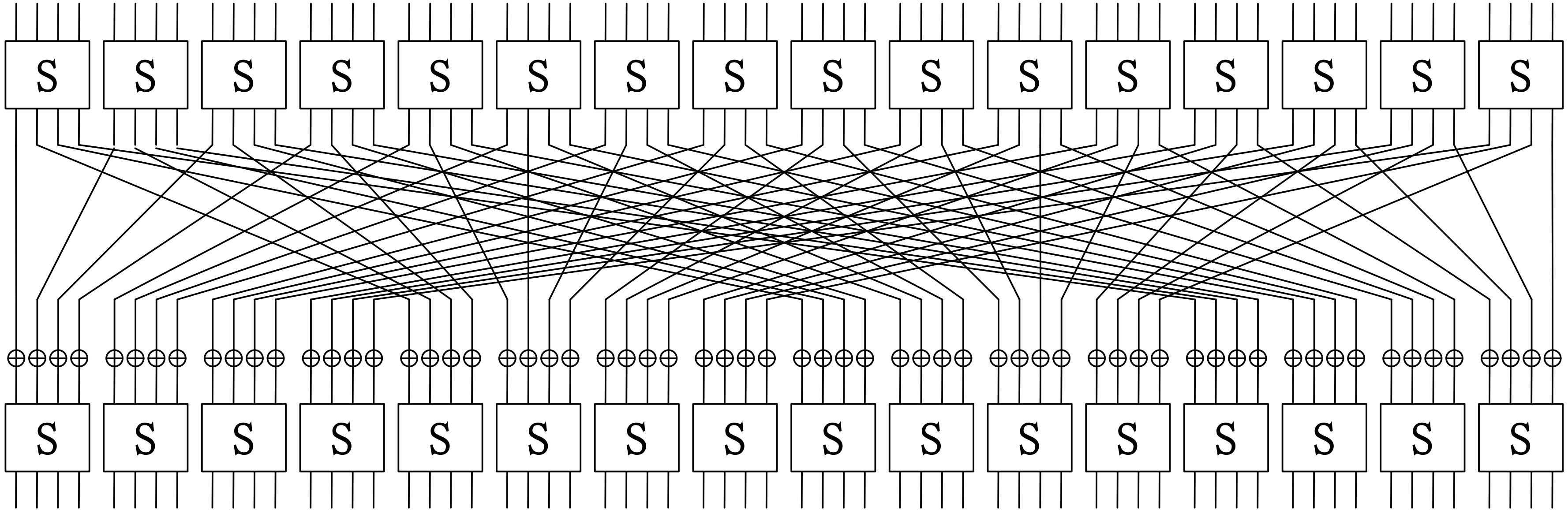


4.1 Application on PRESENT

- **pLayer** (PL)
  - The bit-based permutation, the relationship between the input bit x_i and the output bit y_i can be represented by a permutation table T_P (one bit to one bit):

$$x_i + y_{T_P[i]} = 0, \qquad 0 \leq i < n$$

  - $T_P = [0, 16, 32, 48, 1, 17, \cdots, 15, 31, 47, 63]$



4.1 Application on PRESENT

➢ sBoxLayer (SB)

- The truth table of faulty S-box can be transformed into an Algebraic Normal Form (ANF) first, which can be later re-transformed to CNF and fed into the general SAT solvers.

The truth table of PRESENT							
X3	X2	X1	X0	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1
0	0	1	0	0	1	1	0
0	0	1	1	1	0	1	1
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	0
0	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1
1	0	0	0	0	0	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	1	1
1	1	1	0	0	0	0	1
1	1	1	1	0	0	1	0

⇨

Algebraic Normal Form (ANF)

$$y_0 = x_0 + x_2 + x_3 + x_1x_2$$
$$y_1 = x_1 + x_3 + x_1x_3 + x_2x_3 + x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3$$
$$y_2 = x_0 + x_1 + x_0x_2 + x_1x_2 + x_2x_3 + x_0x_1x_2 + x_1x_2x_3 + x_0x_1x_2x_3$$
$$y_3 = x_2 + x_0x_1 + x_0x_2 + x_0x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3 + x_0x_1x_2x_3$$

⇩

```
252 -272 0
253 -272 0
-250 -251 -252 -253 272 0
x 258 260 261 265 -273 0
x 259 261 266 267 268 269 270 -274 0
x 1 260 261 262 264 266 269 270 -275 0
x 1 258 259 261 265 268 269 270 -276 0
x 277 -254 0
x 278 -255 0
x 279 -256 0
```

