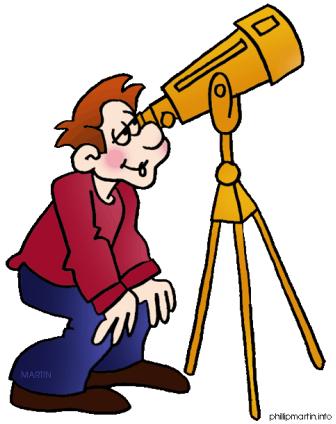


TELE

STETHO

MICRO

SCOPE

Published in SAC 2015

DIAC 2015

Singapore

SCOPE : On the Side Channel Vulnerability of Releasing Unverified Plaintexts

Dhiman Saha, Dipanwita Roy Chowdhury

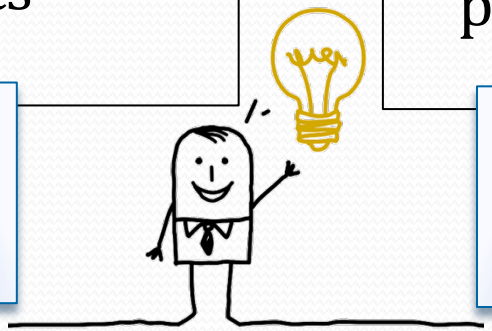
Crypto Research Lab,
Department of Computer Science and Engineering, IIT Kharagpur, India
{dhimans,drc}@cse.iitkgp.ernet.in

DIAC 2015

1

- Revisiting the concept of releasing unverified plaintexts (RUP) in *authenticated decryption* in the context of fault attacks
- The concept of faulty unverified plaintexts

Tries to exploit the RUP property of AE schemes



2

- Study CAESAR submission APE (PRIMATEs)
- Mount DFA using the power of observing faulty unverified plaintexts in RUP

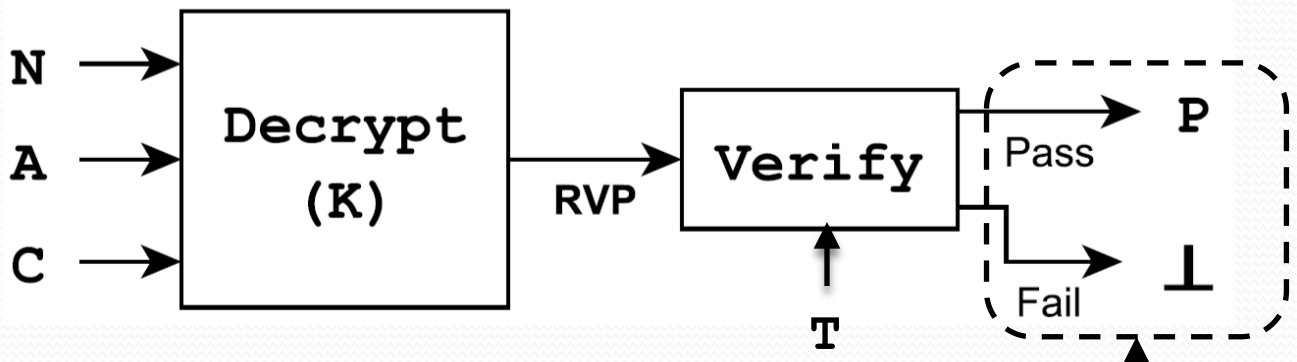
Exploits mode of operation & structural properties

Authenticated Decryption

- The classical approach

"Successful Verification :

A pre-requisite for releasing decrypted ciphertexts"



- We refer to this as the RVP model
  - **RVP : Releasing Verified Plaintexts**

RUP

How to Securely Release Unverified Plaintext in Authenticated Encryption*

Elena Andreeva[1,2], Andrey Bogdanov[3], Atul Luykx[1,2], Bart Mennink[1,2], Nicky Mouha[1,2], and Kan Yasuda[1,4]

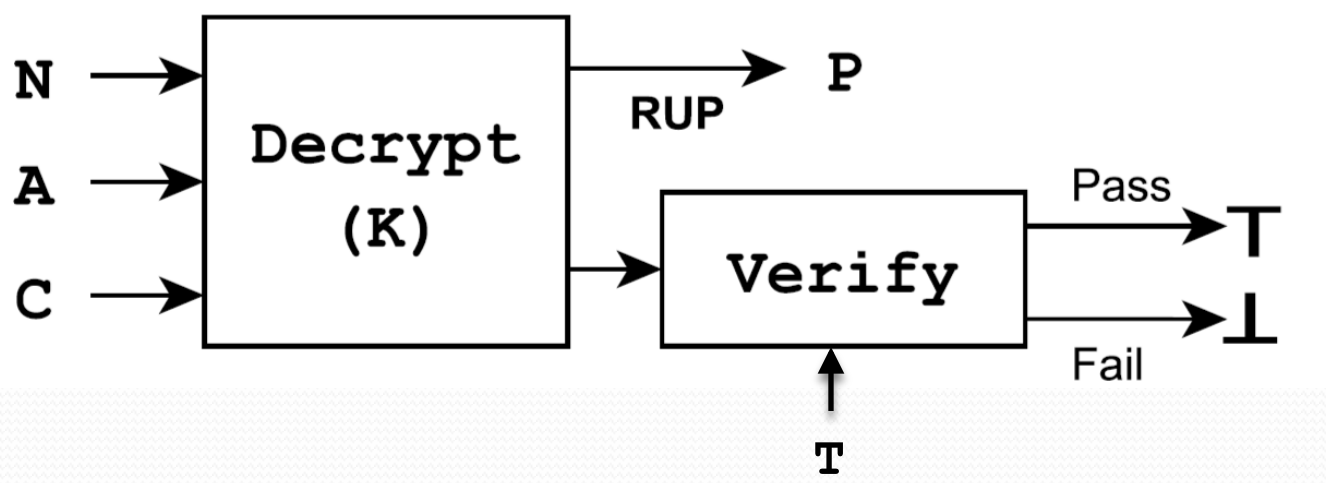
[1] Department of Electrical Engineering, ESAT/COSIC, KU Leuven, Belgium
{firstname.lastname}@esat.kuleuven.be
[2] iMinds, Belgium
[3] Department of Mathematics, Technical University of Denmark, Denmark
anbog@dtu.dk
[4] NTT Secure Platform Laboratories, Japan
yasuda.kan@lab.ntt.co.jp

Releasing Unverified Plaintexts

- The new approach

"Pre-requisite Dropped:

