DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

DFA on LS-Designs with a Practical Implementation on SCREAM

Works presentation at COSADE 2017

Benjamin Lac[1,5], Anne Canteaut[2], Jacques J.A. Fournier[3], Renaud Sirdey[4]

1 CEA-Tech, Gardanne, France,
2 Inria, Paris, France,
3 CEA-Leti, Grenoble, France,
4 CEA-List, Saclay, France
5 ENSM-SE, Saint-Étienne, France,
{benjamin.lac, jacques.fournier, renaud.sirdey}@cea.fr,
anne.canteaut@inria.fr

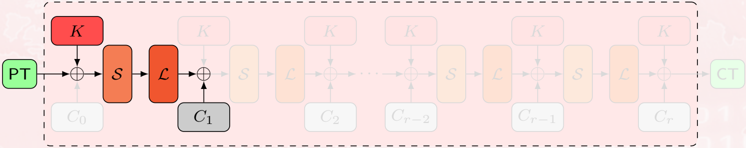
April 14th, 2017

1. LS-Designs
2. Applying DFA on LS-Designs
   - General principle
   - Depending on the fault model
3. Practical implementation of the DFA on SCREAM
   - The TAE mode SCREAM
   - DFA on SCREAM
   - Practical implementation
4. Countermeasures
   - Modes of operation
   - Masking
   - Internal Redundancy Countermeasure
5. Conclusion and perspectives

1. LS-Designs
2. Applying DFA on LS-Designs
   - General principle
   - Depending on the fault model
3. Practical implementation of the DFA on SCREAM
   - The TAE mode SCREAM
   - DFA on SCREAM
   - Practical implementation
4. Countermeasures
   - Modes of operation
   - Masking
   - Internal Redundancy Countermeasure
5. Conclusion and perspectives

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

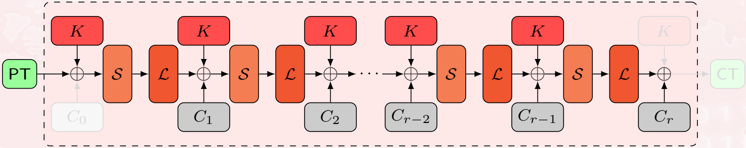


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

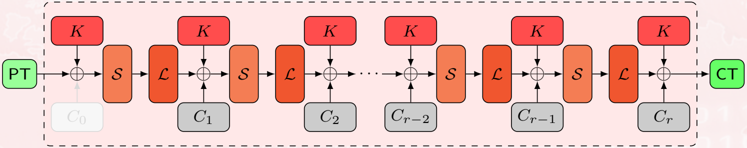


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

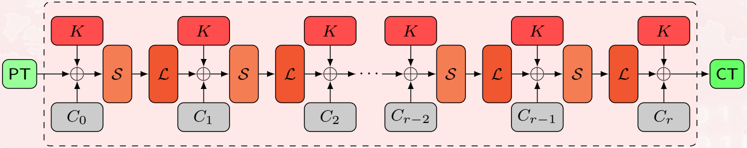


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

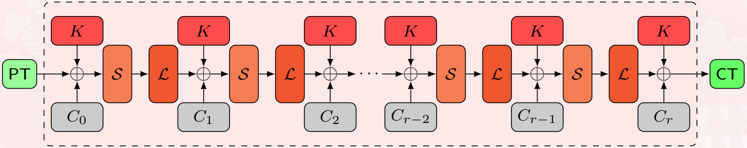


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

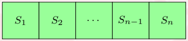
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



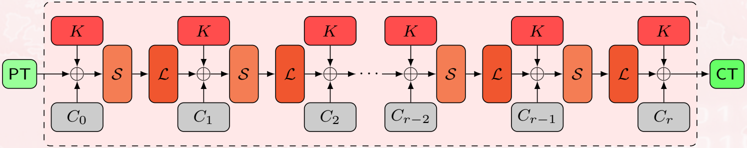
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



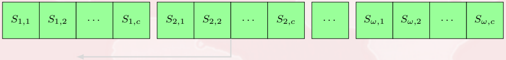
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



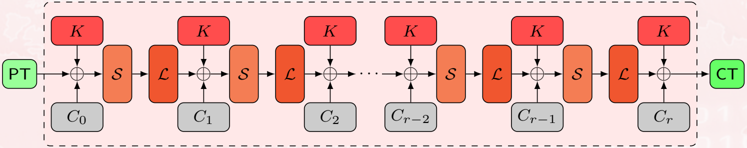
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



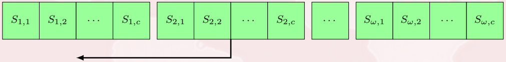
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



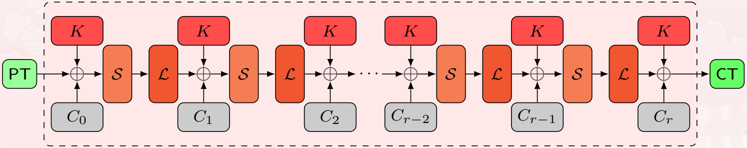
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

