EnCounter: On Breaking the Nonce Barrier in Differential Fault Analysis with a Case-Study on PAEQ

Dhiman Saha, Dipanwita Roy Chowdhury

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

Presented By:

Santosh Ghosh

CHES 2016
UCSB, California, USA

NONCE

'Lets start with some ~~Nonsense~~ **Nonce-Sense**'

Nonce

- Often expanded as (N)umber-Once
- Nonce based encryption : Formalized by Rogaway

Basic Idea

The security proofs rely on the pre-condition of the *uniqueness of the nonce* in every instantiation of the cipher

- So, repetition is prohibited
- Allowed in certain designs
  - "With terms and conditions applied"

Fault Analysis

Inject - Observe - Analyze

- A very popular Side-channel Attack
- Attack the implementation

Basic Idea

Cryptanalyzing a cipher by observing its behaviour under the influence of faults.

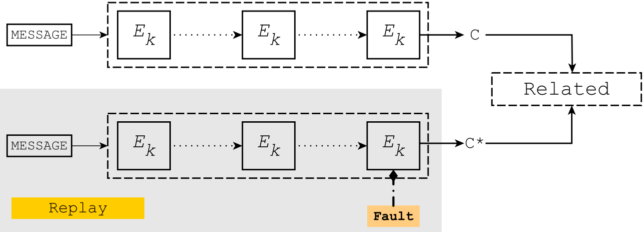
- So, first inject faults in a cryptosystem
- Then exploit information leaked by faulty output
- Most effective analysis strategy :

DFA $\leftrightarrow$ Differential Fault Analysis

Differential Fault Analysis (DFA)

The Assumption : **Replaying criterion**

The attacker must be able to induce faults while **replaying** a previous fault-free run of the algorithm.



The Possibility

Performing a differential analysis starting from an intermediate state of the cipher.

The Implication

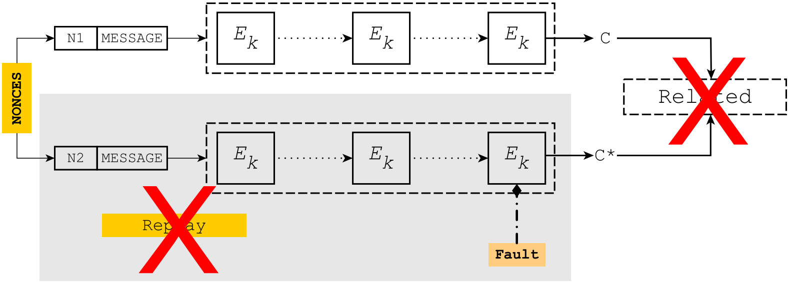
Equivalent to cryptanalyzing a round-reduced version of the cipher.

What happens in the presence of a Nonce?

Hint: Assumption Violated!

- **Replaying Criterion** no longer holds
- DFA fails
- Nonce $\Longrightarrow$ Automatic DFA Counter-measure

The Nonce Barrier



How to counter the counter-measure?

Misuse - Bypass - Avoid

Exploiting Nonce-Misuse Resistance

↑ INDOCRYPT14: Concept of *faulty collisions* demonstrated to apply DFA on nonce misuse resistant AE scheme APE

↓ Solution restricted to a single scheme

Nonce-Bypass by Attacking Decryption

↑ SAC15: DFA applied on APE decryption exploiting Release of Unverified Plaintexts (RUP) property

↓ Possible applications restricted to RUP schemes

Avoiding the Nonce by using Internal DFA

↑ This Work: Introduces internal differential fault analysis

↑ Applies to parallelizable ciphers in the counter mode

Introducing
Internal Differential Fault Analysis

"Divide and Rule"

Internal Differential Fault Analysis (IDFA)

Primary Target

Modes that use easily cancelable differences between invocations of a cryptographic primitive like a block cipher