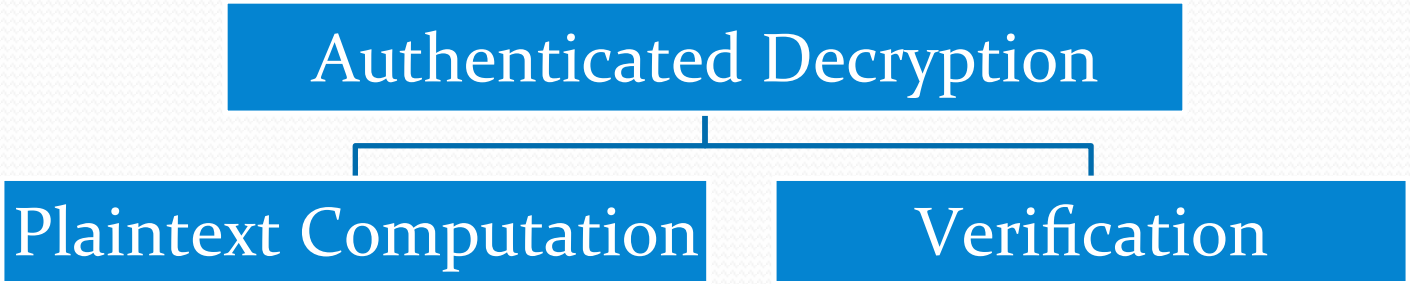
Allows release of decrypted ciphertexts before verification"



- The authors refer to this as the RUP model
  - **RUP : Releasing Unverified Plaintexts**

The Concept of RUP

- Primary idea –



- Supporting notions of
  - INT-RUP(Integrity) & PA (Privacy)

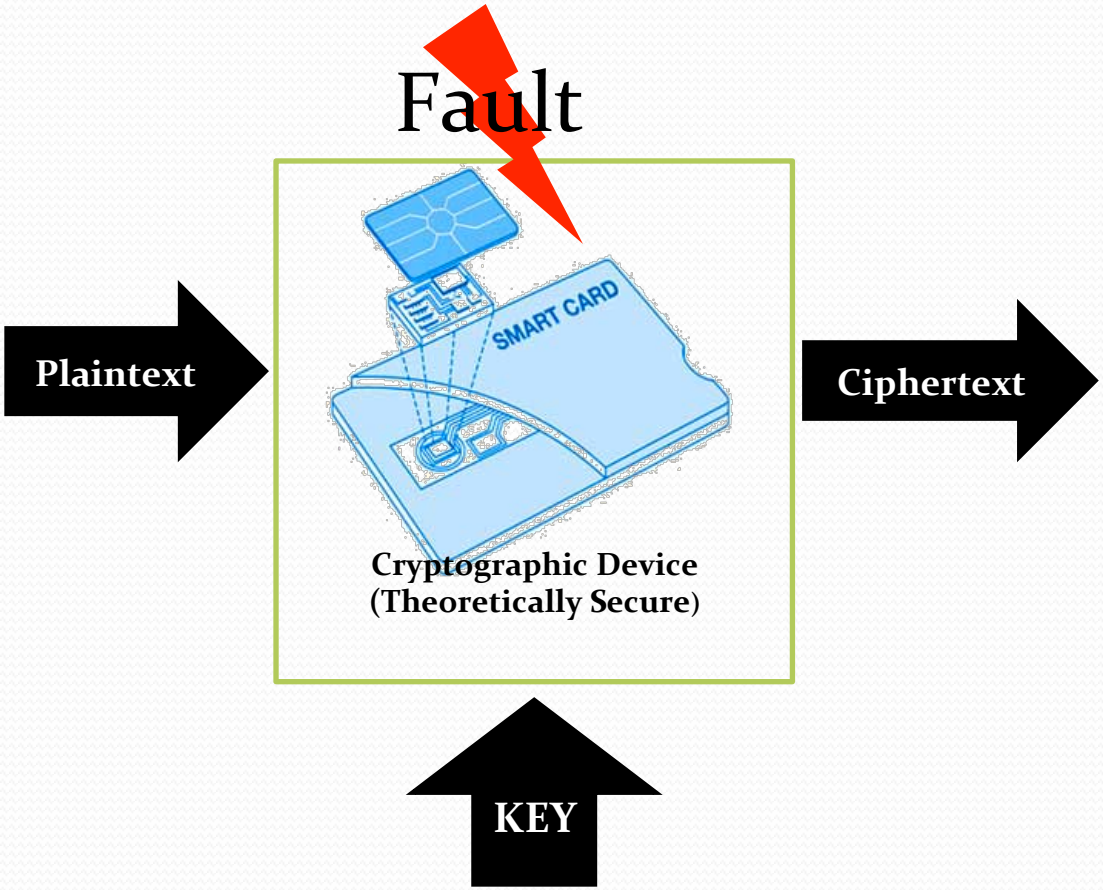
- Motivated by practical concerns faced in RVP:
  - Insufficient memory
  - Real-time usage
  - Efficiency issues

The Big Question?

Can the ability to
"**Observe Unverified Plaintexts**"
serve as a source of
Side-channel information?


DFA

Fault Analysis



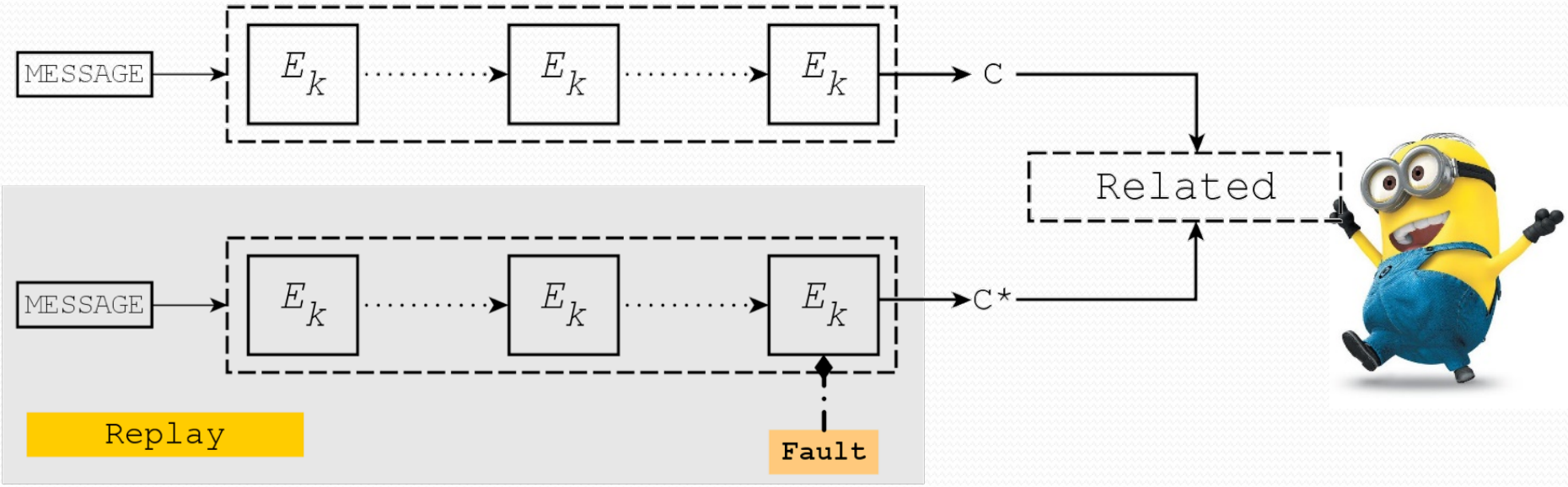
- A side-channel attack
- Attack the implementation
- **Basic Idea:**
  - Inject fault in a cryptosystem
  - Exploit faulty output

Differential Fault Analysis

- One of the most popular side-channel attacks
- DFA assumption –

The Replaying Criterion

Ability to induce faults in the intermediate state of the cipher while **replaying** the encryption with the same plaintext.