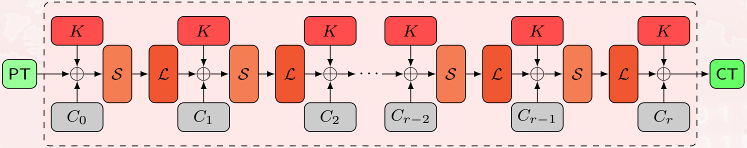


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

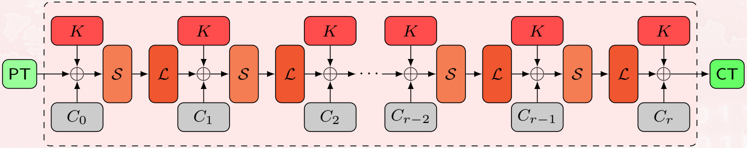


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

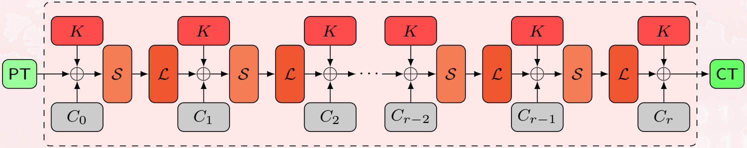


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.

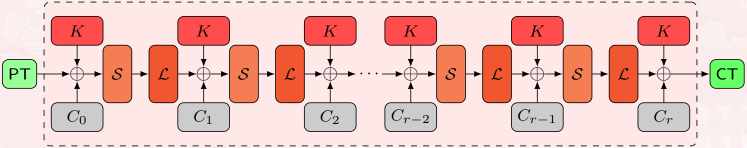


The round function

The inner state is represented as an $n = \omega \times c$-bit array.

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



The round function

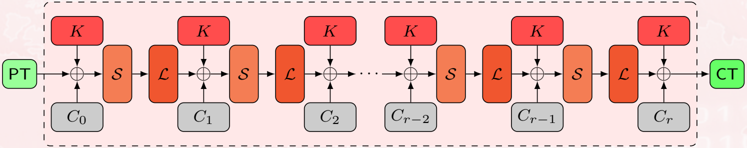
The inner state is represented as an $n = \omega \times c$-bit array.

$s_{1,1}$	$s_{1,2}$	$\cdots$	$s_{1,c}$
$s_{2,1}$	$s_{2,2}$	$\cdots$	$s_{2,c}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$s_{\omega,1}$	$s_{\omega,2}$	$\cdots$	$s_{\omega,c}$

LS-Designs

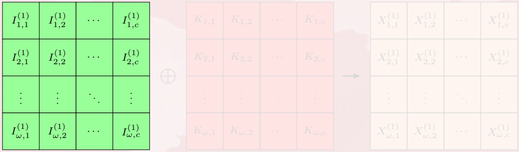
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



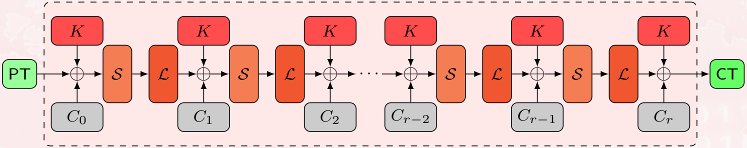
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



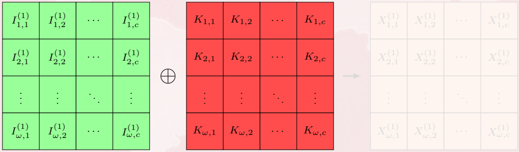
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



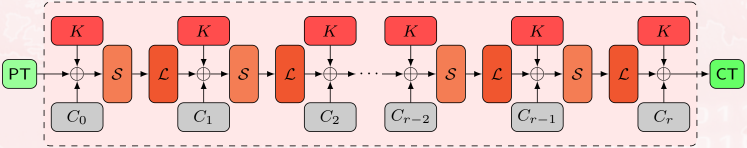
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



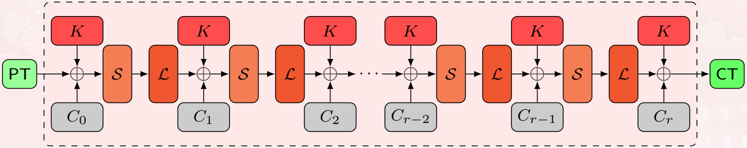
The round function

The inner state is represented as an $n = \omega \times c$-bit array.