Example: Parallelizable ciphers using the counter mode

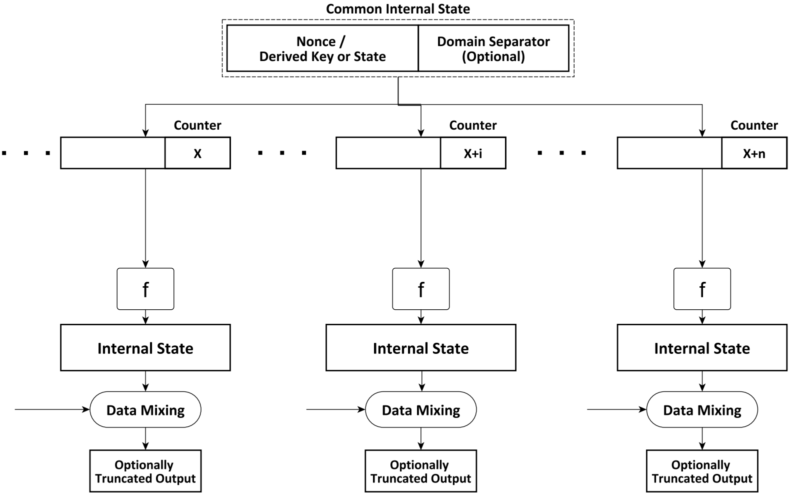
- Inputs differ only in the counter value

Main Idea

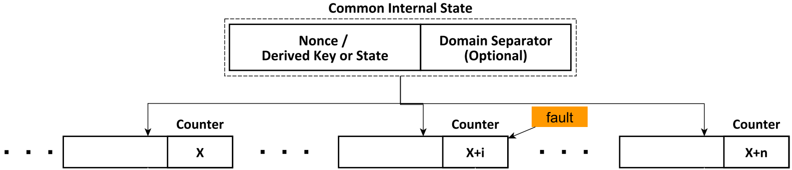
- Use first fault to cancel the input difference
- Use a second fault to generate a more standard fault attack

Requires a **single** run of the algorithm $\Longrightarrow$ **Nonce-independence**

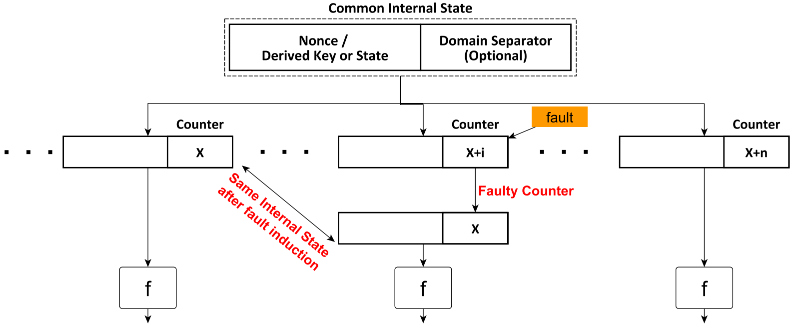
Parallel Cipher in Counter Mode — Generic-View