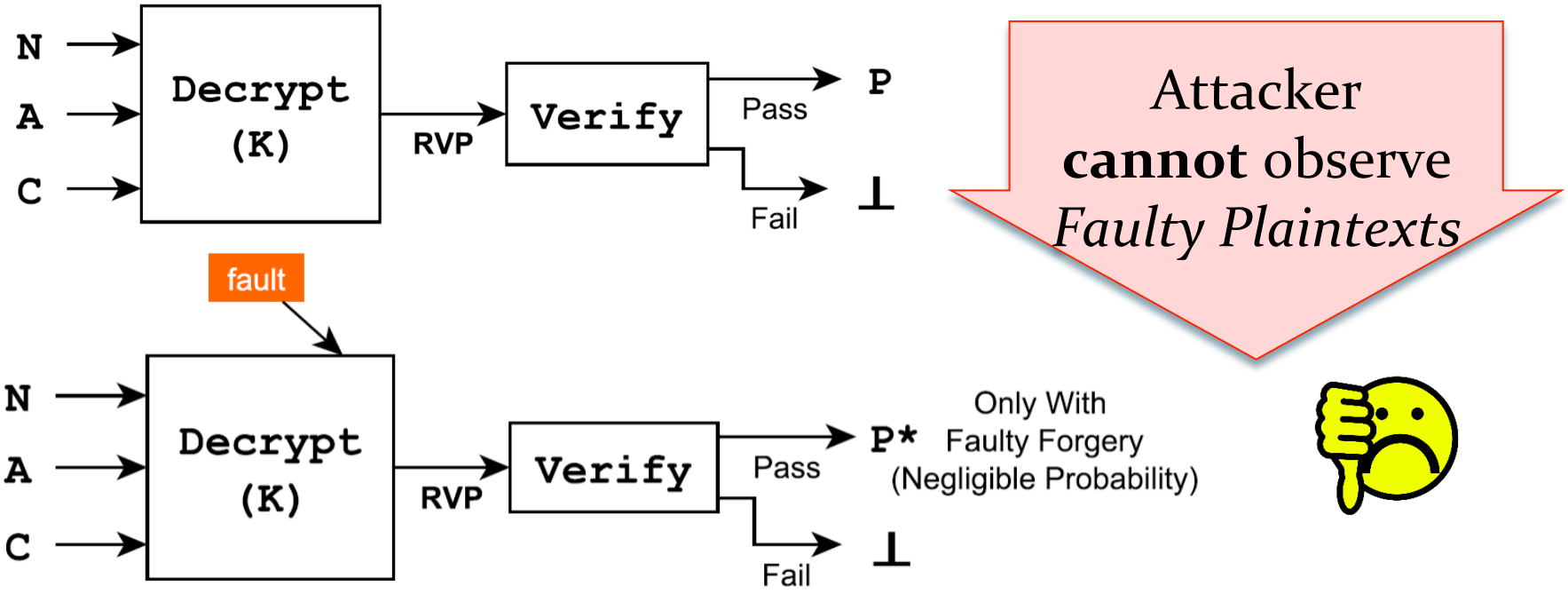


DFA + RVP?

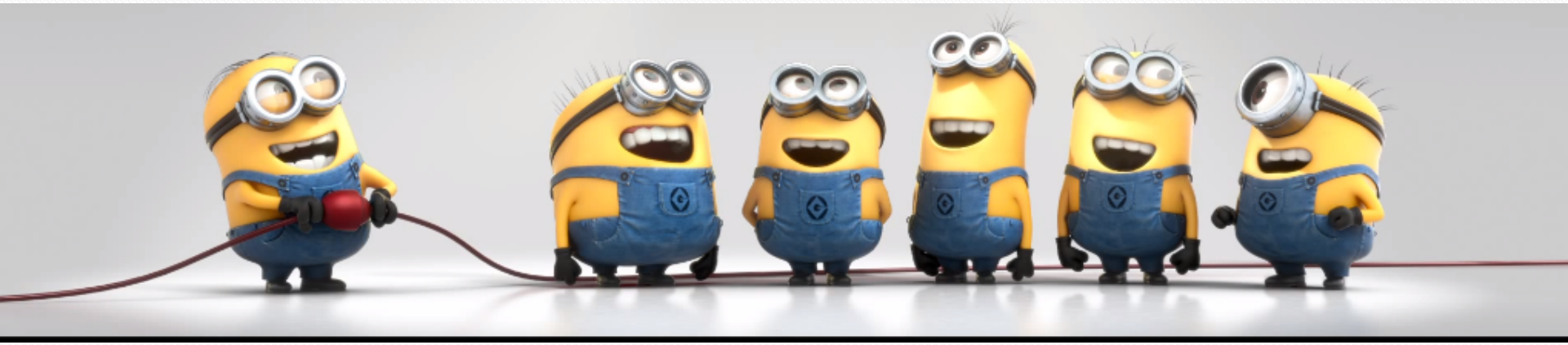
- Recall :
  - *"If verification fails, algorithm outputs NULL under RVP model"*