$$\begin{pmatrix} I^{(1)}_{1,1} & I^{(1)}_{1,2} & \cdots & I^{(1)}_{1,c} \\ I^{(1)}_{2,1} & I^{(1)}_{2,2} & \cdots & I^{(1)}_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ I^{(1)}_{\omega,1} & I^{(1)}_{\omega,2} & \cdots & I^{(1)}_{\omega,c} \end{pmatrix} \oplus \begin{pmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,c} \\ K_{2,1} & K_{2,2} & \cdots & K_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ K_{\omega,1} & K_{\omega,2} & \cdots & K_{\omega,c} \end{pmatrix} \rightarrow \begin{pmatrix} X^{(1)}_{1,1} & X^{(1)}_{1,2} & \cdots & X^{(1)}_{1,c} \\ X^{(1)}_{2,1} & X^{(1)}_{2,2} & \cdots & X^{(1)}_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ X^{(1)}_{\omega,1} & X^{(1)}_{\omega,2} & \cdots & X^{(1)}_{\omega,c} \end{pmatrix}$$

General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



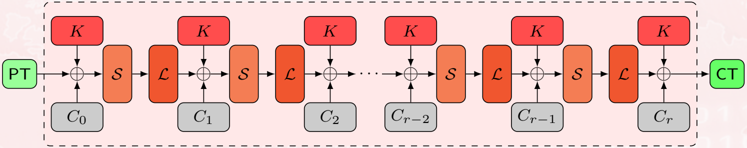
The round function

The inner state is represented as an $n = \omega \times c$-bit array.

$X^{(1)}_{1,1}$	$X^{(1)}_{1,2}$	$\cdots$	$X^{(1)}_{1,c}$
$X^{(1)}_{2,1}$	$X^{(1)}_{2,2}$	$\cdots$	$X^{(1)}_{2,c}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$X^{(1)}_{\omega,1}$	$X^{(1)}_{\omega,2}$	$\cdots$	$X^{(1)}_{\omega,c}$

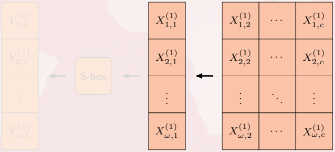
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



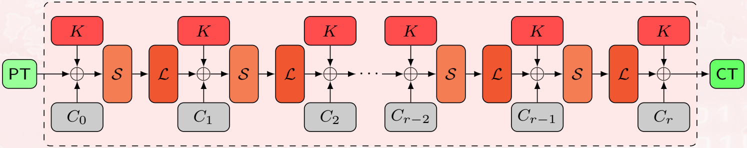
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



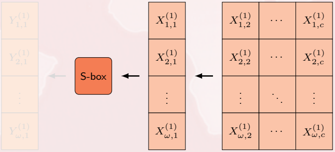
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



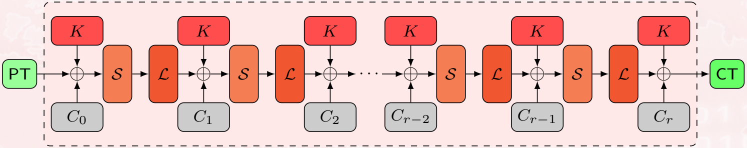
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



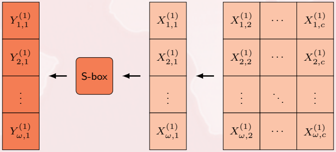
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



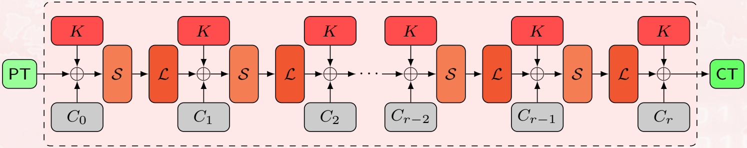
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



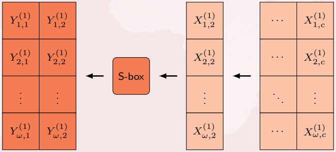
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



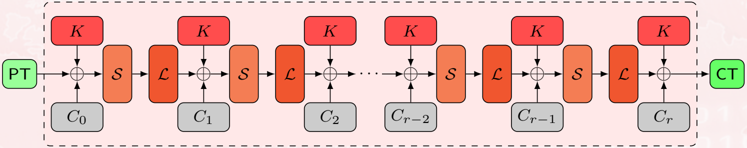
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



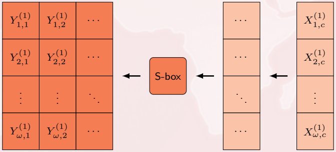
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



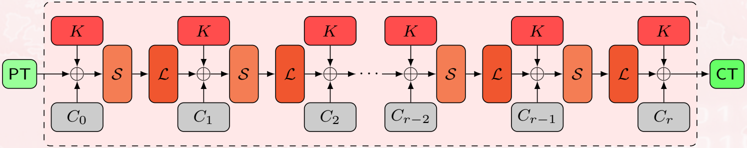
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



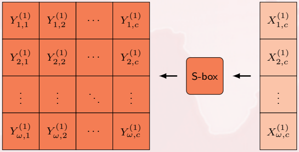
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



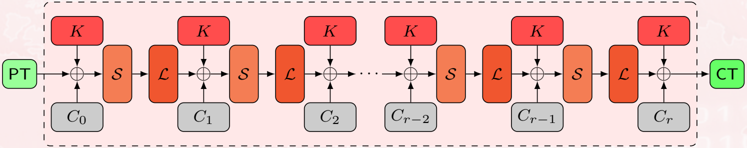
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



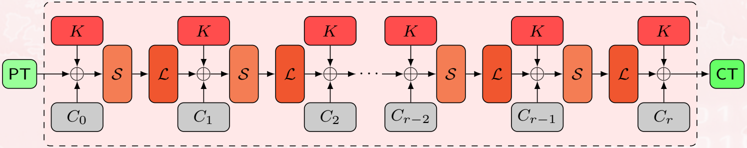
The round function

The inner state is represented as an $n = \omega \times c$-bit array.