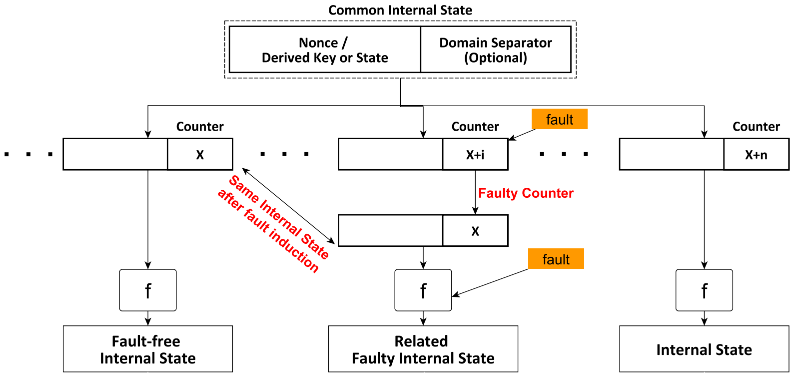
Primary fault in counter



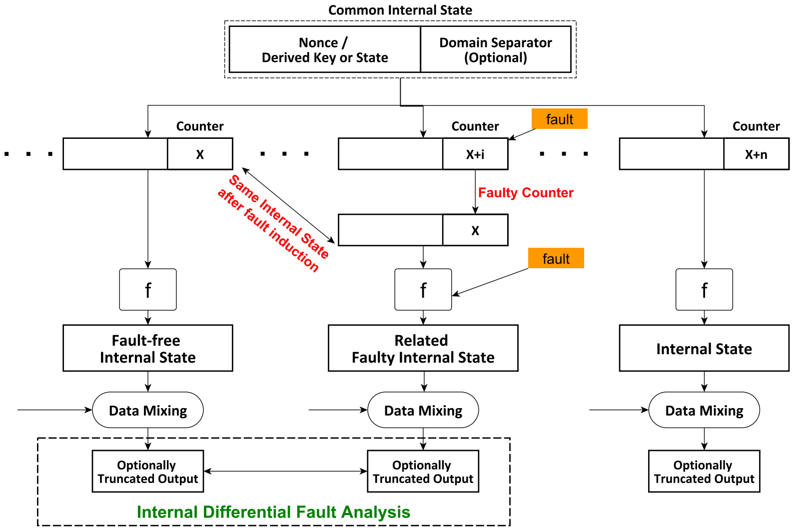
The Counter-Collision



Secondary fault in same branch



Exploiting internal differentials



The Case-Study : From Generic to Specific

"We Pick **PEAQ**!"

Why pick PAEQ?

- Meets basic criteria : Parallelizable + Counter Mode
- Underlying permutation follows AES $\implies$ An edge w.r.t DFA
- The mode of operation
- Among 30 Round 2 candidates of CAESAR

Due to the mode of operation:
Inputs to the internal permutation are only linked by counters

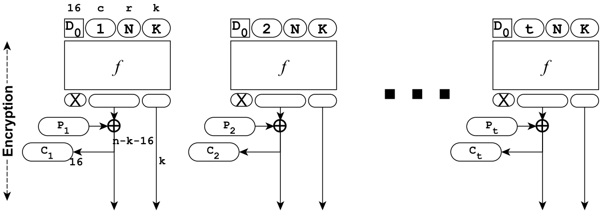
This property makes PAEQ a prime candidate to apply the concept of **fault based internal differentials** proposed in this work.

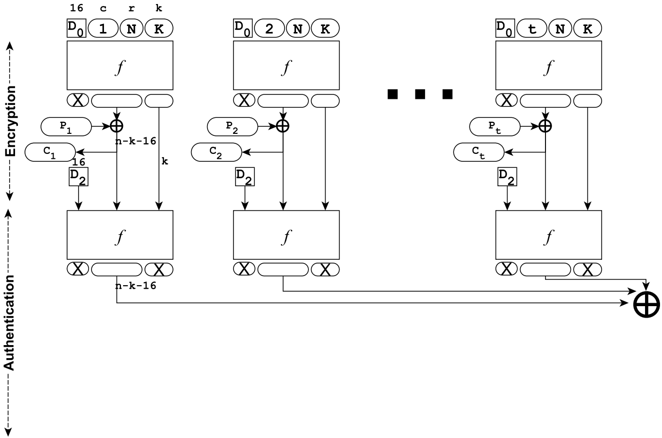
PAEQ $\leftrightarrow$ Parallelizable Authenticated Encryption based on Quadrupled AES

- An Authenticated Encryption scheme
- Fully parallelizable + On-line
- Introduced by Biryukov and Khovratovich in ISC 2014
- Along with a new generic mode of operation PPAE
  - Parallelizable Permutation-based Authenticated Encryption
- And an AES based permutation AESQ
- Security level up to 128 bits & higher, equal to the key length