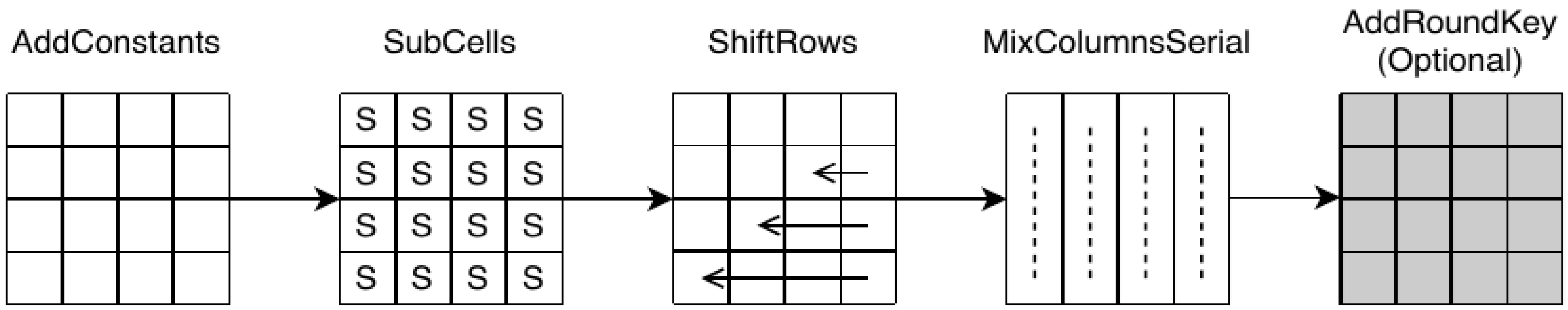
CNF of S-box

4.2 Application on LED

- **AddRoundKey** (AK), which is the same as PRESENST
- **AddConstants** (AC)
  - The input XORed with the constant, denoted as AC, can be representation as:

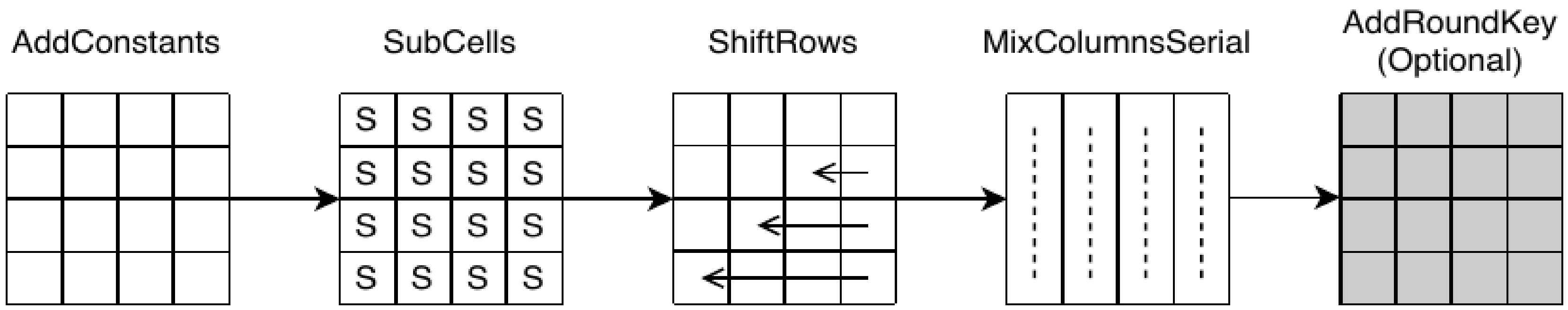
$$x_i + c_i + y_i = 0, \qquad 0 \le i < n$$

where c_i is the bit of the round constant.



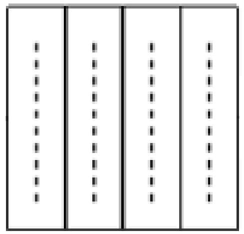
4.2 Application on LED

- **SubCells** (SB), which is the same as PRSENST
- **ShiftRows** (PL), which is the bit-based permutation
  - $$T_P[i] = \left\lfloor \frac{i}{16} \right\rfloor \times 16 + (i - \left\lfloor \frac{i}{16} \right\rfloor \times 16 + \left\lfloor \frac{i}{16} \right\rfloor \times 12) \bmod 16,\ 0 \le i < 64$$



4.2 Application on LED

MixColumnsSerial



➢ **MixColumnsSerial** (PL)