$Y^{(1)}_{1,1}$	$Y^{(1)}_{1,2}$	$\cdots$	$Y^{(1)}_{1,c}$
$Y^{(1)}_{2,1}$	$Y^{(1)}_{2,2}$	$\cdots$	$Y^{(1)}_{2,c}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Y^{(1)}_{\omega,1}$	$Y^{(1)}_{\omega,2}$	$\cdots$	$Y^{(1)}_{\omega,c}$

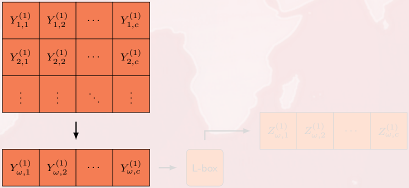
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



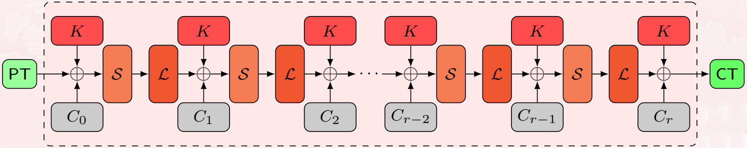
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



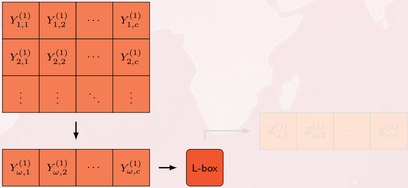
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



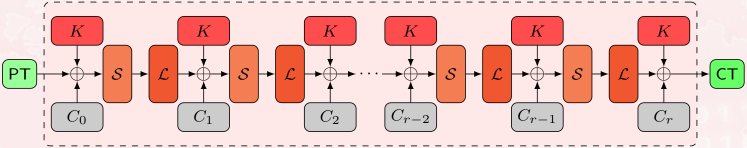
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



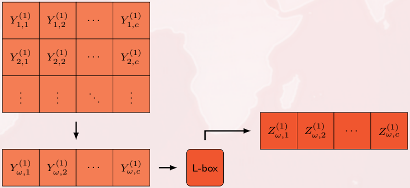
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



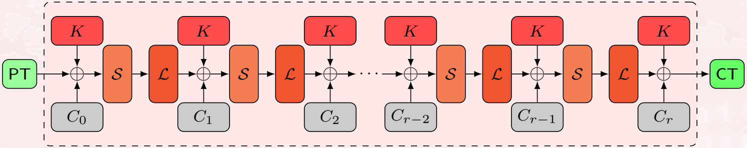
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



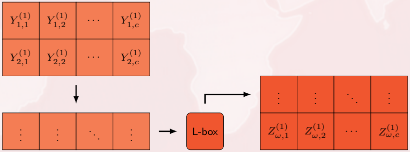
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



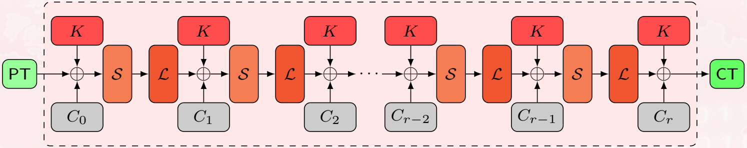
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



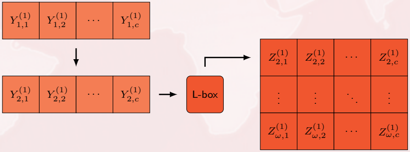
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



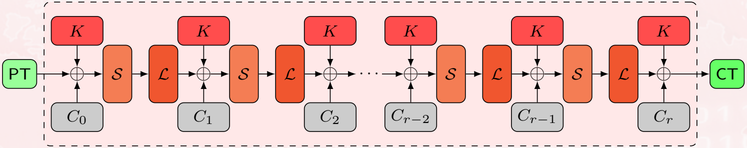
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



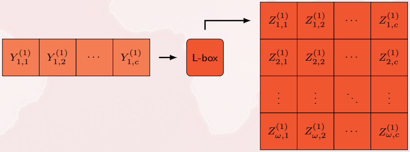
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



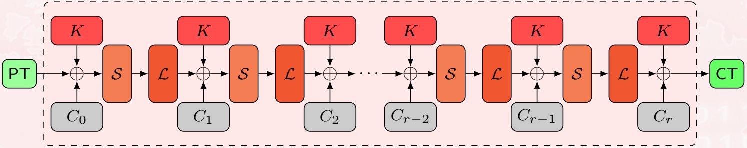
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