- **Highly unlikely that**

  verification will pass in the presence of a fault

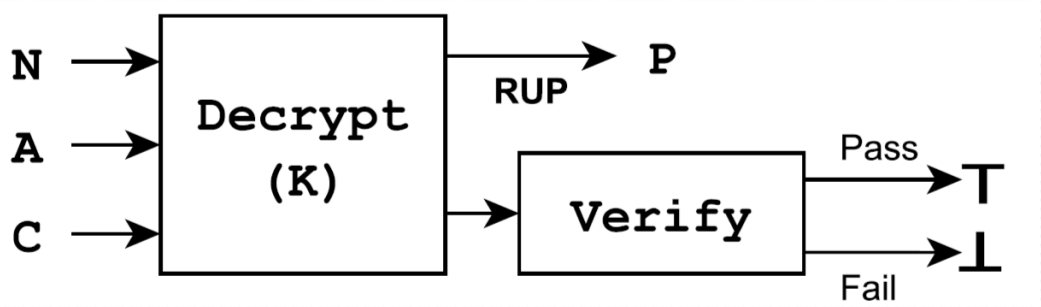


RUP in the light of DFA

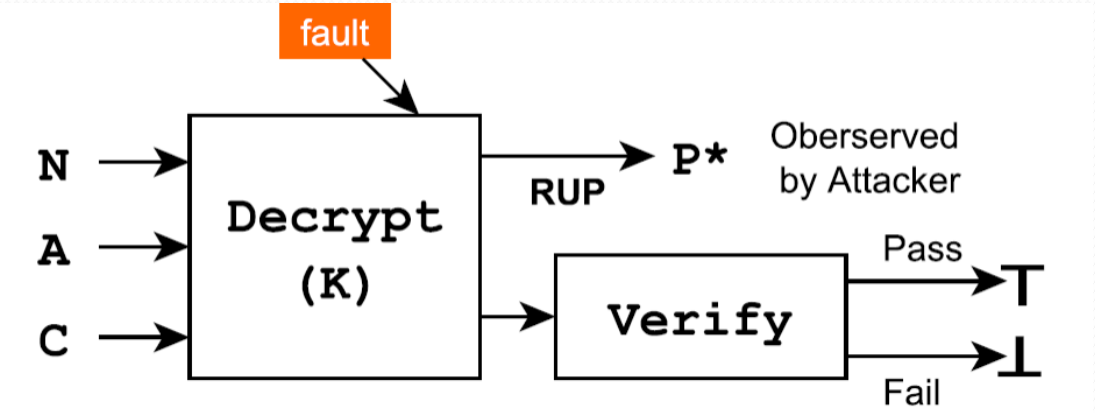


DFA + RUP?

- RUP allows observing plaintexts *even if verification fails*



Attacker can observe ***Faulty*** *Unverified Plaintexts*



Nonce Bypass

- Recall the ***replaying criterion*** of DFA 
- Nonce constraint contradicts replaying criterion of DFA
  - Implies implicit protection
  - Necessitates other handling techniques while attacking encryption
- Attacking decryption gets rid of this problem
  - Nonce constraint is relaxed
  - By definition *decryption oracle can be queried multiple times with same nonce*

The Verdict

- With RUP attacker can bypass nonce and
  a) Induce random faults during decryption and subsequently
  b) Observe

  “faulty unverified plaintext blocks”

Can this be translated to an actual fault attack on an authenticated cipher?

APE

PRIMATEs (CAESAR)

APE

misuse resistance

HANUMAN

security with ideal permutation

GIBBON

trade-off speed/security

http://primates.ae/wp-content/uploads/DIAC2014_Primates.pdf

Authenticated Permutation-based Encryption (APE)

APE

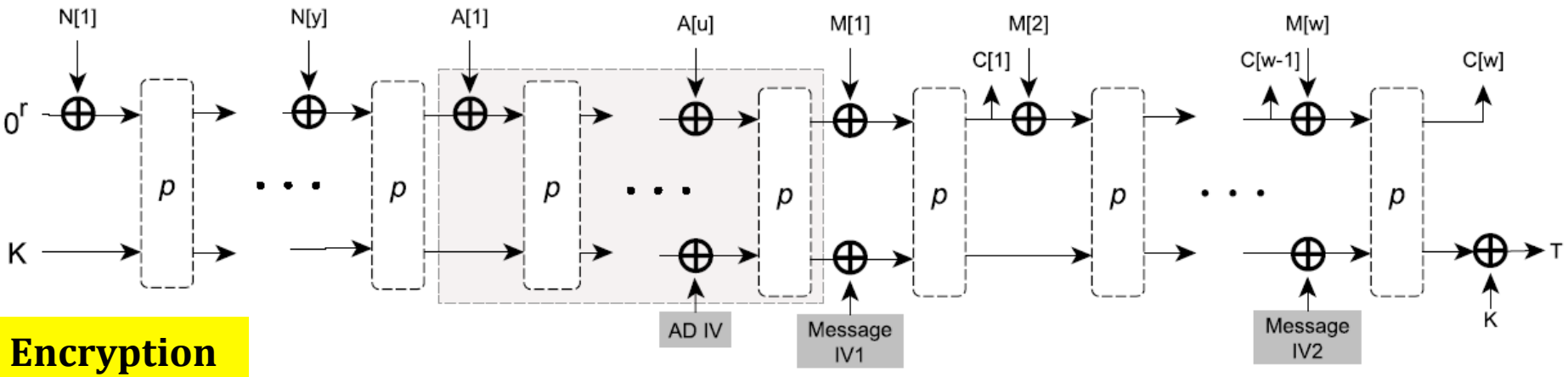
misuse resistance

- Introduced first in FSE 2014
  - Reintroduced in CAESAR with indigenous permutation PRIMATE
- First permutation based AE scheme
- Inspired from SPONGE
- Has been selected for Round 2 of the on-going CAESAR competition

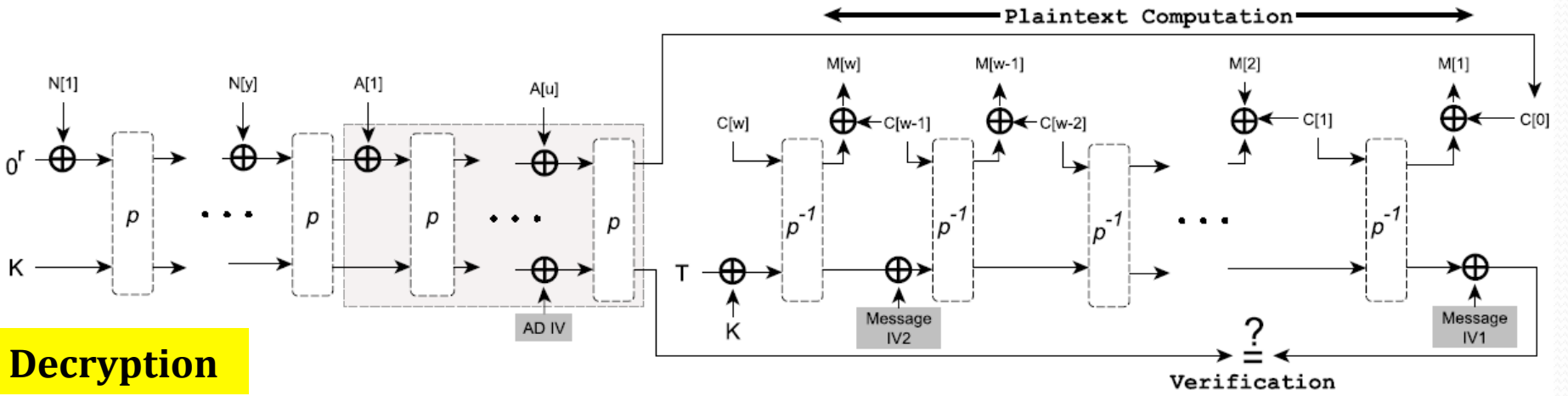
Why do we choose APE?