- The MDS matrix multiplications (multiple bits to one bit), most of them are related to multiplication operations on finite fields.

$$\begin{bmatrix} Y_0 & Y_1 & Y_2 & Y_3 \\ Y_4 & Y_5 & Y_6 & Y_7 \\ Y_8 & Y_9 & Y_{10} & Y_{11} \\ Y_{12} & Y_{13} & Y_{14} & Y_{15} \end{bmatrix} = \begin{bmatrix} \mathrm{0x4} & \mathrm{0x1} & \mathrm{0x2} & \mathrm{0x2} \\ \mathrm{0x8} & \mathrm{0x6} & \mathrm{0x5} & \mathrm{0x6} \\ \mathrm{0xb} & \mathrm{0xe} & \mathrm{0xa} & \mathrm{0x9} \\ \mathrm{0x2} & \mathrm{0x2} & \mathrm{0xf} & \mathrm{0xb} \end{bmatrix} \begin{bmatrix} X_0 & X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 & X_7 \\ X_8 & X_9 & X_{10} & X_{11} \\ X_{12} & X_{13} & X_{14} & X_{15} \end{bmatrix}$$

Table 3: LED's Multiplication over GF(2^4).

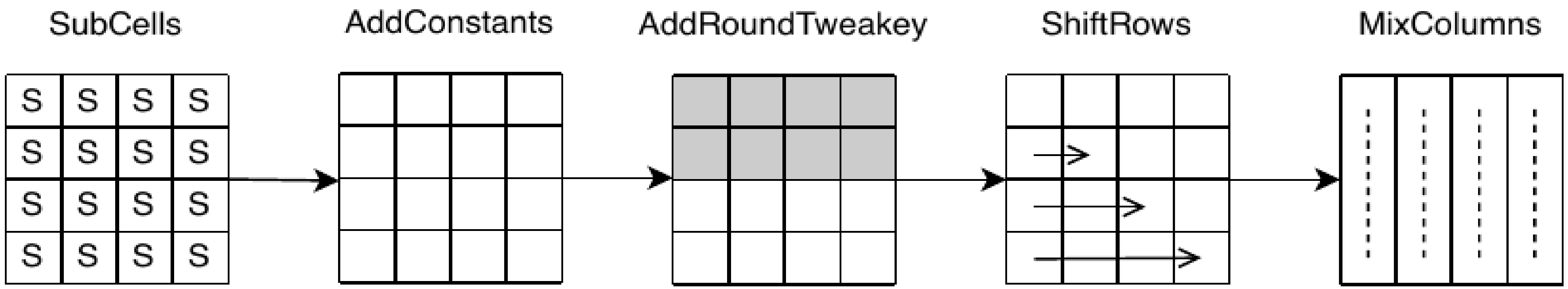
	y_0	y_1	y_2	y_3		y_0	y_1	y_2	y_3
0x2	x_3	x_0+x_3	x_1	x_2	0x9	x_0+x_1	x_2	x_3	x_0
0x3	x_0+x_3	$x_0+x_1+x_3$	x_1+x_2	x_2+x_3	0xa	x_1+x_3	$x_0+x_1+x_2+x_3$	$x_1+x_2+x_3$	$x_0+x_2+x_3$
0x4	x_2	x_2+x_3	x_0+x_3	x_1	0xb	$x_0+x_1+x_3$	$x_0+x_2+x_3$	x_1+x_3	x_0+x_2
0x5	x_0+x_2	$x_1+x_2+x_3$	$x_0+x_2+x_3$	x_1+x_3	0xc	x_1+x_2	x_1+x_3	x_0+x_2	$x_0+x_1+x_3$
0x6	x_2+x_3	x_0+x_2	$x_0+x_1+x_3$	x_1+x_2	0xd	$x_0+x_1+x_2$	x_3	x_0	x_0+x_1
0x7	$x_0+x_2+x_3$	$x_0+x_1+x_2$	$x_0+x_1+x_2+x_3$	$x_1+x_2+x_3$	0xe	$x_1+x_2+x_3$	x_0+x_1	$x_0+x_1+x_2$	$x_0+x_1+x_2+x_3$
0x8	x_1	x_1+x_2	x_2+x_3	x_0+x_3	0xf	$x_0+x_1+x_2+x_3$	x_0	x_0+x_1	$x_0+x_1+x_2$

4.3 Application on SKINNY

- **AddConstants** (AC) and **SubCells** (SB) are the same as LED and PRESENT
- **AddRoundTweakey** (AK), is slightly different, only uses half of the round key for each round