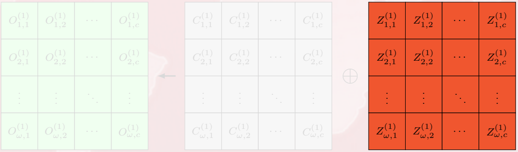
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



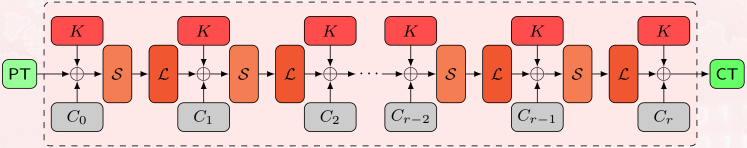
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



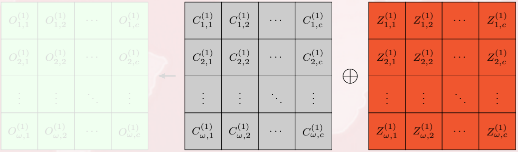
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



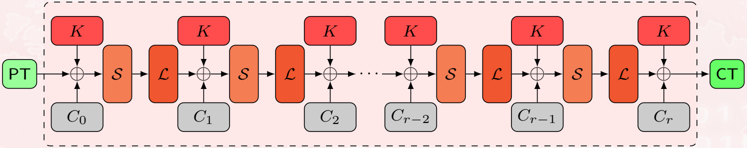
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



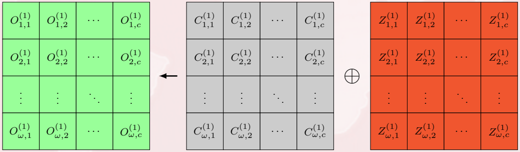
General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



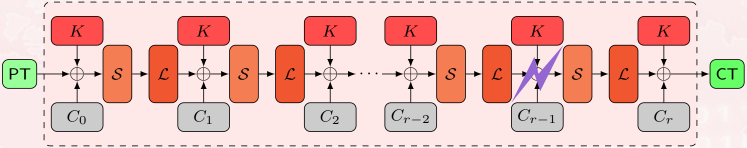
The round function

The inner state is represented as an $n = \omega \times c$-bit array.



General structure

An LS-Design is an iterative block cipher composed of r rounds, introduced by Grosso in 2014. It takes as input an n-bit block, uses an n-bit key and n-bit round constants.



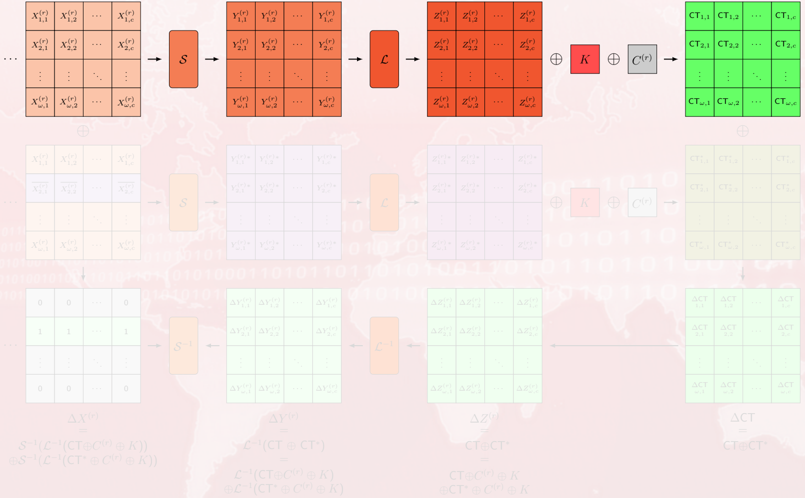
The round function

The inner state is represented as an $n = \omega \times c$-bit array.

