Fault Tolerance : Context in Cryptography

- High-throughput requirements of various information disciplines.
- Cryptographic accelerators are needed
  - Hardware Designs implemented as ASICs and FPGAs.
  - Raises concerns regarding their reliability.
- Faults are catastrophic in context to security algorithms.
  - AES can be broken with a single well-formed fault!

Types of Fault Attacks

- Differential Fault Analysis (DFA)
  - Induce a fault
  - Observe the Difference of the correct and faulty pairs
  - Derive equations to obtain the key
- Differential Fault Intensity Attack (DFIA)
  - Obtain non-uniform faults (biased faults) through non-expensive techniques
  - Perform Side Channel Analysis like power analysis to exploit the bias

Fault Tolerant Architecture

- Countermeasures to DFA
  - Detection Based Countermeasures
    - Based on application of Classical fault tolerant techniques to cryptography
    - Uses Various Forms of Redundancy
  - Infection Based Countermeasures
    - Tries to disturb the information of a fault by infection
    - No explicit detection step

Secured Embedded
Architecture
Laboratory (SEAL)

COUNTERMEASURES VERSUS BIASED FAULTS – PUSHING THE LIMITS

Secured Embedded Architecture Laboratory (SEAL)

Countering Fault Attacks

Whose fault is It?

- Is the flaw in the algorithm?
- Is the flaw in the implementation?

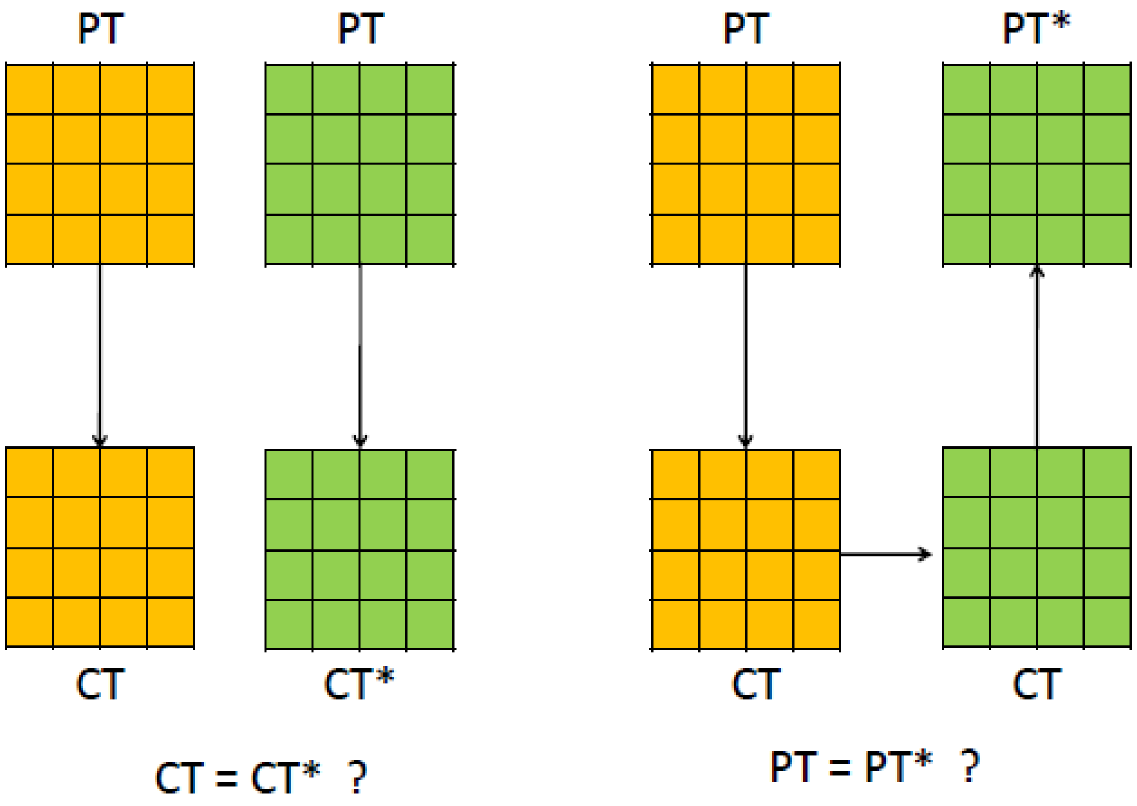
How can Countermeasures be built?

- Does Classical Fault Tolerance work?

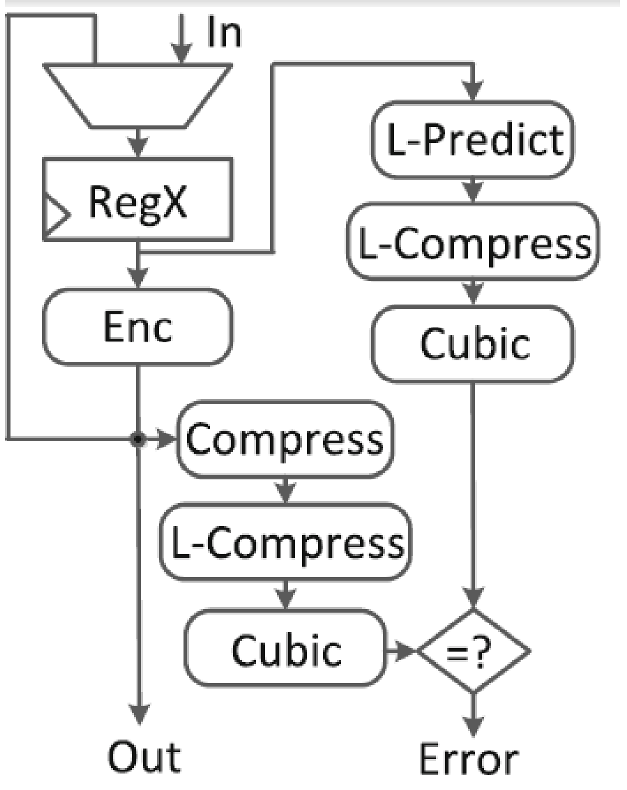
Detection Based Countermeasures

- Also known as Concurrent Error Detection (CED) techniques
- Use various kinds of redundancy to detect faults
- Vulnerable to attacks in the comparison step itself
- Vulnerable to biased fault attacks