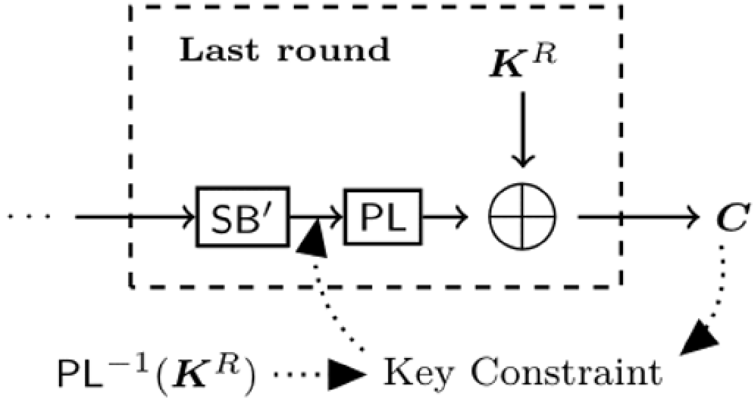
$$\begin{aligned} x_i + k_i + y_i = 0, &\quad 0 \le i < 32 \\ x_i + y_i = 0, &\quad 32 \le i < 64 \end{aligned} \qquad M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

- **ShiftRows** (PL) is similar to LED, but **MixColumns** (PL) only used a binary matrix

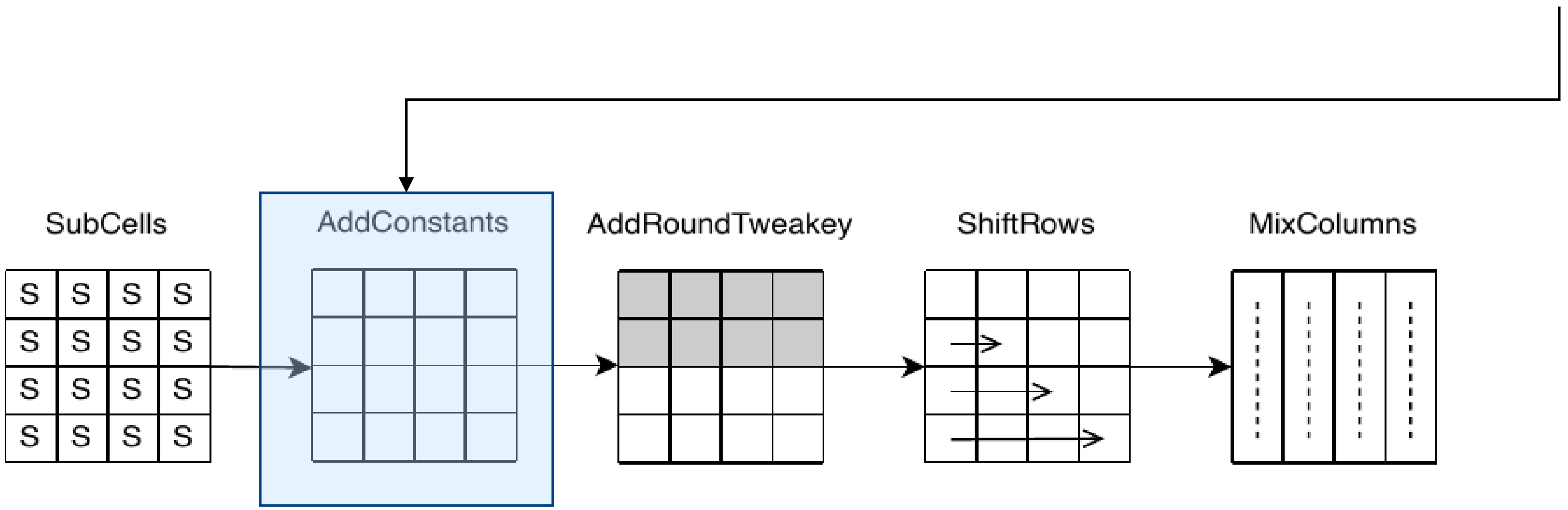


4.4 Application on SKINNY

- There is only **AddConstans** (AC) between **SubCells** (SB) and **AddRoundTweakey** (AK) in SKINNY



- It means that the inverse permutation operation $\mathbf{PL}^{-1}$ is not required when build the equation for fault leakages



4.4 Application on Feistel Block Chiper LBlock

- The round function of LBlock ($2 \leq i \leq 33$):

$$\boldsymbol{X}^i = \mathsf{F}(\boldsymbol{X}^{i-1}, \boldsymbol{K}^{i-1}) \oplus (\boldsymbol{X}^{i-1} <<< 8)$$
$$\mathsf{F} = \mathsf{PL}(\mathsf{SB}(\mathsf{AK}(\boldsymbol{X}^i, \boldsymbol{K}^i)))$$

- Due to the design of Feistel structure, the fault leakage of **F** is masked by the previous intermediate state $\boldsymbol{X}^{i-1}$.