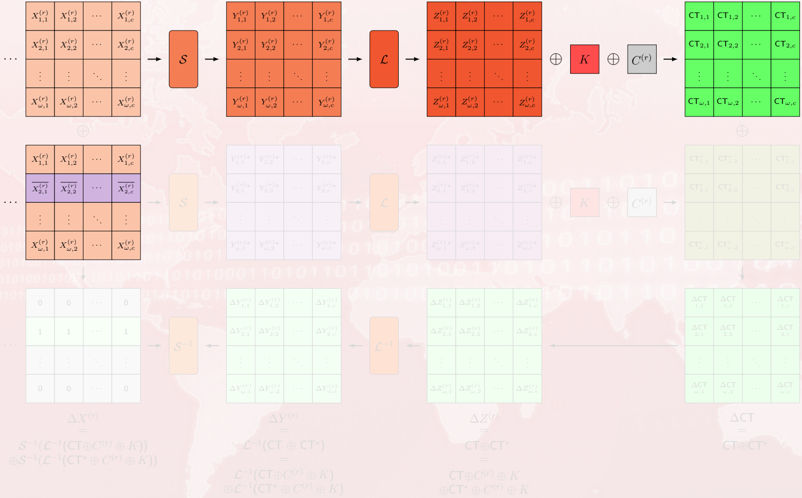
$$\begin{bmatrix} O^{(1)}_{1,1} & O^{(1)}_{1,2} & \cdots & O^{(1)}_{1,c} \\ O^{(1)}_{2,1} & O^{(1)}_{2,2} & \cdots & O^{(1)}_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ O^{(1)}_{\omega,1} & O^{(1)}_{\omega,2} & \cdots & O^{(1)}_{\omega,c} \end{bmatrix} \leftarrow \begin{bmatrix} C^{(1)}_{1,1} & C^{(1)}_{1,2} & \cdots & C^{(1)}_{1,c} \\ C^{(1)}_{2,1} & C^{(1)}_{2,2} & \cdots & C^{(1)}_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ C^{(1)}_{\omega,1} & C^{(1)}_{\omega,2} & \cdots & C^{(1)}_{\omega,c} \end{bmatrix} \oplus \begin{bmatrix} Z^{(1)}_{1,1} & Z^{(1)}_{1,2} & \cdots & Z^{(1)}_{1,c} \\ Z^{(1)}_{2,1} & Z^{(1)}_{2,2} & \cdots & Z^{(1)}_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ Z^{(1)}_{\omega,1} & Z^{(1)}_{\omega,2} & \cdots & Z^{(1)}_{\omega,c} \end{bmatrix}$$

1. LS-Designs
2. Applying DFA on LS-Designs
   - General principle
   - Depending on the fault model
3. Practical implementation of the DFA on SCREAM
   - The TAE mode SCREAM
   - DFA on SCREAM
   - Practical implementation
4. Countermeasures
   - Modes of operation
   - Masking
   - Internal Redundancy Countermeasure
5. Conclusion and perspectives