APE supports RUP

APE Mode of Operation



Encryption



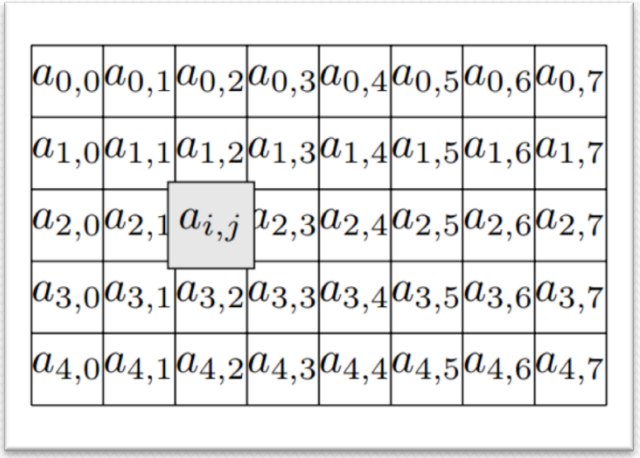
Decryption

The PRIMATE Permutation

- **Structurally follows AES round function**

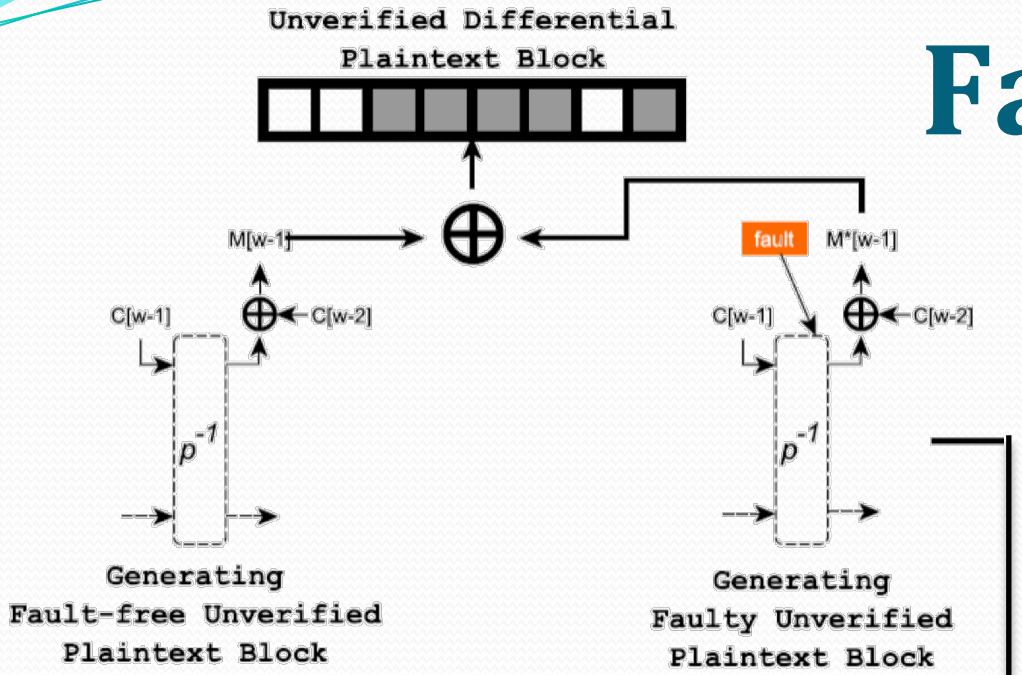
- Has two variants : PRIMATE-80/120
- State size - (5 x 8) / (7 x 8)
- Operates on five-bit elements
- Key-Size is 160 bit for APE-80

- This work targets APE decryption
  - Uses Inverse PRIMATE permutation
  - Applies inverse of component transformations in reverse order



Analysing APE Decryption

Fault Injection

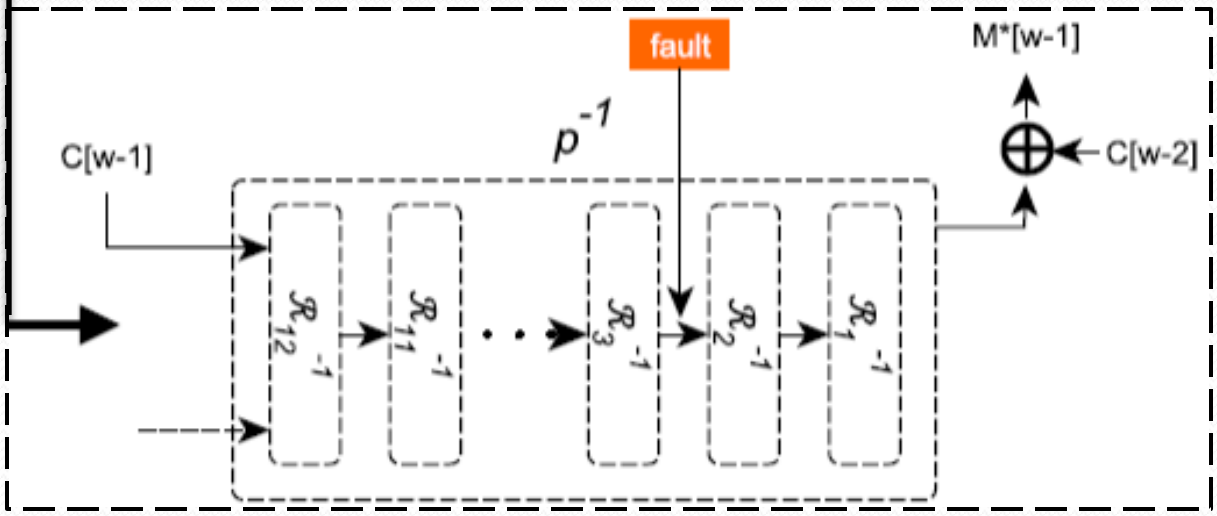


LOCATION

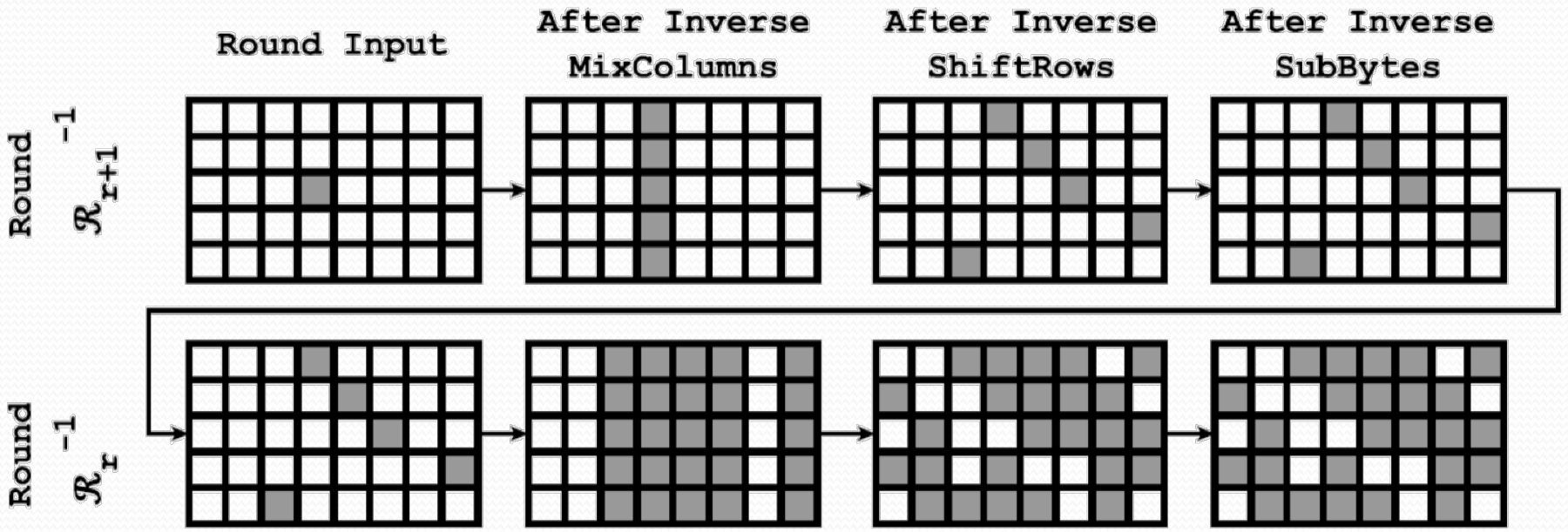
Input of penultimate round of the inverse PRIMATE permutation