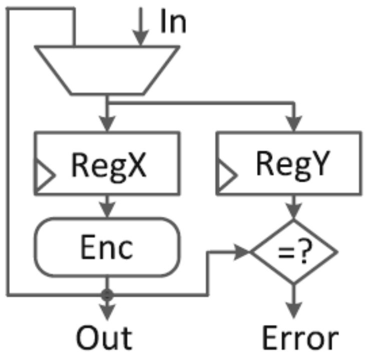
The Basic Principle of CEDs



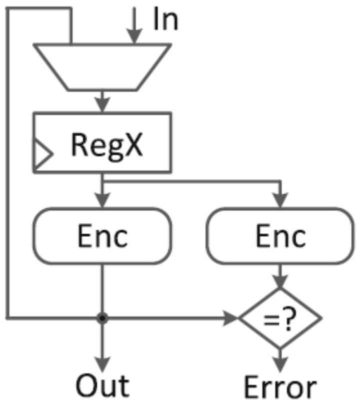
Examples of CED



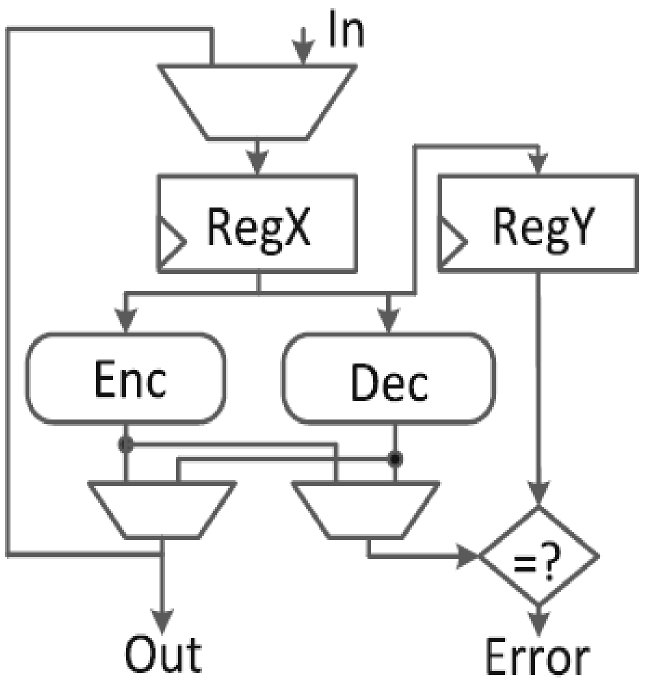
Information Redundancy – Robust Codes



Time Redundancy



Hardware Redundancy

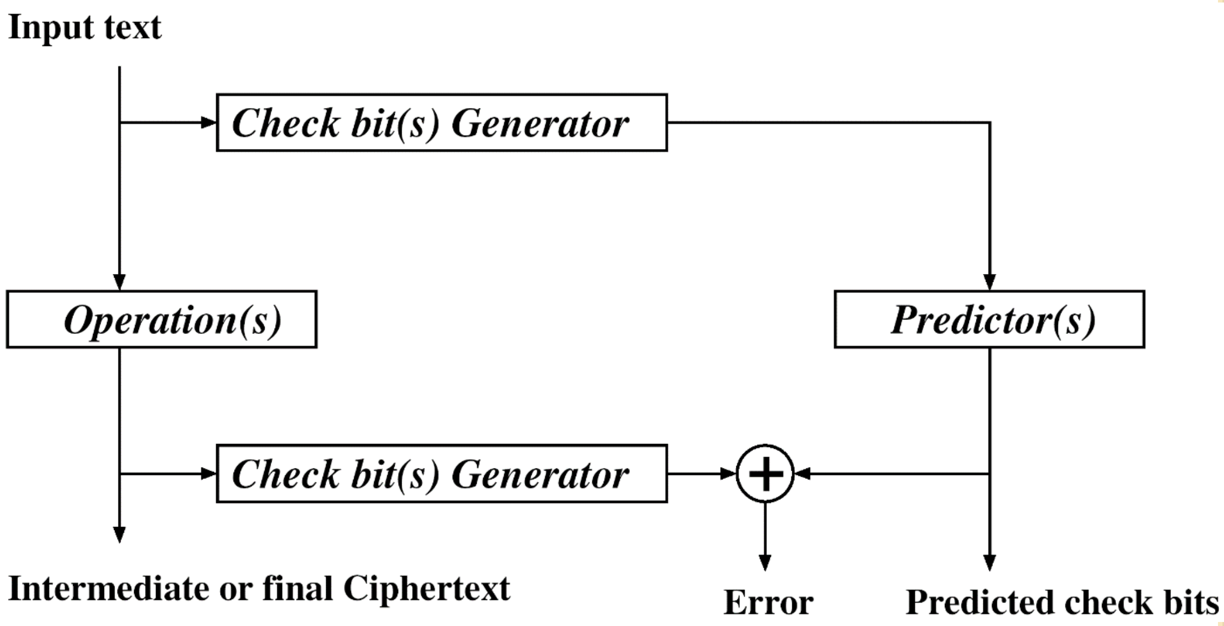


Hybrid Redundancy - REPO

Source : Guo et. al. , Security analysis of concurrent error detection against differential fault analysis – Journal of Cryptographic Engineering, 2014

Error Detecting Codes (EDCs)

- First generate check bits
- For each operation within encryption predict check bits
- Periodically compare predicted check bits to generated ones
- Predicting check bits for each operation - most complex step
  - Should be compared to duplication
- Examples of EDC – parity based and residue checks
- Can be applied at different levels – word, byte, nibble



Source : Koren and Krishna, Morgan-Kaufman 2007

Parity-based Code for AES

- Operations operate on bytes so byte-level parity is natural
- **ShiftRows:** Rotating the parity bits
- **AddRoundKey:** Add parity bits of state to those of key
- **SubBytes:** Expand Sbox to 256×9 – add output parity bit; to propagate incoming errors (rather than having to check) expand to 512×9 – put incorrect parity bit for inputs with incorrect parity
- **MixColumns:** The expressions are:

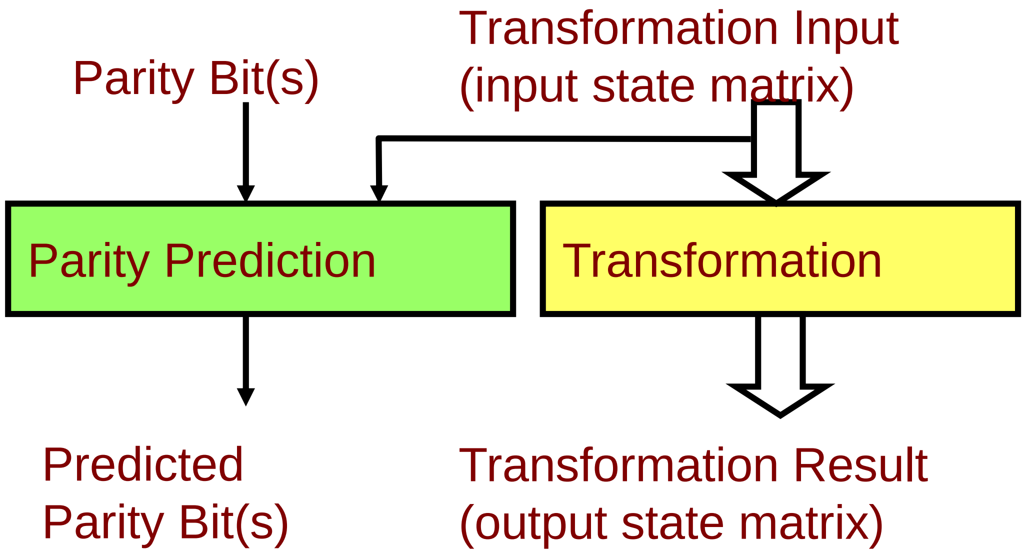
$$p_{0,j} = p_{0,j} \oplus p_{2,j} \oplus p_{3,j} \oplus S_{0,j}^{(7)} \oplus S_{1,j}^{(7)}$$

$$p_{1,j} = p_{0,j} \oplus p_{1,j} \oplus p_{3,j} \oplus S_{1,j}^{(7)} \oplus S_{2,j}^{(7)}$$

$$p_{2,j} = p_{0,j} \oplus p_{1,j} \oplus p_{2,j} \oplus S_{2,j}^{(7)} \oplus S_{3,j}^{(7)}$$

$$p_{3,j} = p_{1,j} \oplus p_{2,j} \oplus p_{3,j} \oplus S_{3,j}^{(7)} \oplus S_{0,j}^{(7)}$$

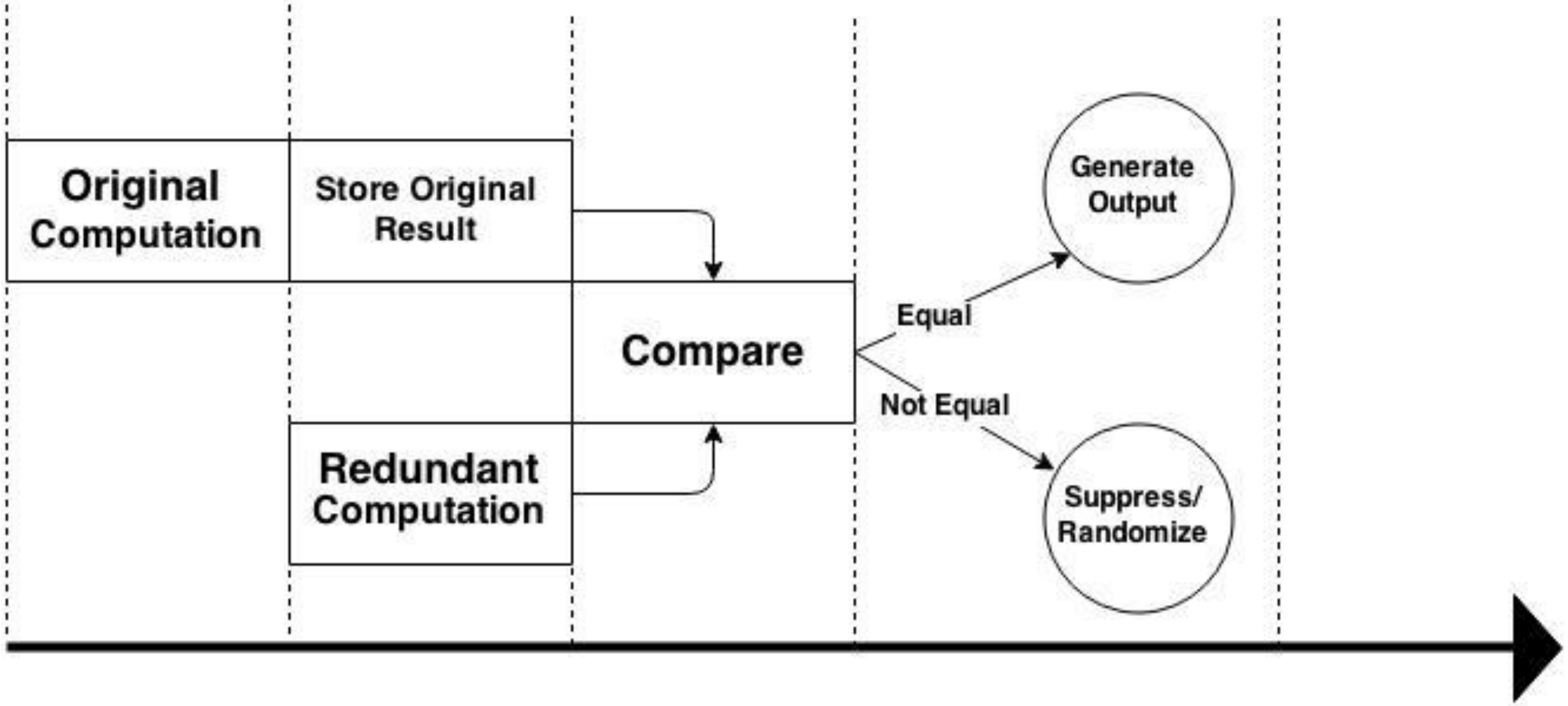
where $S_{i,j}^{(7)}$ is the msb of the state byte in position i,j



Source : Koren and Krishna, Morgan-Kaufman 2007

Does Detection Always Guarantee Security?