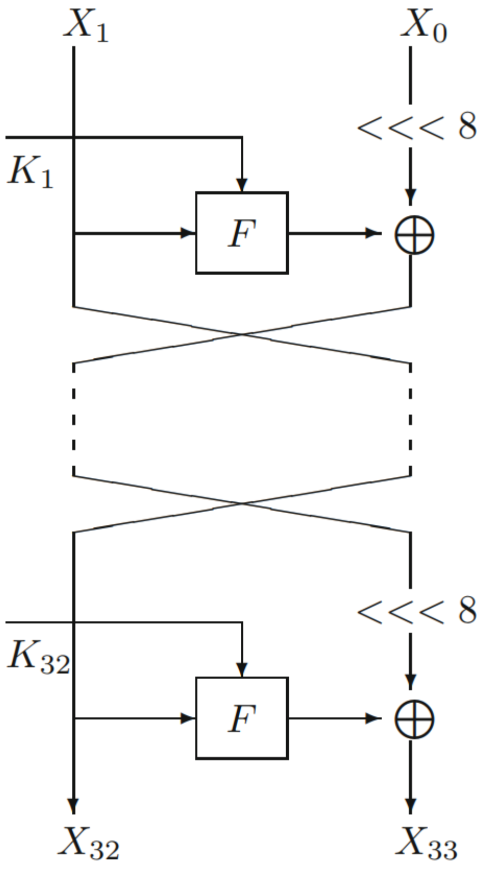
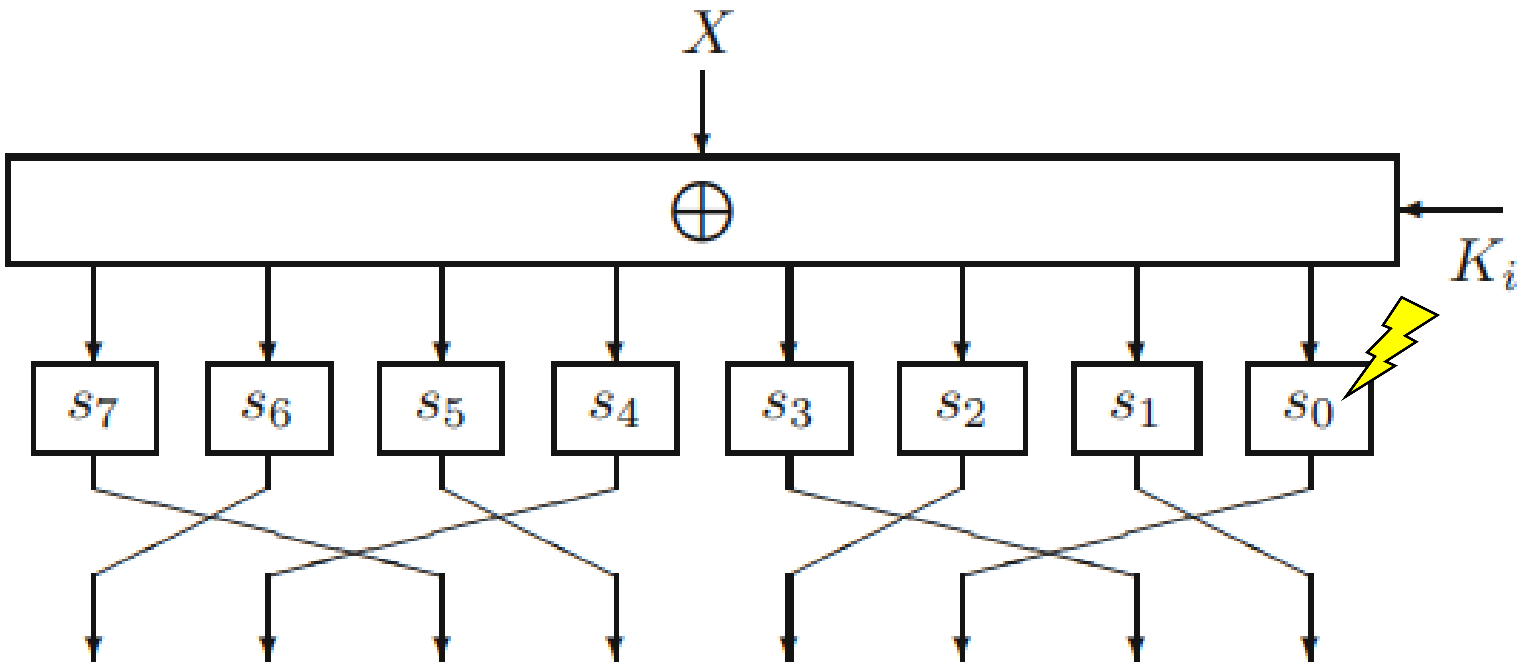