Breaking News

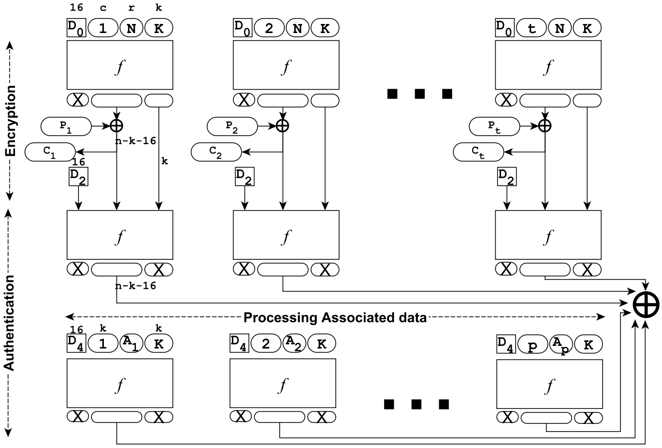
Round-3 CAESAR Candidates Announced.
PAEQ did not make it!

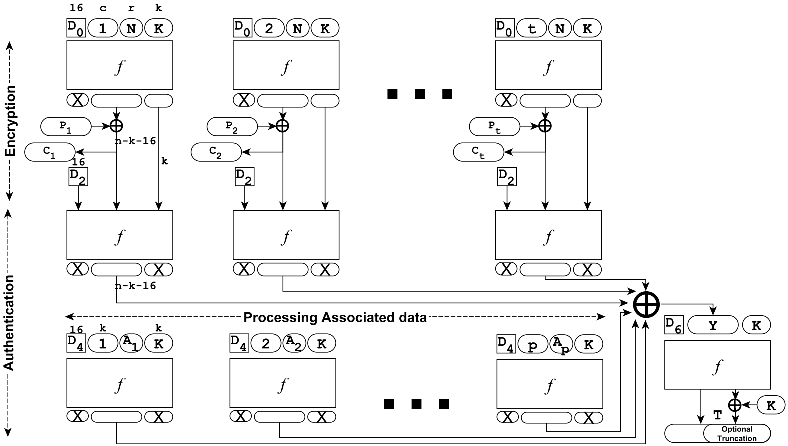

16
c
r
k
D_0
1
N
K
f
X
P_1
C_1
n-k-16
16
k
D_0
2
N
K
f
X
P_2
C_2
D_0
t
N
K
f
X
P_t
C_t
Encryption



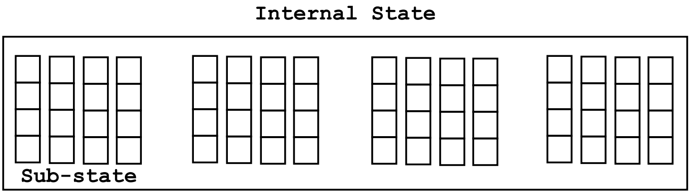
Encryption
Authentication
16 c r k
D0 1 N K
f
P1
C1
n-k-16
16
k
D2
f
n-k-16
D0 2 N K
f
P2
C2
D2
f
D0 t N K
f
Pt
Ct
D2
f

Handling Associated Data

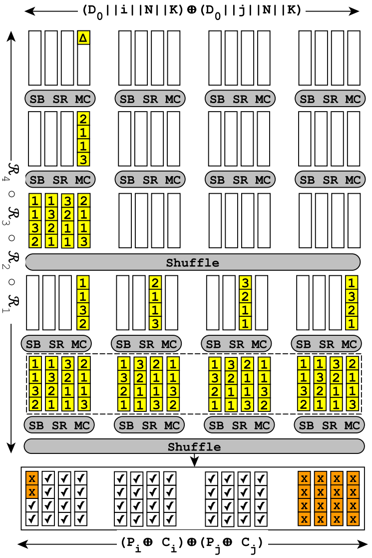



Encryption
Authentication
16 c r k
D0 1 N K
f
X
P1
C1
n-k-16
k
16
D2
D0 2 N K
P2
C2
D0 t N K
Pt
Ct
Processing Associated data
16 k k
D4 1 A1 K
D4 2 A2 K
D4 p Ap K
D6 Y K
K
T
Optional Truncation