FAULT MODEL

Random Word Fault Model



Fault Diffusion



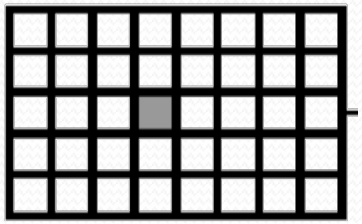
Every row in the last state matrix has three zero-words

Is there a relation between the positions of the zeros-words and the source fault

The Bijection Lemma

Lemma 1. *If fault is induced in the* j^{th} *column of the state at the input of* $\mathcal{R}_{r+1}^{-1}$, *then the fault-free words in the differential plaintext block released after* $\mathcal{R}_{r}^{-1}$ *are* $((j+3),(j+5),(j+6)) \bmod 8$.

Column Position of Fault



One-to-one relation

Zero-Word Pattern

This is observed by the attacker

- Surfaces due to non-square state matrix
- Knowledge of column position make the attack developed later 8 times faster
- ***Note that row position cannot be distinguished***

Bijection Lemma In-Action

$p \oplus p'_i$ (Differential plaintext block)								Traced Back Column (Bijection Lemma)
5	6	22	0	5	0	0	30	0
0	25	18	27	14	0	20	0	2
22	19	0	16	0	0	9	2	7
6	0	0	2	3	10	1	0	4
0	31	0	0	9	13	22	4	5
28	18	17	1	0	14	0	0	1
22	0	15	0	0	21	20	2	6
23	27	15	8	0	2	0	0	1
12	0	0	18	6	29	31	0	4
0	17	13	21	13	0	21	0	2
20	10	11	0	17	0	0	18	0
0	29	0	0	7	20	21	6	5

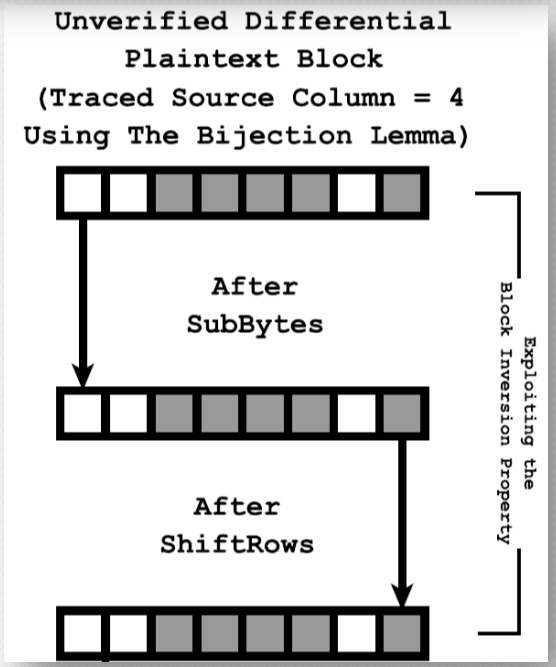
Table 5: Extracting information from unverified fault-free and faulty plaintext blocks

The Block Inversion Property

Property 1 *If the state after* μ_1^{-1} *in* $\mathcal{R}_1^{-1}$ *(the last round of* p^{-1}*) be represented as* $t = [t_{i,j}]$ *and the released plaintext block and next ciphertext block be* p *and* c *respectively, then* $t_{0,*}$ *is public by the following expression:*

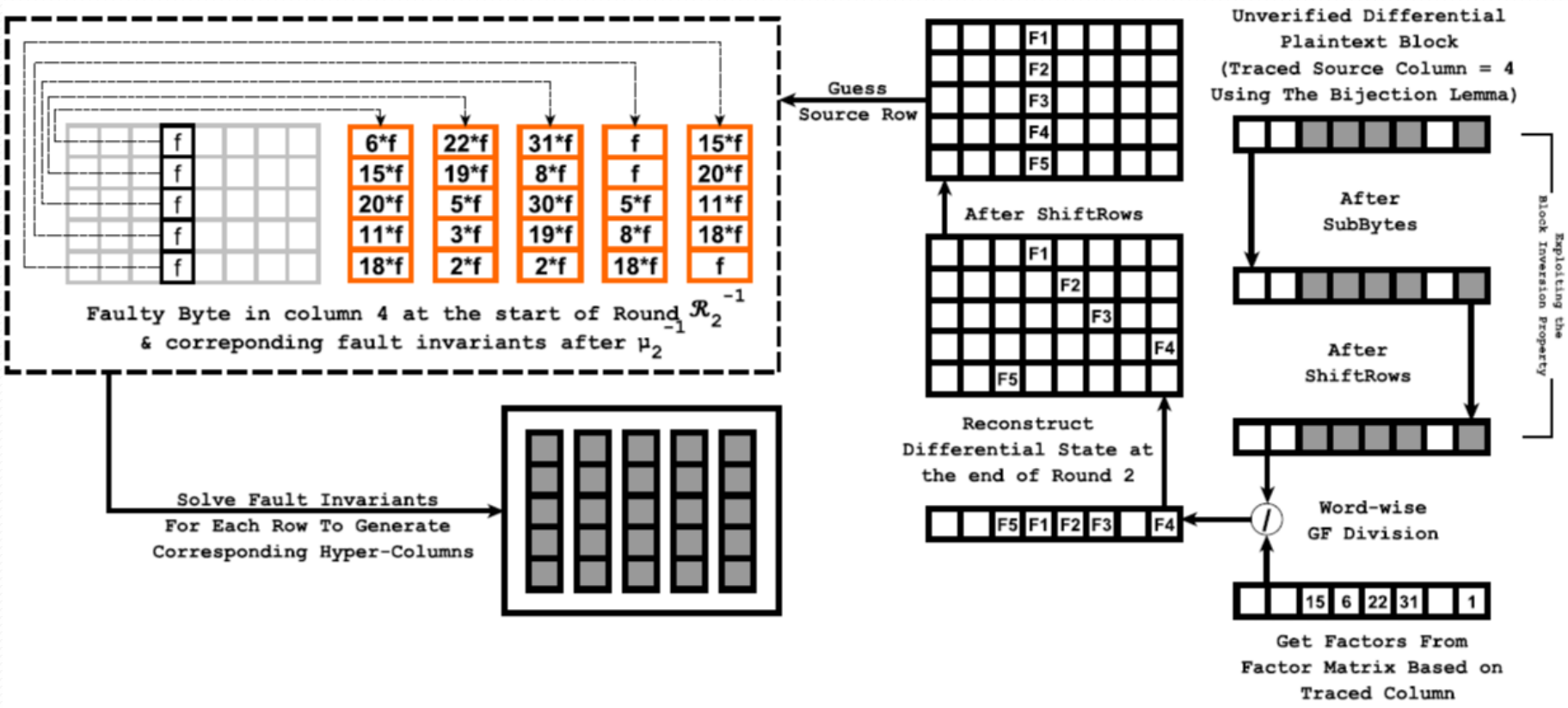
$$t_{0,i} = S(v_i) \quad where, \begin{cases} v_i \in (p \oplus c), \\ S \rightarrow \texttt{PRIMATE Sbox (Table 1)} \end{cases}$$