Fig. 1. Encryption procedure of LBlock

4.4 Application on Feistel Block Chiper LBlock

- There are **8 different parallel S-boxes** in SB of LBlock.
- Assume a single fault has been injected into one S-box S_i, and the fault location is l.



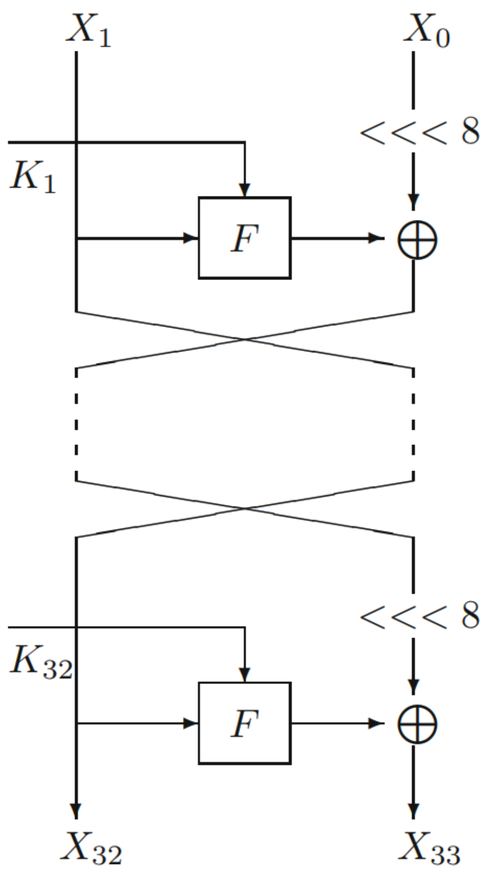


Fig. 1. Encryption procedure of LBlock

4.4 Application on Feistel Block Chiper LBlock

- The adversary needs to encrypt the same plaintext for twice, one for the **normal encryption** and the other for the **encryption with faulty S-box**.
- We can exploit those ineffective ciphertexts do not visit $S_i[l]$
- Fault leakage becomes $K_i \neq X_i \oplus l$
- We use a total of **112 ciphertexts** to recover the master key