- Internal state size of 512 bits
- Comprises of 4 sub-states of 128 bits each
- Sub-states correspond to AES state matrix
- AESQ is a composition of 20 round functions with a Shuffle operation after every 2 rounds.
- Every round applies a composition of four bijective functions which are basically the standard AES round operations



Diffusion of Internal Difference

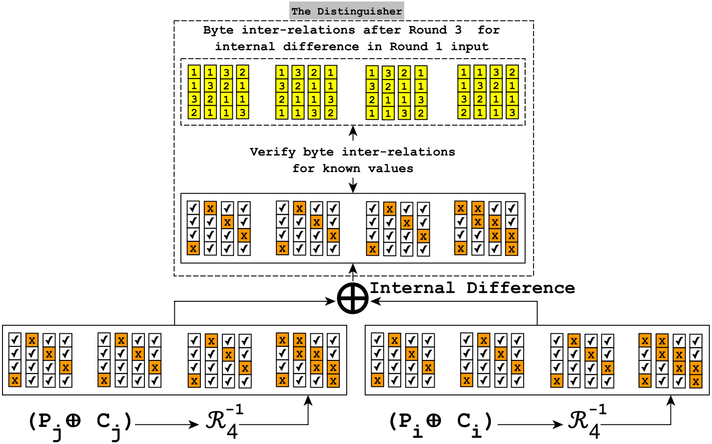


Observation

Two parallel branches of PAEQ with the same domain separator *differ only in the counter value.*

- PAEQ encryption phase
- Any two parallel branches
- Internal difference in the input limited to a byte
- Observe that bytes become related after Round 3
- These relations lead to a distinguisher

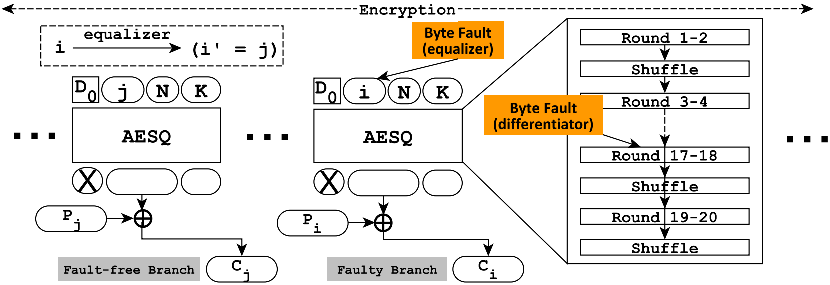
4-Round Internal-Differential Distinguisher



- Distinguisher works by verifying byte-interrelations after inverting known values of fourth round
- Used to develop concept of Fault Quartets

The Fault Model

"equalize then differentiate"



equalizer

- In last byte of Counter
- Intended for Counter collision of two branches

differentiator

- Anywhere in the state
- Creates one-byte internal difference in Round-17 input

Note: Distinguisher shown earlier can now be verified from Round-20

Introducing
Fault Quartets

Finding fault-free branch using faulty branch

Fault Quartet

- Configuration of four internal states : $\mathcal{Q}_{i,j} = \{s, s^\#, t, t^\#\}$

- $s, t \rightarrow$ branch input states
- $s \oplus t = \mathbf{0}$

- $s^\# = \texttt{AESQ}^{16}(s)$,
  $t^\# = \texttt{AESQ}^{16}(t)$
- $s^\#$ and $t^\#$ have an internal difference of 1 byte

- Generated using `equalizer` and `differentiator` faults
- Almost guaranteed[1] for a 255 complete block message
- Located by verifying the 4-round distinguisher from last round
- **In turn reveals location of fault-free branch**