- Surfaces due to mode of operation
- One can invert an observed unverified plaintext block
- Exploit the fact that SubBytes and ShiftRows operate word-wise



SCOPE

The INBOUND Phase



- Hyper-Column Generation
- Repeated for every faulty unverified plaintext block

INBOUND In-Action

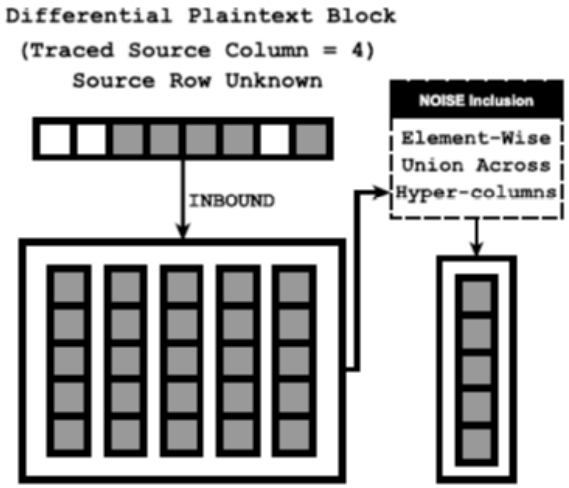
$\mathbb{H}_{0,1}$				
Hyper-Column (Row 0)	Hyper-Column (Row 1)	Hyper-Column (Row 2)	Hyper-Column (Row 3)	Hyper-Column (Row 4)
3 17 28 30	8 18	8 13 18 20	∅	2 5 6 7
9 15 19 24	21 28	3 8 30 31	∅	11 14 16 29
8 15 16 24	4 23	16 24 25 26	∅	3 6 7 13
5 10 18 19	8 26	1 6 20 31	∅	4 16 29 30
9 16 24 30	14 18	8 9 15 30	∅	5 12 20 22

Table 7: $\mathbb{H}_{0,1} \leftarrow \textsc{Inbound}\left(p, c, p'_1\right)$

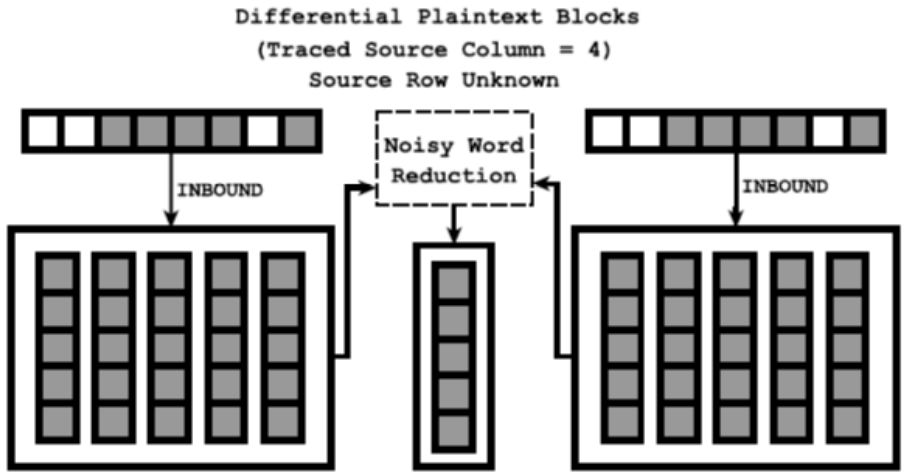
$\mathbb{H}_{0,2}$				
Hyper-Column (Row 0)	Hyper-Column (Row 1)	Hyper-Column (Row 2)	Hyper-Column (Row 3)	Hyper-Column (Row 4)
10 19 23 28	∅	∅	3 7 8 16	∅
5 12 24 25	∅	∅	4 6 11 21	∅
1 9 22 29	∅	∅	2 6 16 23	∅
4 17 20 21	∅	∅	2 7 14 16	∅
6 18 27 30	∅	∅	10 12 17 25	∅

Table 8: $\mathbb{H}_{0,2} \leftarrow \textsc{Inbound}\left(p, c, p'_{11}\right)$

Noise Handling



(a) NOISE Inclusion if Only One Fault Traced back to a Column



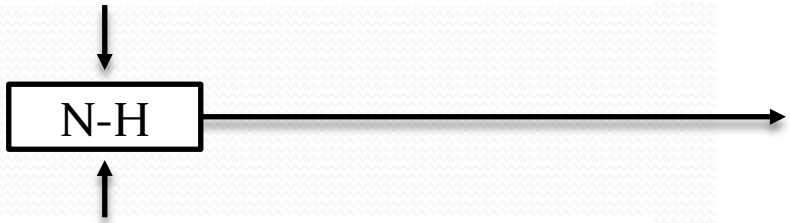
(b) NOISE Reduction if Multiple Faults Traced back to the same Column

- Recall that row position of source fault cannot be ascertained
- Exploits prospect of different faults being traced back to same column

Noise Handling In-Action

$\mathbb{H}_{0,1}$				
Hyper-Column (Row 0)	Hyper-Column (Row 1)	Hyper-Column (Row 2)	Hyper-Column (Row 3)	Hyper-Column (Row 4)
3 17 28 30	8 18	8 13 18 20	∅	2 5 6 7
9 15 19 24	21 28	3 8 30 31	∅	11 14 16 29
8 15 16 24	4 23	16 24 25 26	∅	3 6 7 13
5 10 18 19	8 26	1 6 20 31	∅	4 16 29 30
9 16 24 30	14 18	8 9 15 30	∅	5 12 20 22