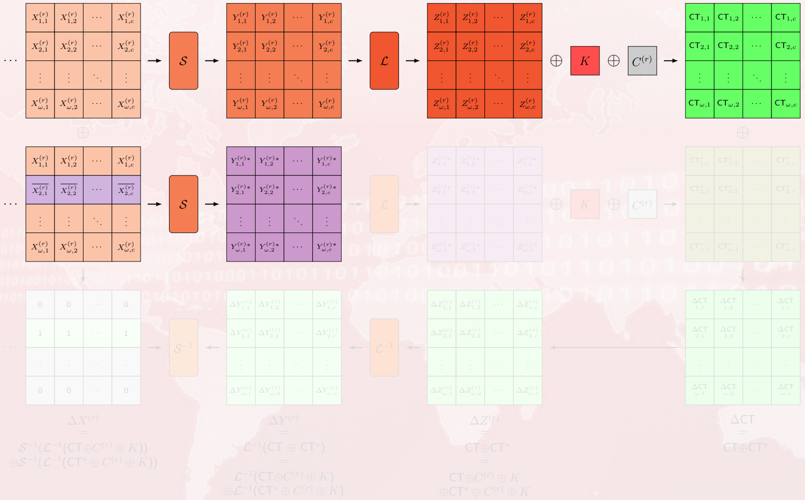
General principle



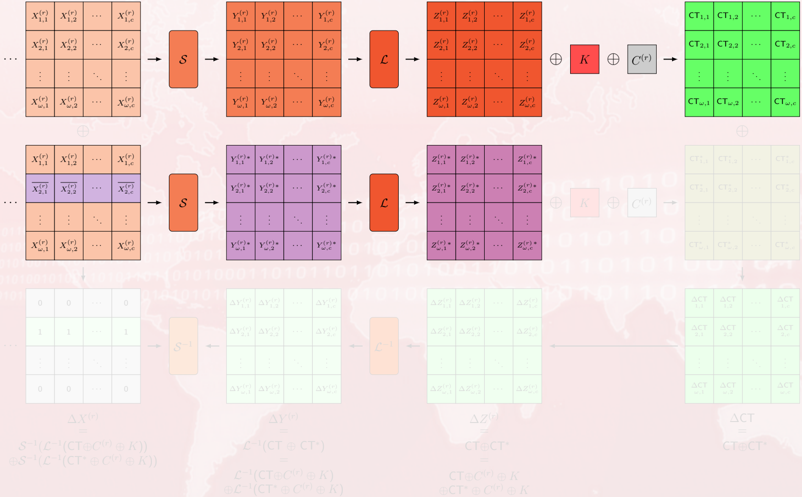
General principle



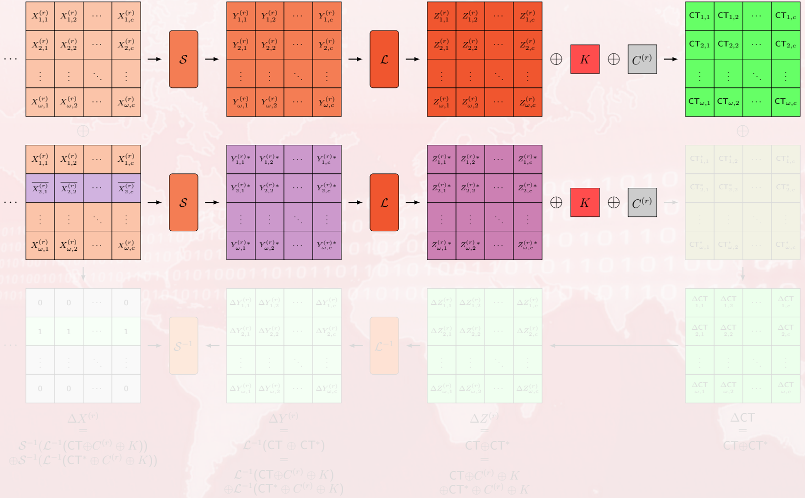
General principle



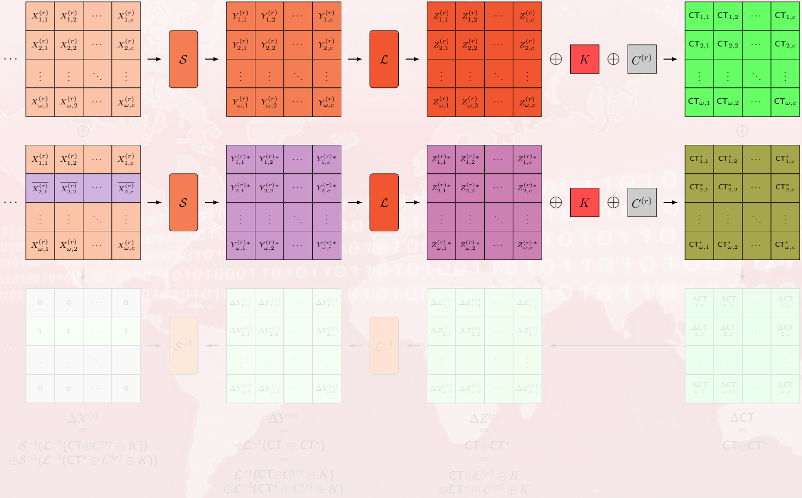
General principle



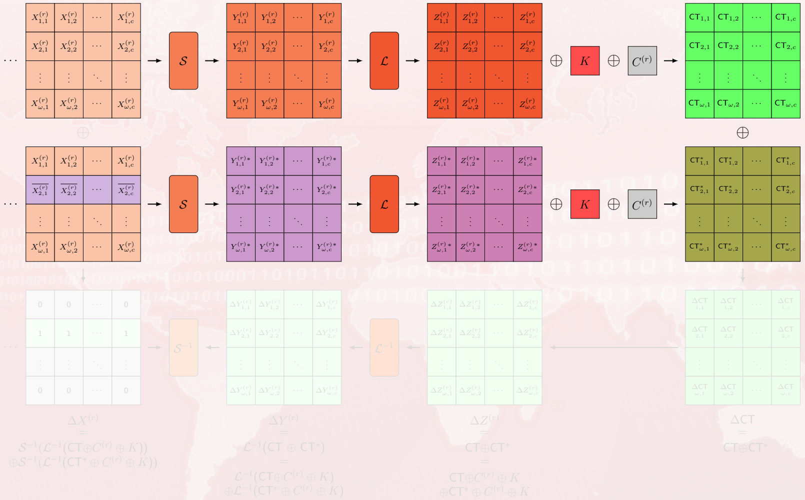
General principle