[1] Refer paper for details

EnCounter

Fault Analysis of PAEQ using Internal Differentials

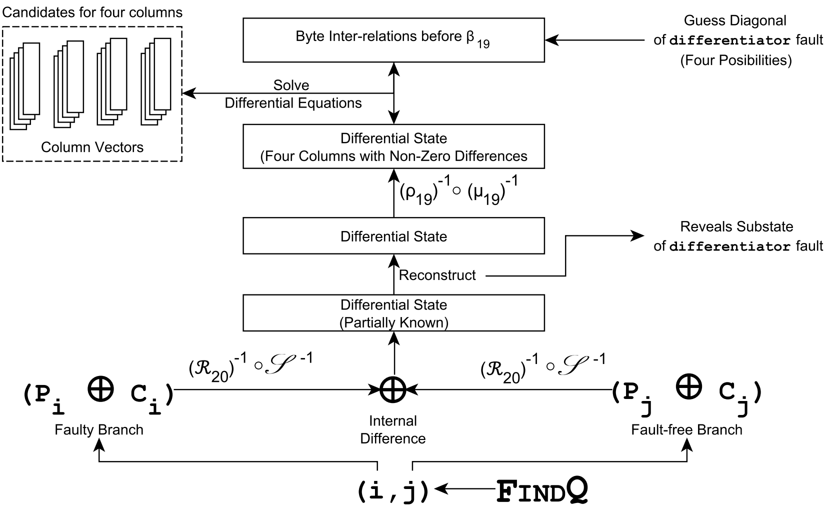
- Run PAEQ on a plaintext with **255 complete blocks**.
- Inject the `equalizer` and `differentiator` faults in any branch *i* in the encryption phase.
- Locate corresponding fault-free branch *j* by finding the Fault Quartet

$$\texttt{ENCOUNTER Input} \begin{cases} \mathtt{P} = \mathtt{P}_1 || \mathtt{P}_2 || \cdots || \mathtt{P}_i || \cdots || \mathtt{P}_j || \cdots || \mathtt{P}_{255} \\ \mathtt{C} = \mathtt{C}_1 || \mathtt{C}_2 || \cdots || \mathtt{C}_i^{\star} || \cdots || \mathtt{C}_j || \cdots || \mathtt{C}_{255} || \mathtt{Tag}^{\star} \end{cases}$$

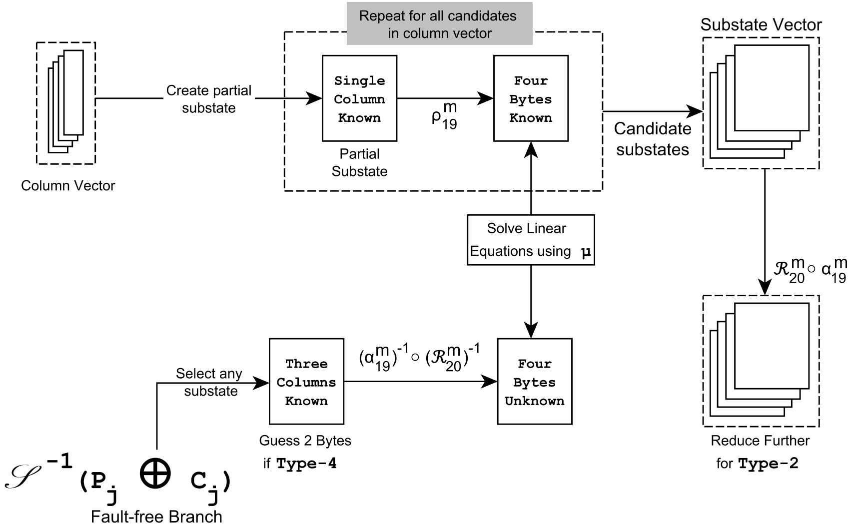
Attack works on primary PAEQ variants: `paeq-64/80/128`