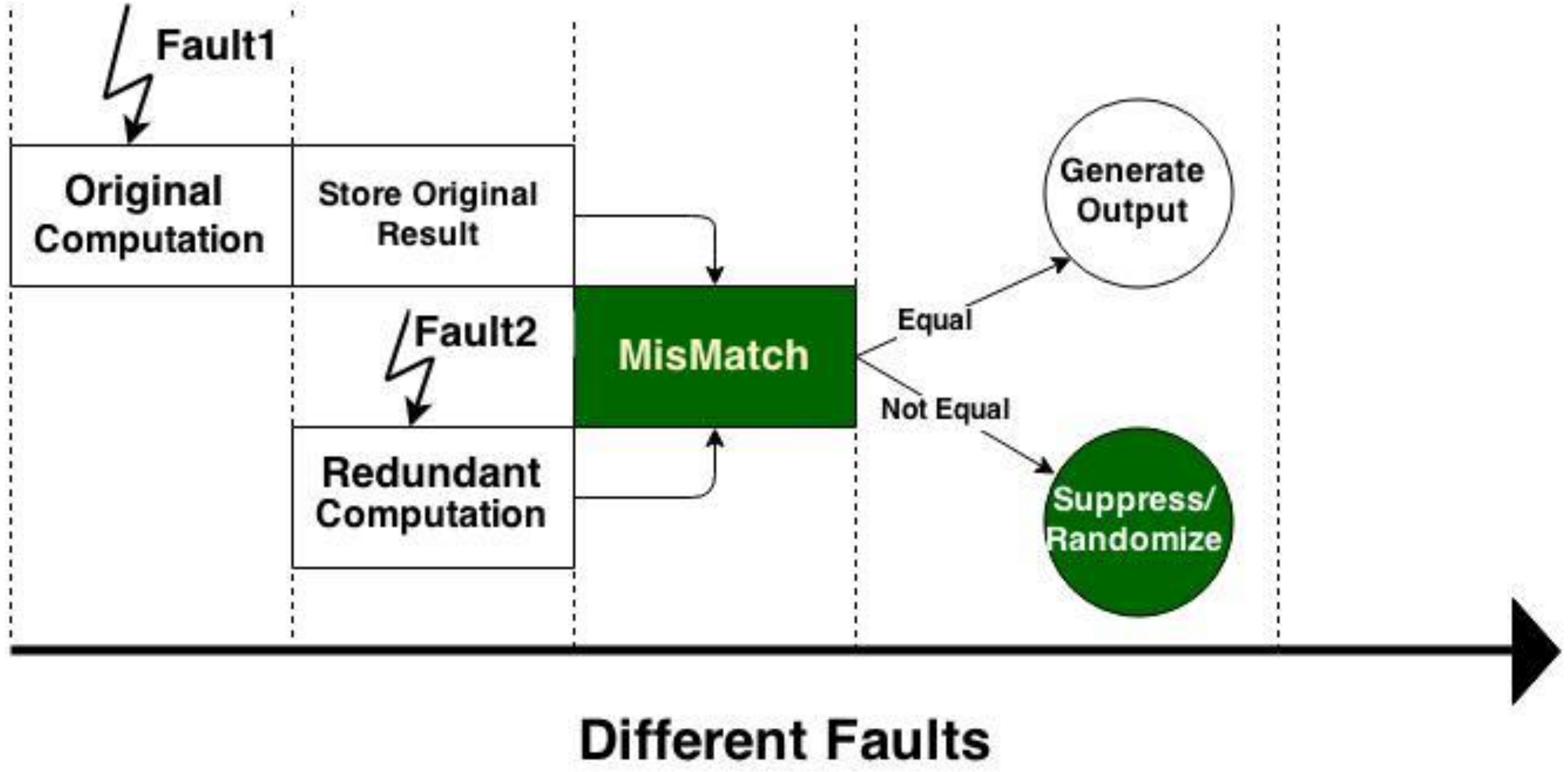
The Time Redundancy Countermeasure



S.Patranabis, A.Chakraborty, P.H.Nguyen and D.Mukhopadhyay. A Biased Fault Attack on the Time Redundancy Countermeasure for AES. In *Proceedings of Constructive Side Channel Analysis and Secure Design 2015 (COSADE 2015*), Berlin, Germany, April 2015

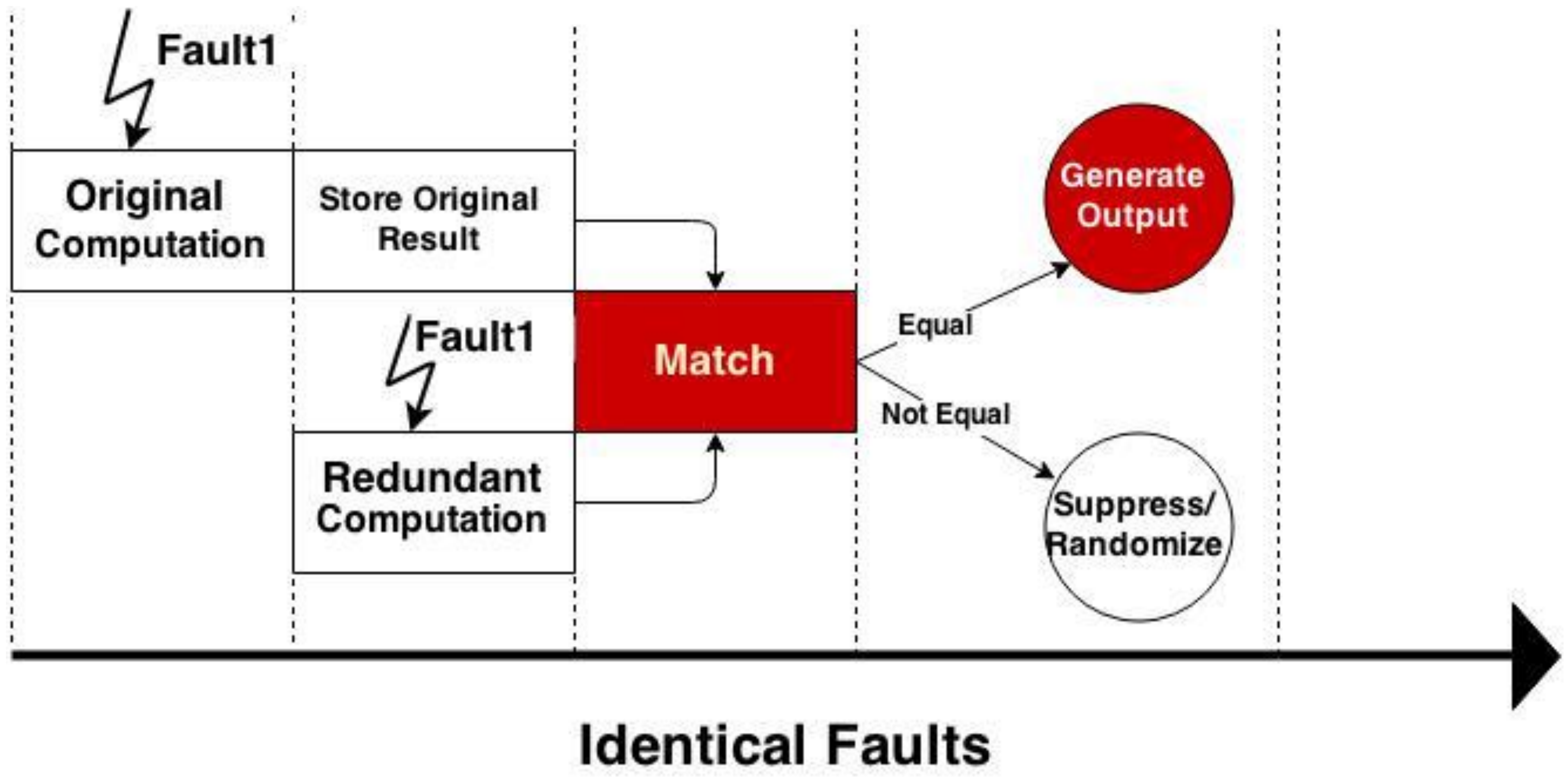
Secured Embedded Architecture Laboratory (SEAL)

Against Double Fault Attacks : Detection



Secured Embedded Architecture Laboratory (SEAL)

Against Double Fault Attacks: Misses



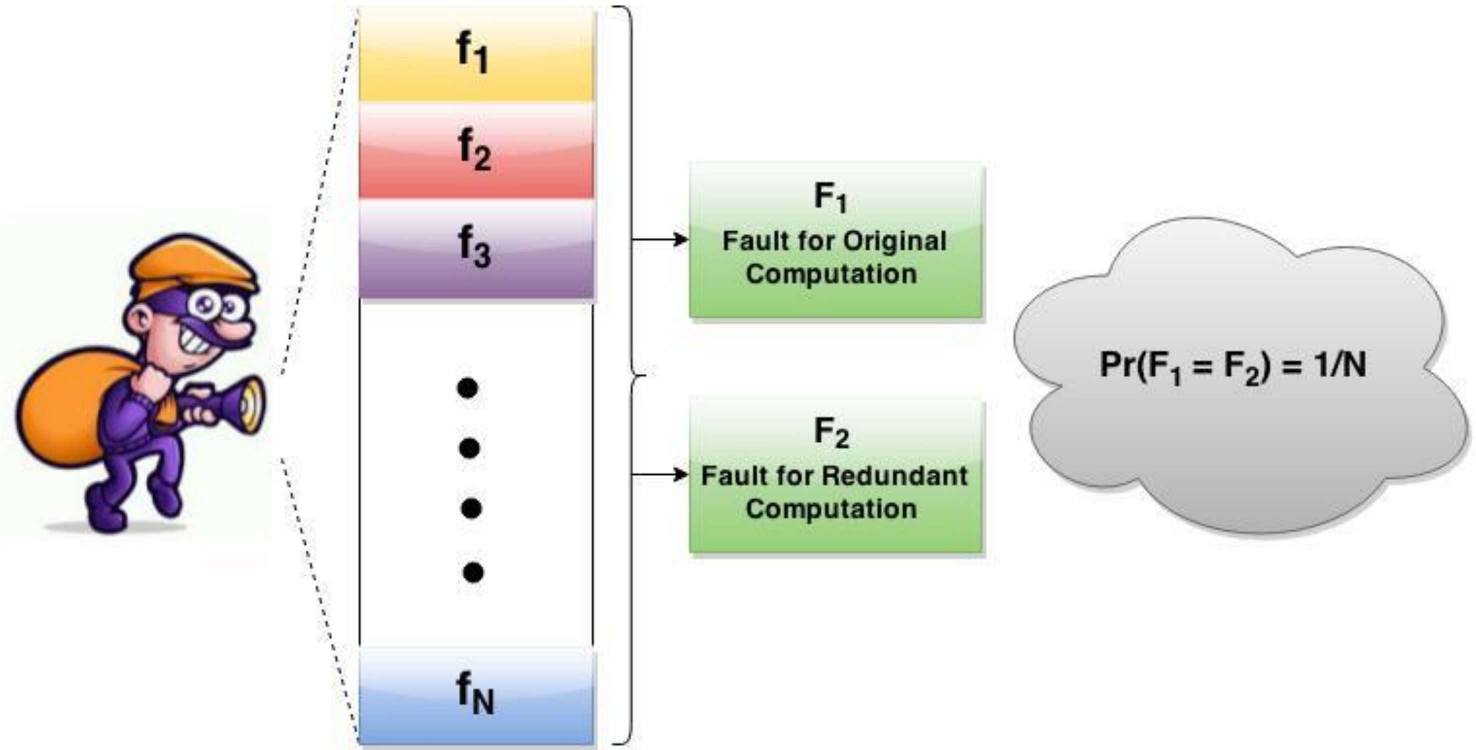
Secured Embedded Architecture Laboratory (SEAL)