Table 7: $\mathbb{H}_{0,1} \leftarrow \textsc{Inbound}(p, c, p'_1)$



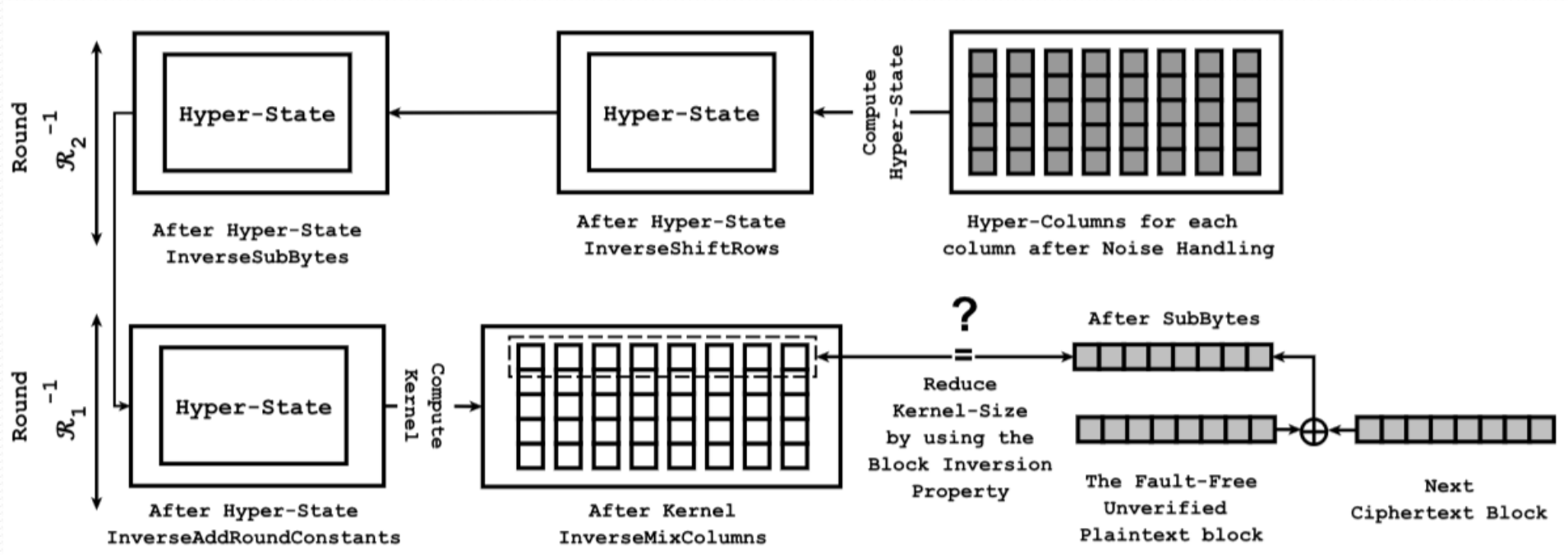
$\mathbb{H}_{0,2}$				
Hyper-Column (Row 0)	Hyper-Column (Row 1)	Hyper-Column (Row 2)	Hyper-Column (Row 3)	Hyper-Column (Row 4)
10 19 23 28	∅	∅	3 7 8 16	∅
5 12 24 25	∅	∅	4 6 11 21	∅
1 9 22 29	∅	∅	2 6 16 23	∅
4 17 20 21	∅	∅	2 7 14 16	∅
6 18 27 30	∅	∅	10 12 17 25	∅

Table 8: $\mathbb{H}_{0,2} \leftarrow \textsc{Inbound}(p, c, p'_{11})$

$\mathcal{H}_0$ Final Hyper-Column for Column 0 (After HandleNoise)
7
11
6
16
12

Table 9: $\mathcal{H}_0 \leftarrow$ Noise Reduction $(\mathbb{H}_{0,1}, \mathbb{H}_{0,2})$

The OUTBOUND Phase



- Exploits knowledge of fault-free plaintext block
- Enumerates each column-kernel and verifies first word with corresponding word in the plaintext block

SCOPE ALGO

procedure SCOPE$(p, c, \{p'_1, p'_2, \cdots, p'_n\})$

$\left\{\begin{array}{c} p \rightarrow \text{Fault-free unverified plaintext} \\ c \rightarrow \text{Next ciphertext block} \\ p'_i \rightarrow \text{Faulty unverified plaintext} \\ n \rightarrow \#\text{ of faulty outputs} \end{array}\right\}$

for $i = 0 : 7$ **do**

  $\mathcal{F}(i) = 0$ ▷ Initialize fault count vector

end for

for $i = 1 : n$ **do**

  $col \xleftarrow{\text{Lemma (1)}} (p \oplus p'_i)$ ▷ Get faulty column using the Bijection Lemma

  $\mathcal{F}(col) = \mathcal{F}(col) + 1$ ▷ Update fault count vector

  $\mathbb{H}_{col,\mathcal{F}(col)} \leftarrow \text{INBOUND}(p, c, p'_i)$ ▷ Get set of hyper-columns for column col (Fig 5)

end for

$\{\mathcal{H}_0, \mathcal{H}_1, \cdots, \mathcal{H}_7\} \leftarrow \text{HANDLENOISE}(\mathcal{F}, \mathbb{H})$ ▷ Final set of 8 hyper-columns

$s^h \leftarrow \{\mathcal{H}_0, \mathcal{H}_1, \cdots, \mathcal{H}_7\}$ ▷ Construct Hyper-State

$\mathcal{K}_{red}^{t^h} \leftarrow \text{OUTBOUND}(s^h, p \oplus c)$ ▷ Get Reduced Kernel using the Block Inversion Property (Fig. 7)

$\mathbb{S} = \varnothing$ ▷ Initialze final state-space set

for all $w \in \left(\bigtimes_{j=0}^{7} \mathcal{K}_{red}^{t^h_{*,j}}\right)$ **do** ▷ Unroll Kernel to generate state-space

  $\mathbf{s} = \beta_1^{-1}(\rho_1^{-1}(w))$

  $\mathbb{S} = \mathbb{S} \cup \{\mathbf{s}\}$

end for

return $\mathbb{S}$ ▷ Return final state-space after $\mathcal{R}_1^{-1}$

end procedure

Results & Comments

Avg. Key-space Reduction	No. of faults
2^{160} to 2^{50}	12
2^{160} to 2^{24}	16

- Fault distribution governs actual key-space reduction

$$\mathcal{F}_1 = \{1, 2, 3, 0, 2, 2, 1, 1\} \text{ and } \mathcal{F}_2 = \{2, 2, 2, 0, 2, 2, 1, 1\}$$

- Reduced key-space for F_1 is 2^{48} and for F_2 is 2^{28}
- A more uniform distribution leads to better reduction

Summary

- Knowledge of unverified plaintexts serves as a new source of side-channel information under fault analysis
- Exploited to mount SCOPE attack on APE decryption
- Attacking decryption more feasible due to prospect of nonce bypass
- This work also addresses a associated bigger problem of fault attacks with partial state information

THANK YOU

http://de.ci.phe.red

http://cse.iitkgp.ac.in/~dhimans

http://de.ci.phe.red