- Initiate InBound phase using plaintext-ciphertext blocks of both branches
- Guess[2] **diagonal** of `differentiator` fault to compute column vectors for the state after Round-19 Subbytes
- Initiate OutBound phase using these column vectors to recover candidates of all substates after Round-20
- Finally, repeat InBound phase for every guess of the **diagonal** and consequently OutBound too
- Results accumulated as substate vectors for all Round-20 substates
- Cross-product of these vectors gives reduced state-space after Round-20 which is used to reveal the **key**

[2] Not required for `paeq-64`



Refer paper for notations

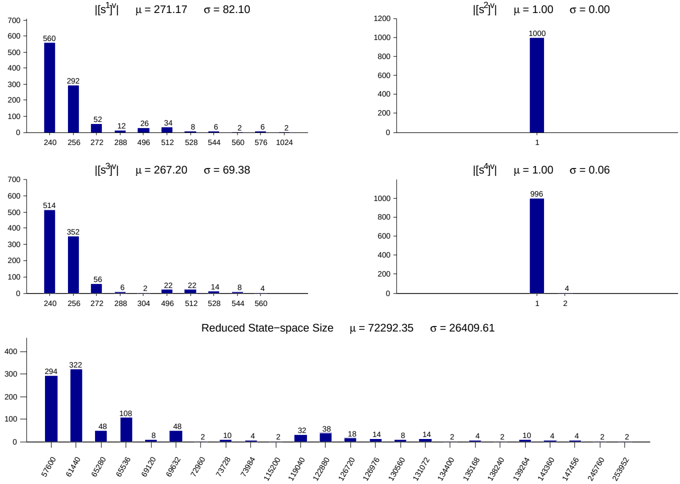


Refer paper for notations

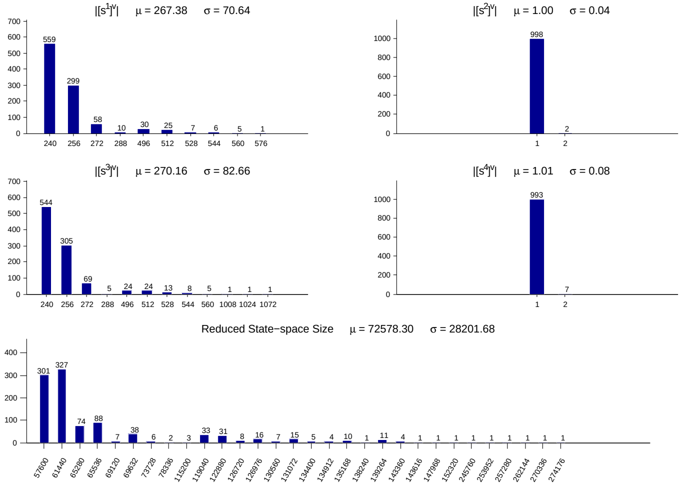
Recall : Reduced state-space after Round-20 gives the complexity

- Computer simulations performed over 1000 randomly chosen nonces, keys.
- Sizes of substate vectors along with size of the reduced state-space were noted after every experiment
- Statistical markers were studied
- Interestingly, we get similar reduction for both paeq−64 & paeq−80

PAEQ	Security-Level	Reduced State-space
paeq−64	64 bits	$2^{16.14}$
paeq−80	80 bits	$2^{16.14}$
paeq−128	128 bits	2^{50} (estd.)



Bar diagram for sizes of substate vectors and reduced state-space



Bar diagram for sizes of substate vectors and reduced state-space

Epilogue

- Introduced notion and scope of fault analysis based on internal differentials
- Proposed approach requires only one run of the algorithm thereby **overcoming the nonce barrier** of DFA
- Mount EnCounter on a **single instance** of PAEQ using two random byte faults exploiting a 4-round internal-differential property
- Achieve average key-space reductions of around 2^{16} for both paeq-64/80 and estimated about 2^{50} for paeq-128
- Presented the first analysis of PAEQ

15^{th} August: PAEQ is out of Round-3 of CAESAR Competition!

Thanks!

Sorry
for missing this
"ENCOUNTER"
with you all.

Queries

crypto@dhimans.in