Beating The Countermeasure

- Improving **fault collision probability**
  - Enhancing the probability of identical faults in original and redundant rounds
- Two major aspects
  - The size of the fault space
  - The probability distribution of faults in the fault space
- A smaller fault space enhances the fault collision probability
- A non-uniform probability distribution of faults in the fault space also enhances the fault collision probability

Uniform Fault Model

- All faults are equally likely



Biased Fault Model

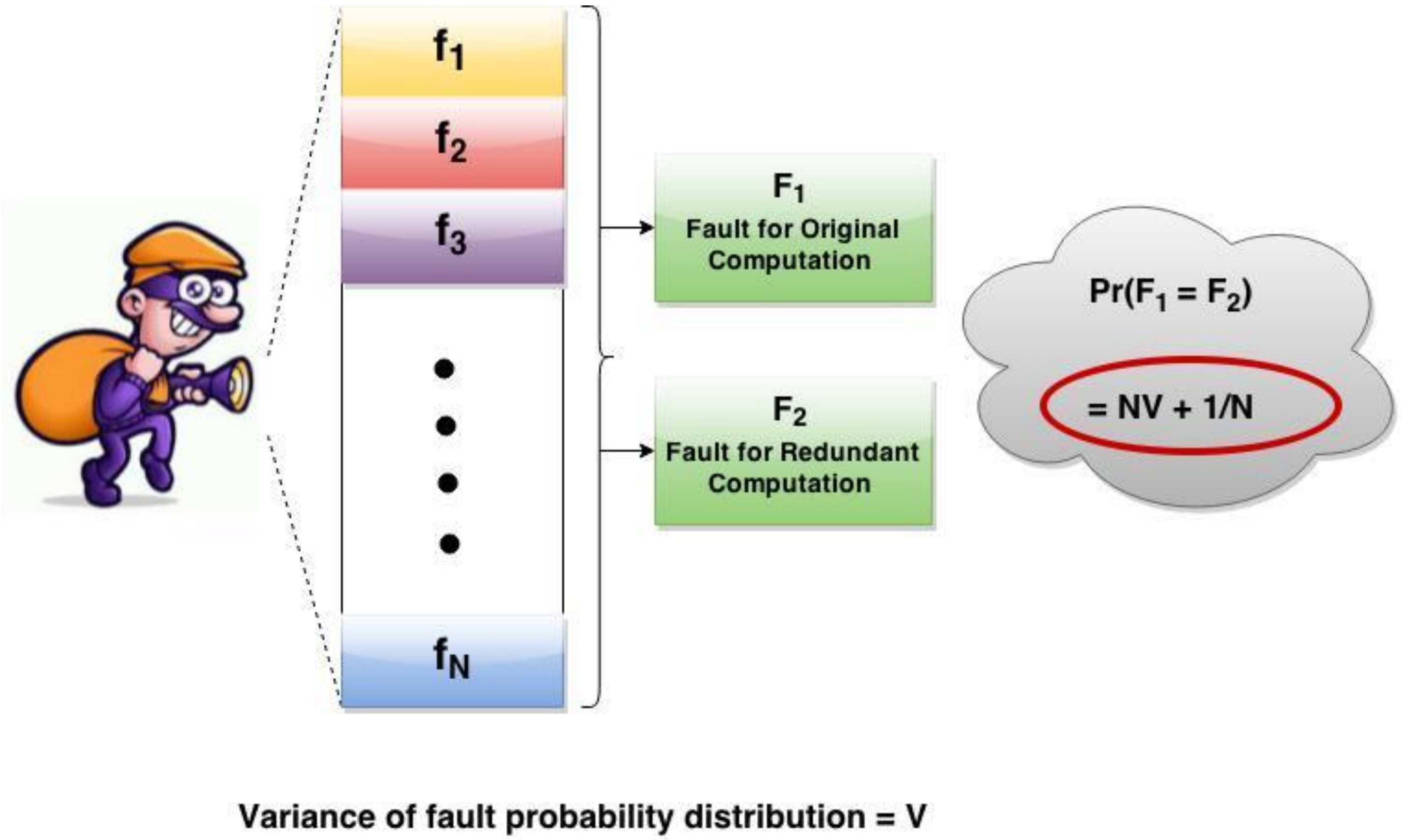
- A total of n faults possible under a fault model F
- Each fault fi has a probability of occurrence Pr[fi]
- Let V be the variance of the fault probability distribution
- Degree of Bias of a fault model increases with increase in V

Fault Model	Pr[f1]	Pr[f2]	Pr[f3]	Pr[f4]	Pr[f5]	Pr[f6]	Pr[f7]	Pr[f8]	V
1	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	**0**
2	0.225	0.200	0.175	0.125	0.100	0.075	0.050	0.050	**0.004**
3	0.500	0.250	0.125	0.050	0.050	0.025	0	0	**0.026**

Secured Embedded Architecture Laboratory (SEAL)

The Fault Collision Probability

- With increase in bias, collision probability increases



The Adversarial Perspective

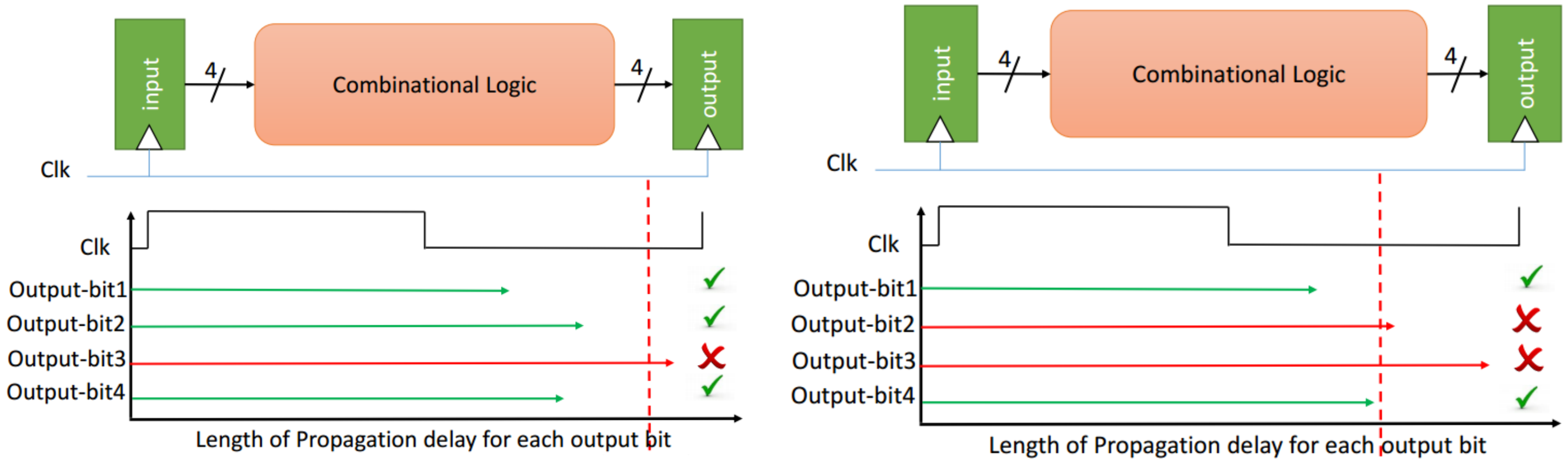
Precise Fault Models

Biased Fault Models

Secured Embedded Architecture Laboratory (SEAL)

Fault Intensity

The impact of fault varies with the tuning of the parameters of the fault inducing process.
More true for low cost equipment.



Insertion of Fault through Clock Glitches:

With increase of clock frequency more bits start getting affected.
We say the fault intensity increases!