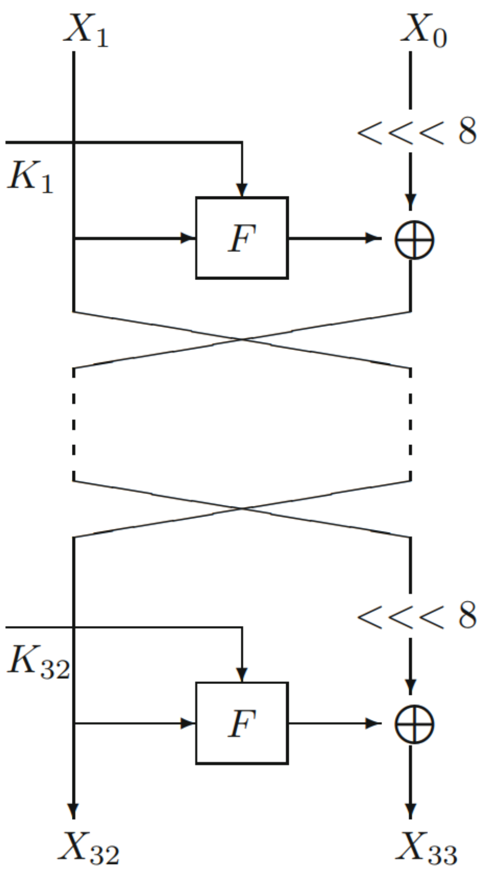


Fig. 1. Encryption procedure of LBlock

4.5 Experiment

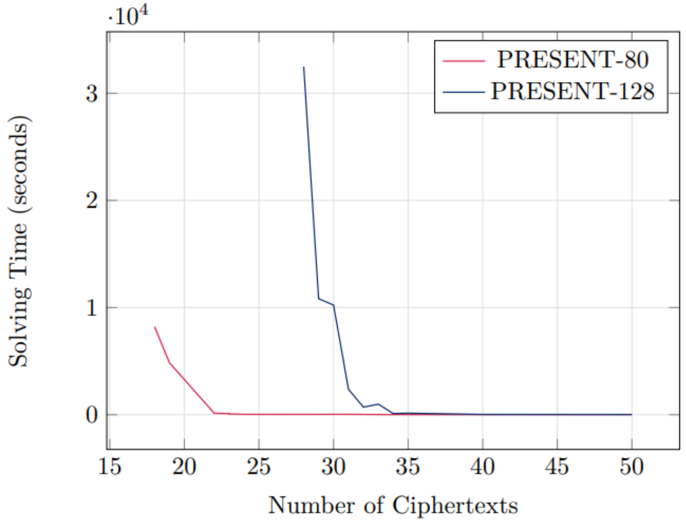


Figure 3: APFA on PRESENT-80 and PRESENT-128.

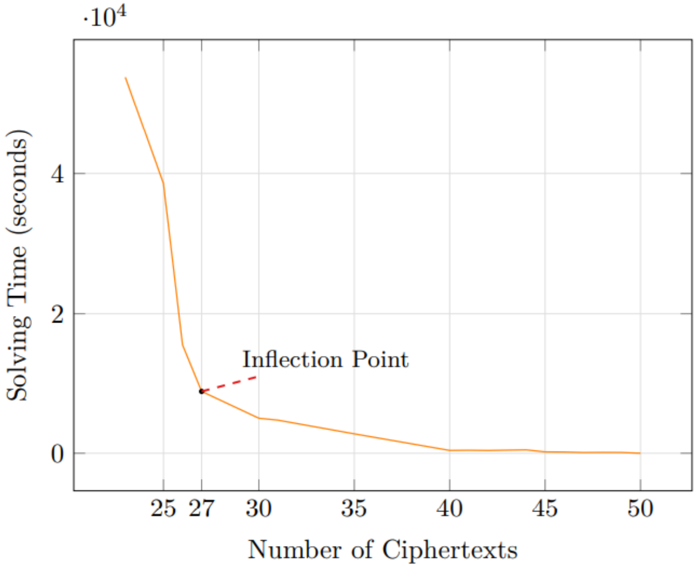


Figure 5: APFA on LED-64.

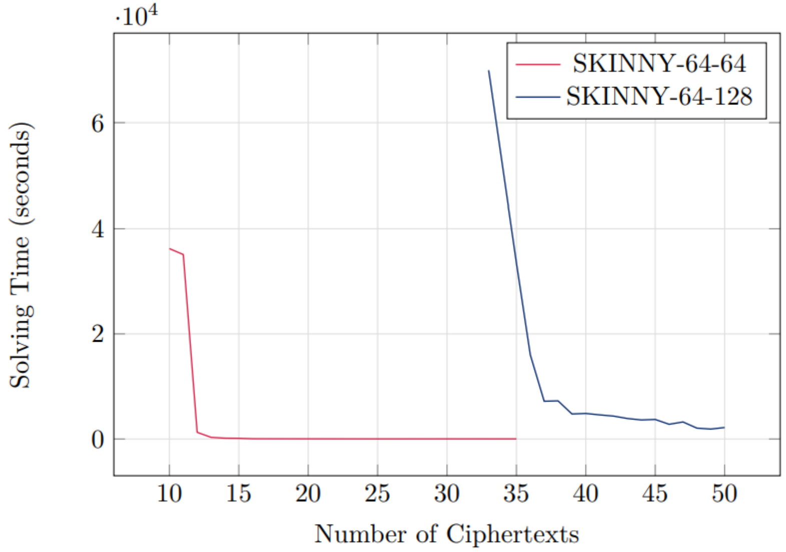


Figure 6: APFA with SKINNY-64-64 and SKINNY-64-128.

Since APFA can use multiple rounds fault leakages, it is not sensitive to the key length, and the number of ciphertexts required by PRESENT-80 and 128 are similar.

4.5 Experiment

From the slope of SKINNY-64-64, it can be found that the deeper rounds, the less fault leakages can be exploited.