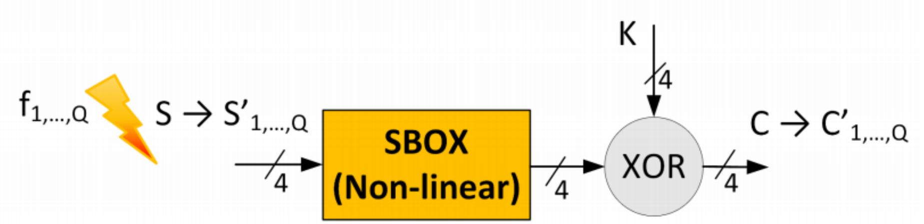
Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa M. I. Taha, Patrick Schaumont: **Differential Fault Intensity Analysis.** FDTC 2014: 49-58

Differential Fault Intensity Analysis (DFIA)

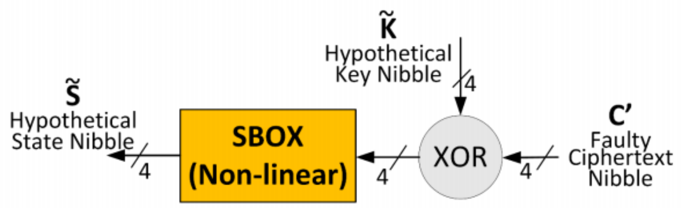
- Combines fault injection and DPA principles
- Induces biased faults by varying the fault intensity
- Applies a hypothesis test with biased faults
- Uses biased faults as the source of leakage

Steps of a DFIA

- Step 1: Biased Fault Injection
  - Apply Q different fault intensities ($f_{1,\dots,Q}$)
  - Induce biased faults ($S'_{1,\dots,Q}$)
  - Collect faulty ciphertexts ($C'_{1,\dots,Q}$)



- Step 2: Hypothesis Test with Biased Faults



Given: C' and a KNOWN fault bias f
Find: Most likely key nibble $\tilde{K}$

For all $\tilde{K}$, find $\tilde{S} = SBOX^{-1}(C' \oplus \tilde{K})$
Accumulate $\rho_{\tilde{K}} = \sum HD(\tilde{S})$
Select K = argmin ρ

The extraction of the key is like a side channel analysis: Guessing the key correctly helps in observing the bias in the fault distribution

Attack on the Time redundancy Countermeasure

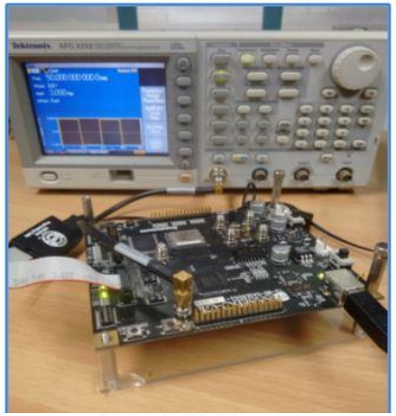
- All faults are restricted to a single byte
- Two kinds of fault models
  - Situation-1: Attacker has control over target byte
  - Situation-2: Attacker has no control over target byte
- Control over target byte makes fault model more precise but is costly to achieve

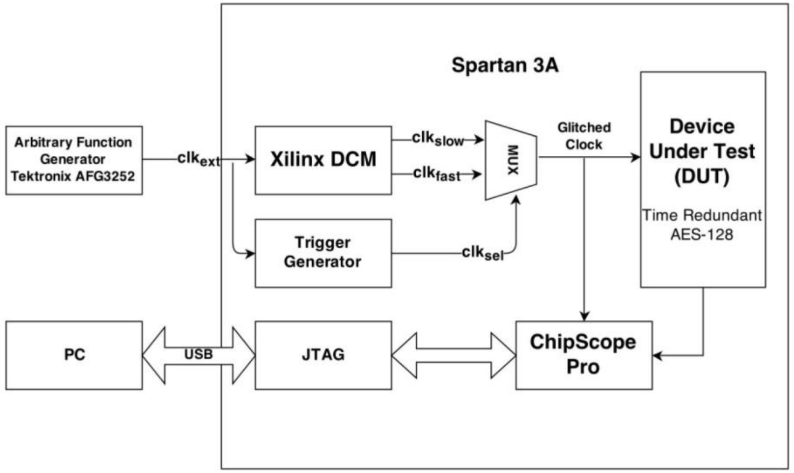
Suitable

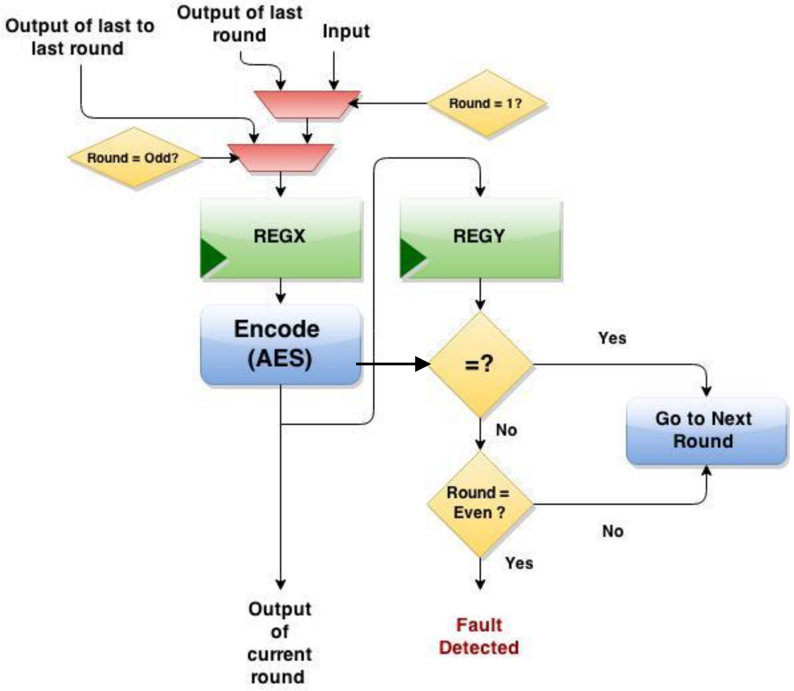
Symbol	Fault Model
FF	Fault Free
SBU	Single Bit Upset
SBDBU	Single Byte Double Bit Upset
SBTBU	Single Byte Triple Bit Upset
SBQBU	Single Byte Quadruple Bit Upset
OSB	Other Single Byte Faults
MB	Multiple Byte Faults

Fault Model	Faults Possible(n) (Situation-1)	Faults Possible(n) (Situation-2)
SBU	8	128
SBDBU	28	448
SBTBU	56	896
SBQBU	70	1120
OSB	93	1488

The Fault Injection Set-Up







- Time redundant AES-128 implemented in Spartan 3A FPGA
- Fault injection using clock glitches at various frequencies
- Xilinx DCM to drive fast clock frequency
- Internal state monitoring using ChipScope Pro 12.3

The Attack Procedure

Fault Model	Frequency range for both original and redundant rounds (MHz)
SBU	125.3-125.4
SBDBU	125.6-125.7
SBTBU	126.0-126.1
SBQBU	126.3-126.4

Fault Distribution

Distinguishers used :

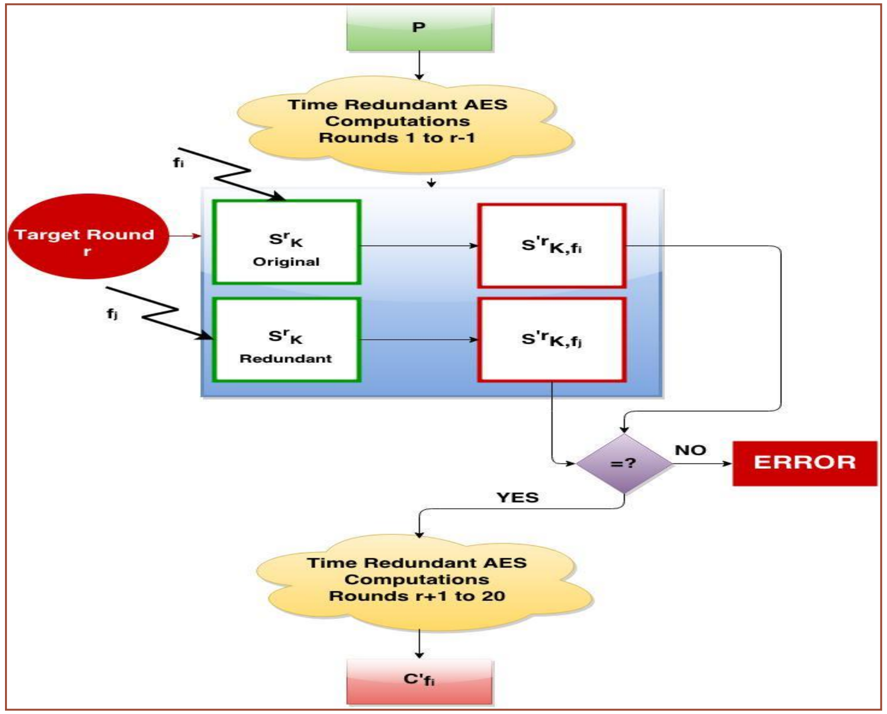
- Hamming Distance (HD)
- Squared Euclidean Imbalance (SEI)

Make a key hypothesis k and evaluate the distinguishers

Correct hypothesis gives minimum and maximum values respectively

$$H(k) = \sum_{i=1}^{N_C} \sum_{j=1}^{i-1} HD({S'^r}_{k,f_i}, {S'^r}_{k,f_j})$$

$$S(k) = \sum_{i=1}^{N_C} \sum_{\delta=0}^{255} \left(\frac{\#\{b \mid {S'^r}_{k,f_i}[b] = \delta\}}{N_C} - \frac{1}{256}\right)^2$$



Secured Embedded Architecture Laboratory (SEAL)

Simulations- I

Number of ciphertexts required to guess the AES key with 99% accuracy

- Identical faults introduced into both original and redundant rounds
- Target byte chosen at random
    - Same fault for original and redundant computations
    - Each fault injection yields a *useful ciphertext*
- Attacks simulated on rounds 8 and 9
- Performed separately for each fault model

Simulation results

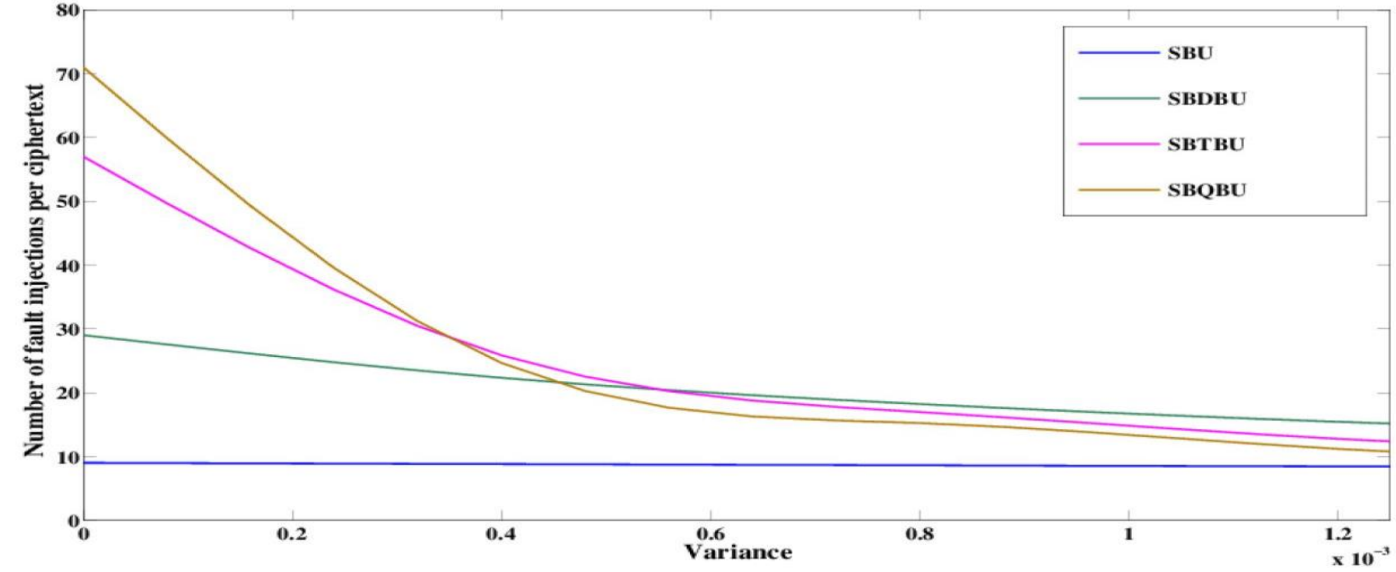
Round	Fault Model	N_C
8	SBU	320-340
	SBDBU	580-600
	SBTBU	1000-1040
	SBQBU	1900-2000
9	SBU	288-320
	SBDBU	608-640
	SBTBU	832-880
	SBQBU	1360-1440

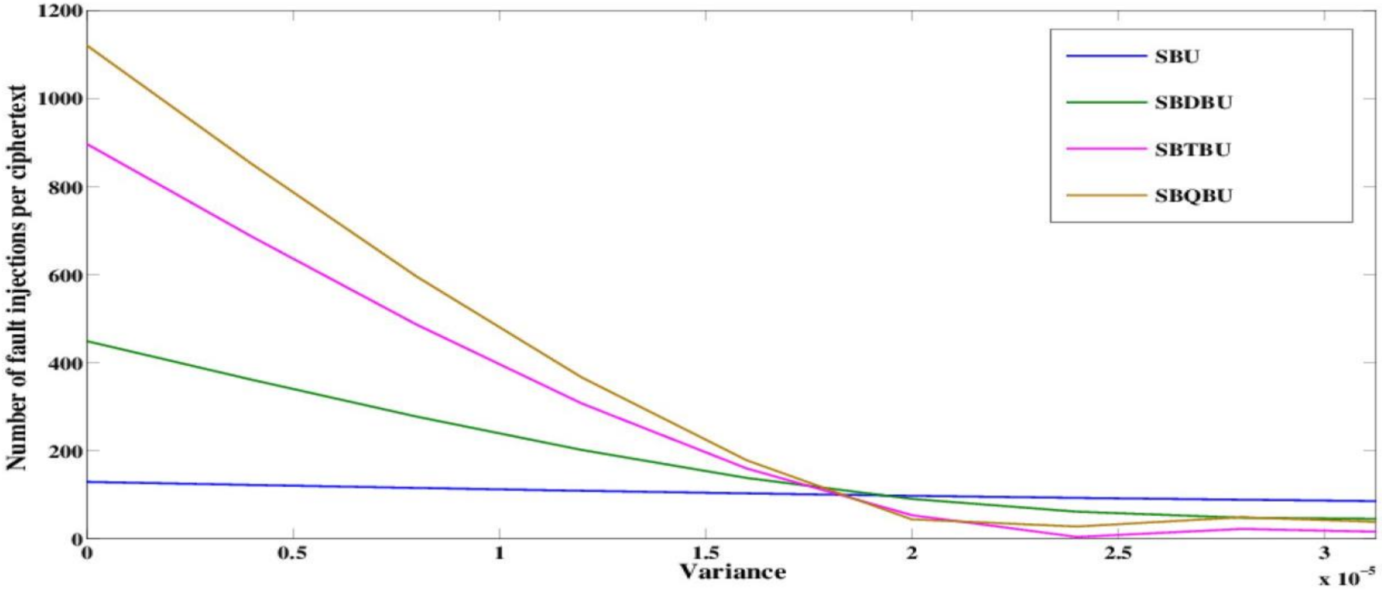
Simulations-2

- Vary the degree of bias in the fault model
  - Control the variance of the fault probability distribution
- Observe the number of fault injections to get a faulty ciphertext
- Two adversarial models:
  - Type 1: Perfect control over target byte
  - Type 2: No control over target byte

Secured Embedded Architecture Laboratory (SEAL)

Simulations-2 (contd.)





Experimental Results

Useful ciphertexts

Total Fault Injections

Round	Fault Model	Fault Variance		N_C	N_F(simulation)		N_F(experimental)	
		Type-1	Type-2		Type-1	Type-2	Type-1	Type-2
8	SBU	9.5×10^{-2}	3.6×10^{-3}	304.75	340.48	647.52	387.67	687.91
	SBDBU	1.4×10^{-2}	9.2×10^{-4}	625.12	1456.25	1506.25	1448.45	1652.30
	SBTBU	9.7×10^{-3}	4.9×10^{-4}	1020.49	1815.60	2315.40	1974.86	2395.83
	SBQBU	3.2×10^{-3}	5.9×10^{-5}	1878.55	7868.82	28038.54	8003.14	30201.41
9	SBU	9.2×10^{-2}	3.5×10^{-3}	304.24	385.88	603.11	387.98	632.71
	SBDBU	8.8×10^{-2}	7.9×10^{-4}	624.65	641.18	1487.36	647.82	1556.69
	SBTBU	8.1×10^{-2}	6.7×10^{-4}	832.32	873.56	2054.00	878.23	2489.25
	SBQBU	7.5×10^{-2}	3.5×10^{-5}	1328.22	1788.84	17239.10	1809.25	20145.66

Comments on Detection Schemes

- Bias of a fault model can be quantified in terms of the variance of fault probability distribution
- Detection based countermeasures are vulnerable against biased fault attacks that are practically achievable

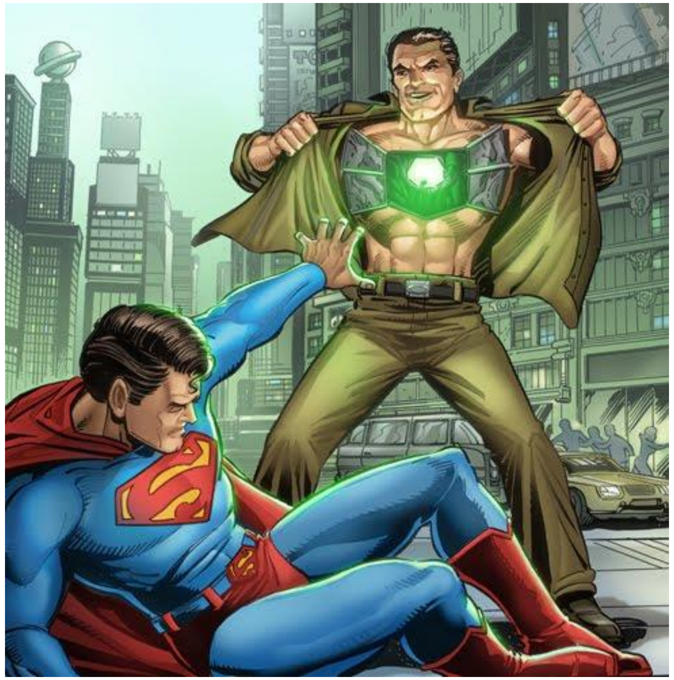
Secured Embedded Architecture Laboratory (SEAL)

- **Fault Tolerance for DFA needs to be revisited?**

Cover all of the essential

or

almost all???