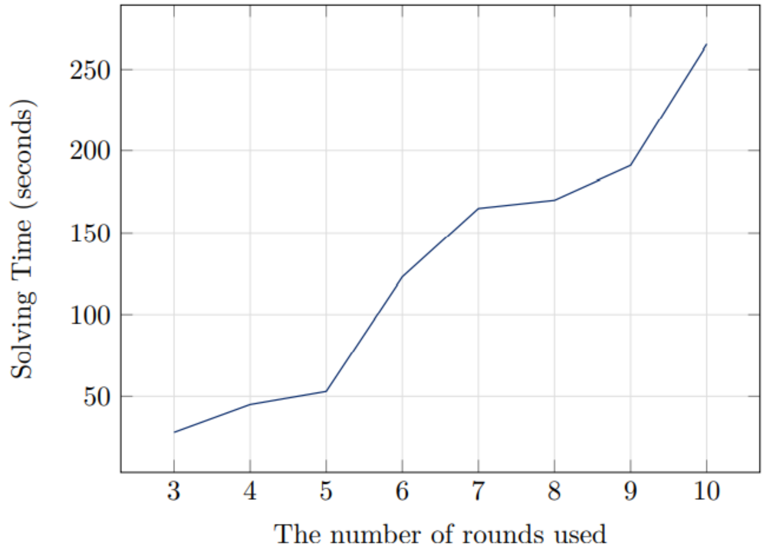


Figure 4: APFA on PRESENT-128 with 40 faulty ciphertexts.

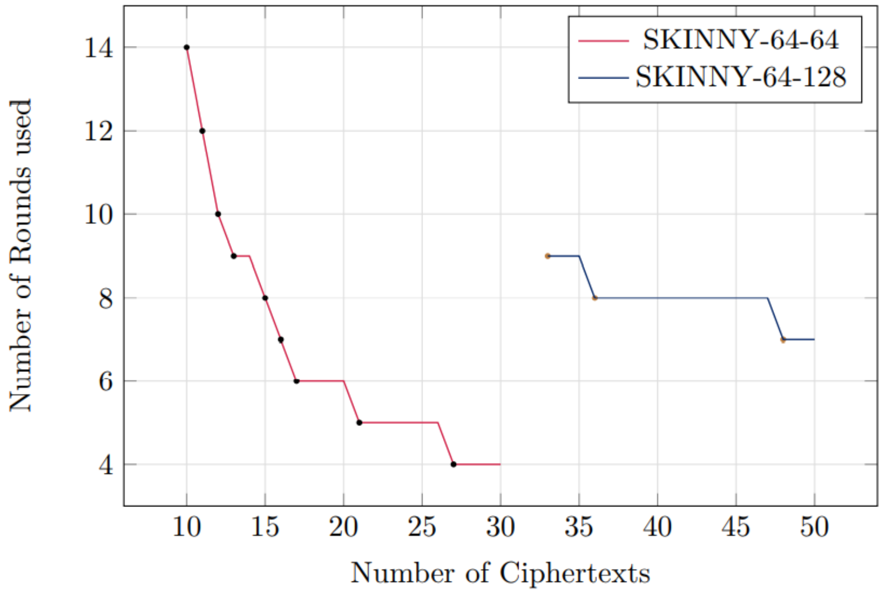


Figure 7: Relationship between N_c and N_r for SKINNY-64-64 and SKINNY-64-128.

When fault leakages are sufficient to solve the master key, increasing the number of analysis rounds does not reduce the solving time (the extra time is used for the intermediate variables).

4.6 Conclusion

➢ **For the first time**, We combine algebraic fault analysis with persistent fault analysis

- It uses fewer ciphertexts as well as deeper rounds of fault information to recover the key.
- It can apply to various SPN-based block ciphers by algebraic versatility.
- It is extended to Feistel-based light weight block ciphers (e.g., LBlock).

Table 1: Comparison of different PFA methods on different ciphers.

Type	RowID	Design	Cipher	Analysis Method					Reduced number of ciphertexts in times
				PFA-18 [ZZJ+20]	PFA-20 [ZZJ+20]	EPFA [XZY+20]	This paper	Section	
Lightweight Block Ciphers	1	SPN	PRESENT-80	-	101	-	18	Sec. 7.3	5.61×
	2		PRESENT-128	-	-	-	28		-
	3		LED-64	-	-	75	23	Sec. 7.4	3.26×
	4		SKINNY-64-64	-	-	1550	10	Sec. 7.5	155.00×
	5		SKINNY-64-128	-	-	-	33		-
	6	Feistel	LBlock-80	-	-	-	112	Sec. 8.1	-
Classic Block Ciphers	7	SPN	AES-128	2281	1641	1000	1300	Sec. 8.2	-1.30×

Q & A

THANKES!