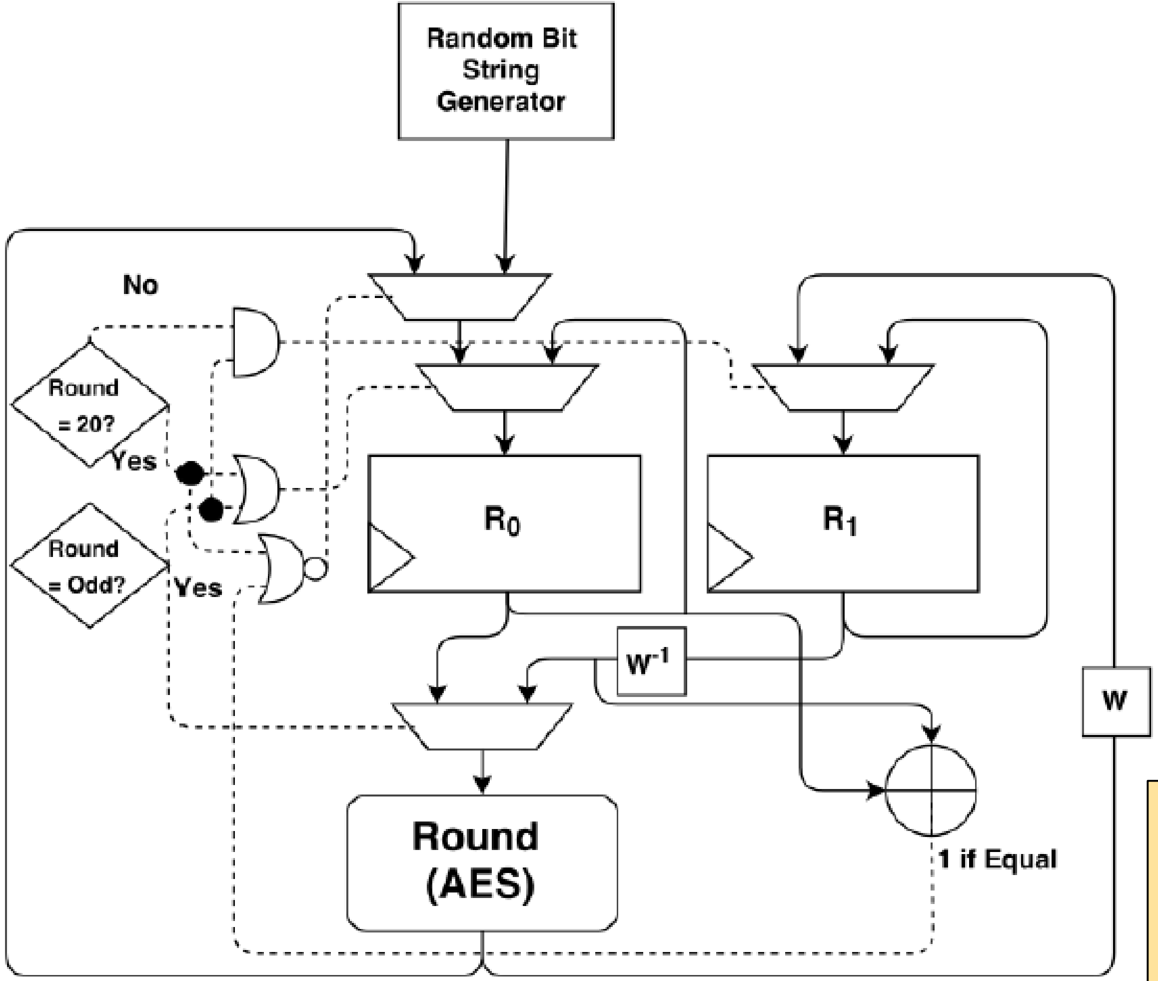
Countermeasures Must Be Augmented

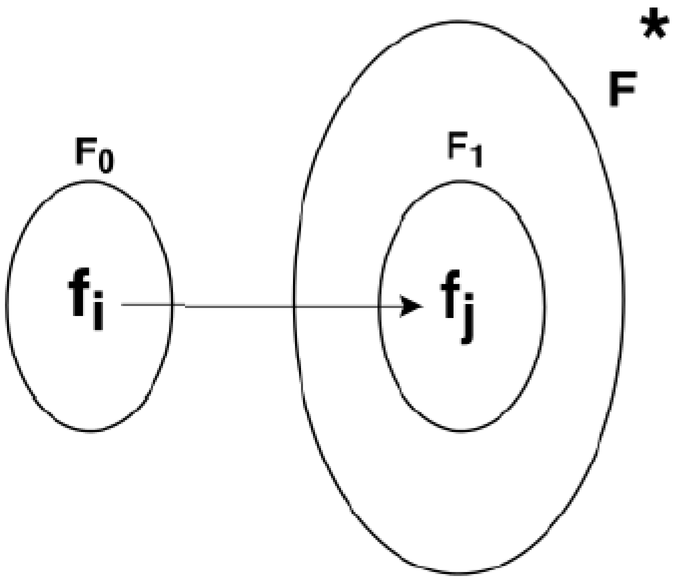
- Detection alone does not guarantee security against fault attacks, especially in the wake of biased fault models
- Need to augment the countermeasure scheme to tackle biased fault attacks
- Two possible strategies:
  - Fault Space Transformation
  - Infective Countermeasures

Fault Space Transformation

- Ensure that the adversary cannot exploit the biased nature of the fault model
- Fault spaces for the original and redundant computations are different
- Adversary cannot ensure the occurrence of equivalent faults in the two different fault spaces at the same time.

Fault Space Transformation to Counter Biased Fault Attacks



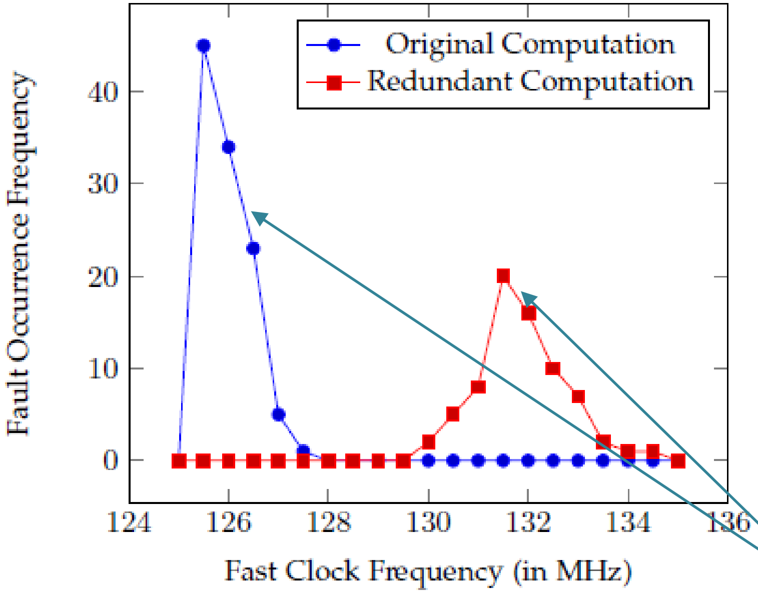


Sikhar Patranabis, Abhishek Chakraborty, Debdeep Mukhopadhyay, P. P. Chakrabarti:
Using State Space Encoding To Counter Biased Fault Attacks on AES Countermeasures. IACR Cryptology ePrint Archive 2015: 806 (2015)

The Impact of Transformation

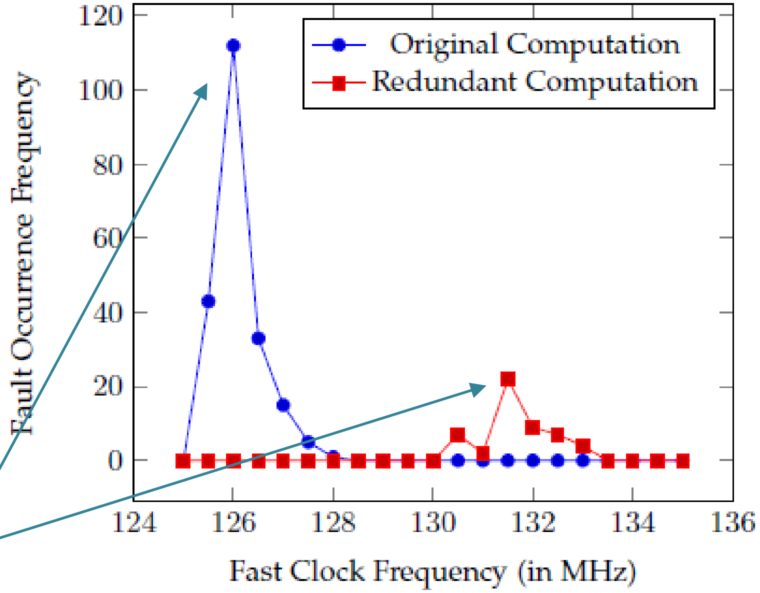
- Transforming the fault space implies that the adversary cannot beat the countermeasure by merely introducing the same fault twice
- It is most unlikely that the transformed fault space will have a one-to-one correspondence in terms of bias with the original
- Mathematically, the expected fault collision probability over all possible transformations is the same as for uniform fault models

Results on Hardware



Transformation used is the MixColumn of AES

Peaks occur at disjoint frequency regions



Single Bit Upset (SBU)

Single Byte Double Bit Upset (SBDBU)

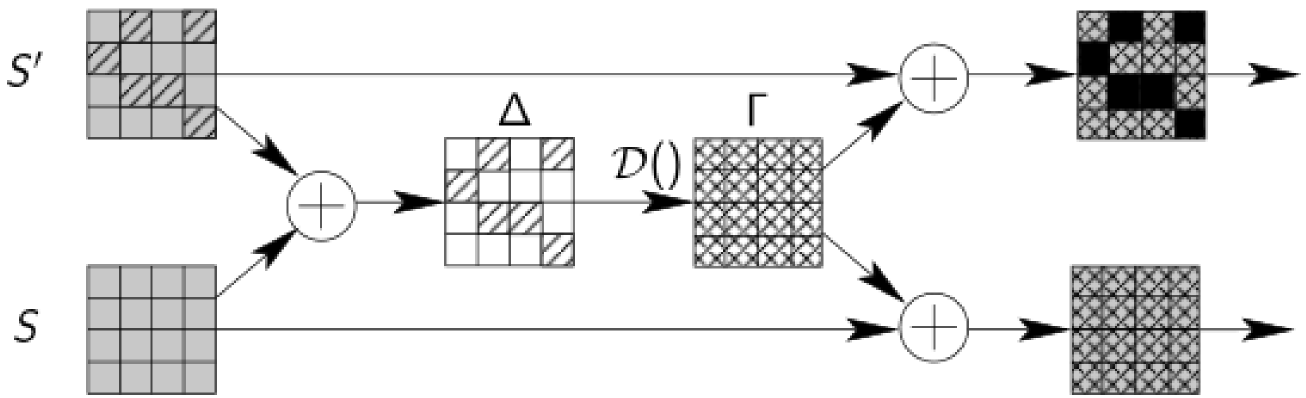
Infective Countermeasures

The main initial idea behind infective countermeasures was to diffuse the impact of the fault such that even if the adversary were to attack the comparison step, the state would still be affected

The Infection Mechanism

Generic sketch exhibiting the Infection CM:

- S, S' the two States
- $\mathcal{D}$ the *diffusion function* (such as $\mathcal{D}(0) = 0$)



Source : Lomne et. al. , On the Need of Randomness in Fault attack Countermeasures – Application to AES, FDTC 2012

Infective Countermeasures : State of the Art

Prior to 2012

- Fournier et. al. and Joye et. al. suggested infective countermeasure schemes using deterministic diffusion functions
- Used consistency checks between cipher and redundant computations
- Proved to be inherently insecure by Lomne et. al. in FDTC 2012

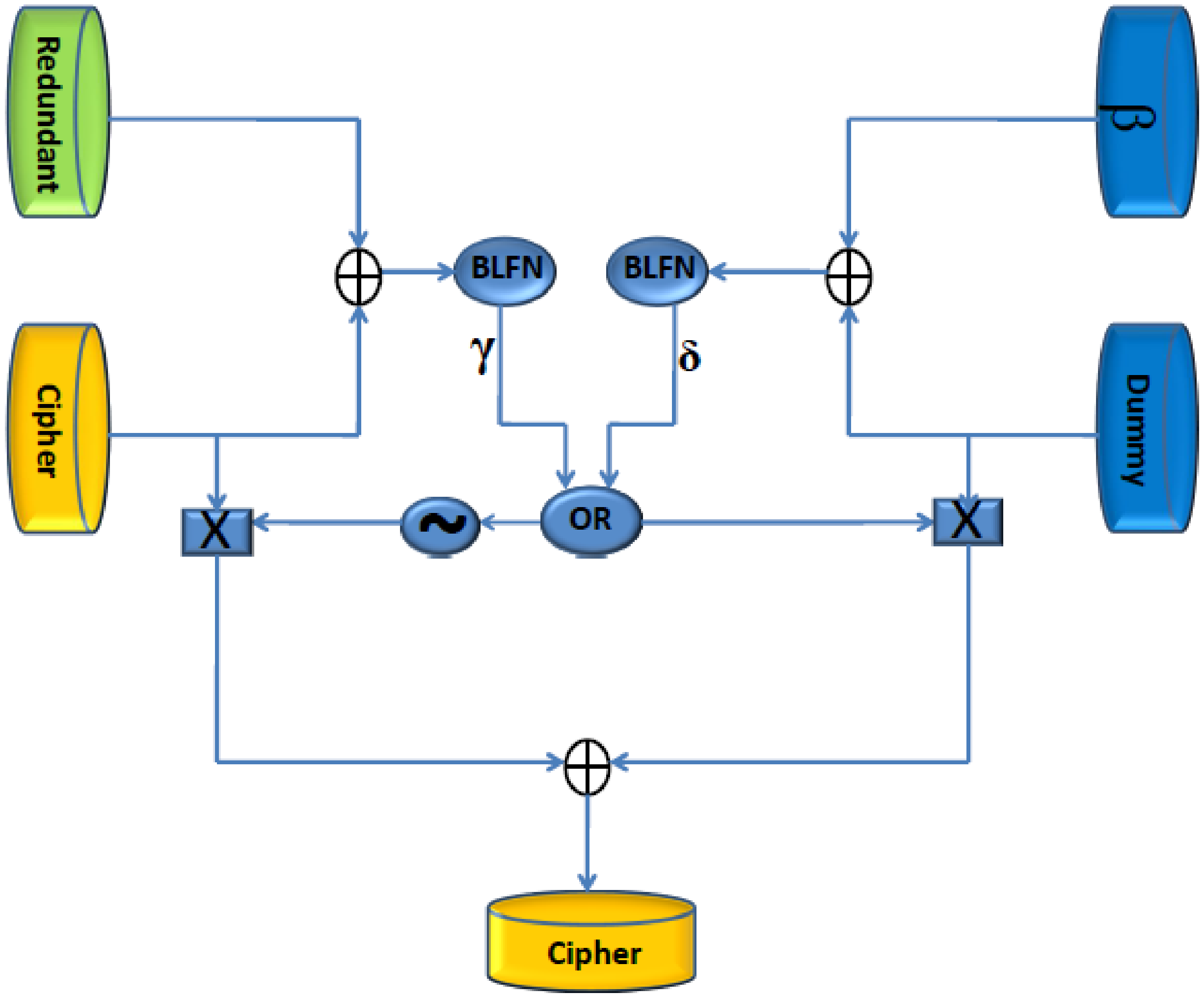
2012-2014

- Gierlichs et. al. proposed in LatinCrypt 2012 a randomized infective countermeasure that totally does away with explicit consistency checks by clever use of random and dummy rounds
- Propagation of faults prevents an attacker from being able to conduct any fault analysis of corrupted ciphertexts
- Proved to be insecure against attacks *on the last round* by Battistello et. al. in FDTC 2013 and Tupsamudre et. al. in CHES 2014

Since 2014

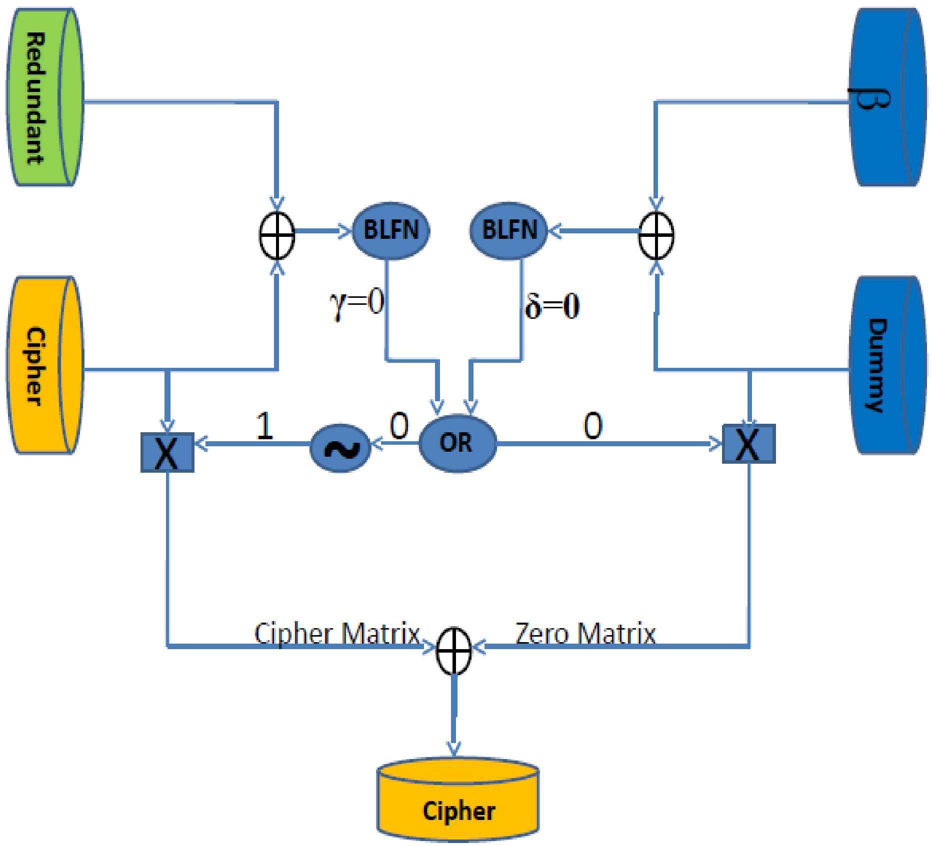
- Tupsamudre et. al. proposed a randomized infective countermeasure in CHES 2014
- Addresses several pitfalls of the earlier infective countermeasure scheme
- Does not provide any formal proofs of security
- Does not consider attacks where the execution order of instructions could be changed

CHES 2014 Infective Countermeasure



CHES 2014 Countermeasure (Contd.)

Correct Computation



Faulty Computation