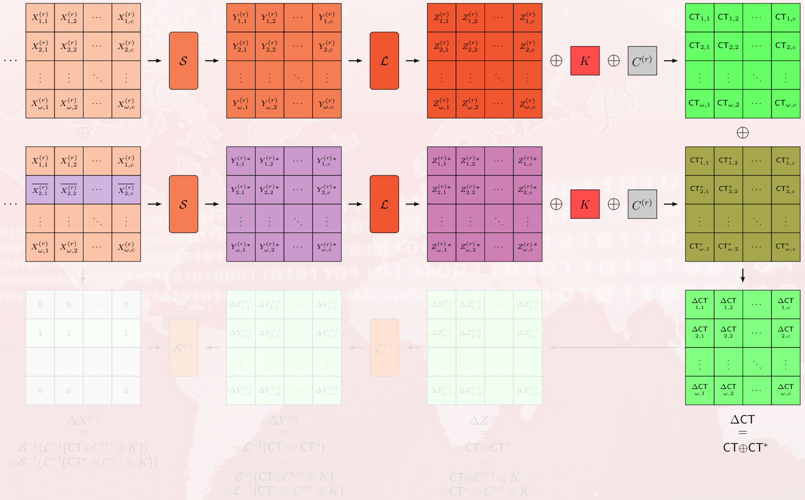
General principle



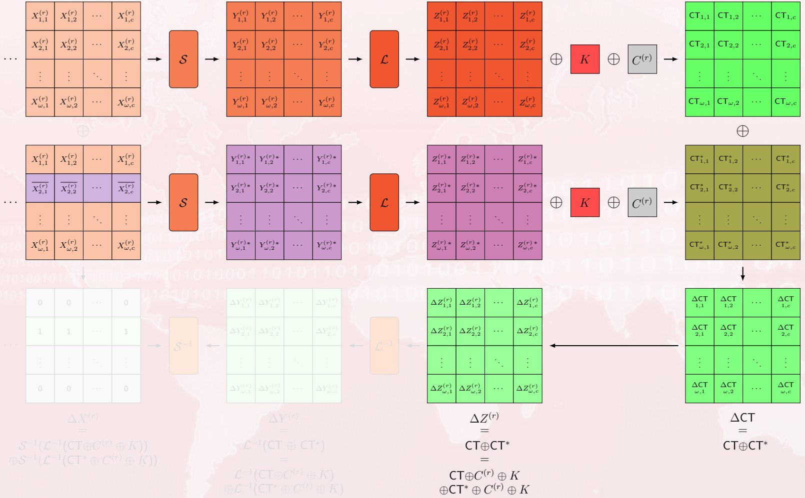
General principle



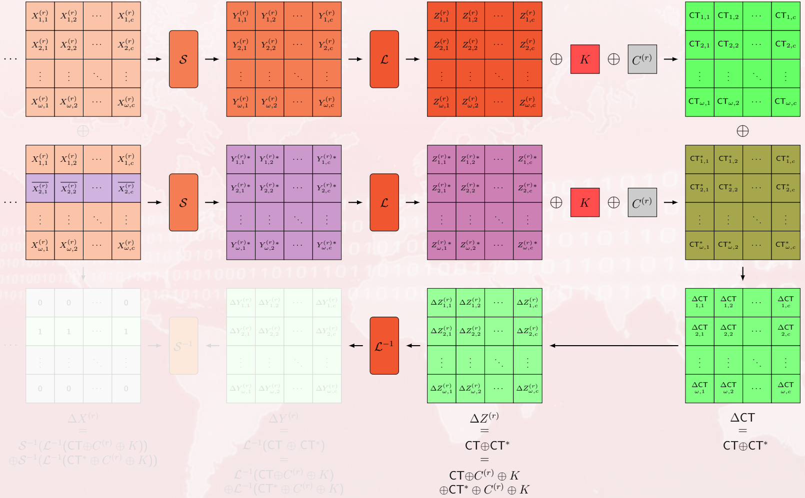
General principle



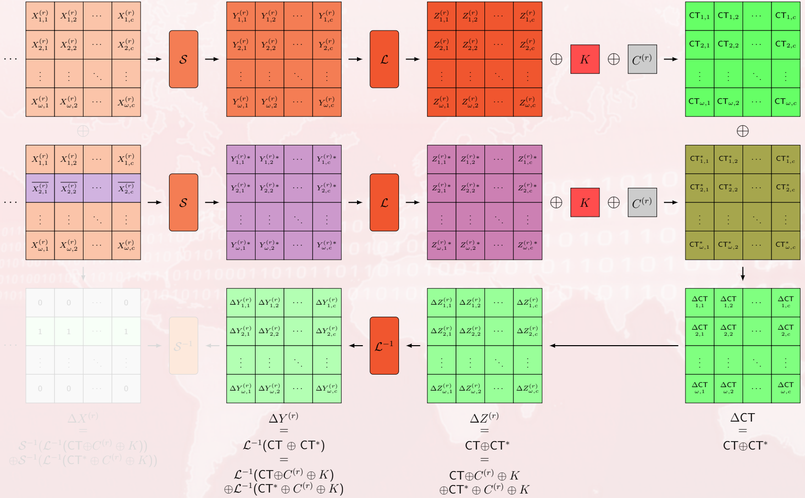
General principle



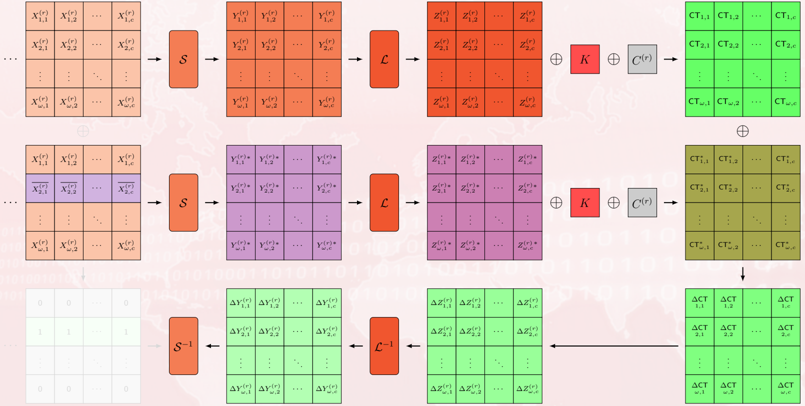
General principle



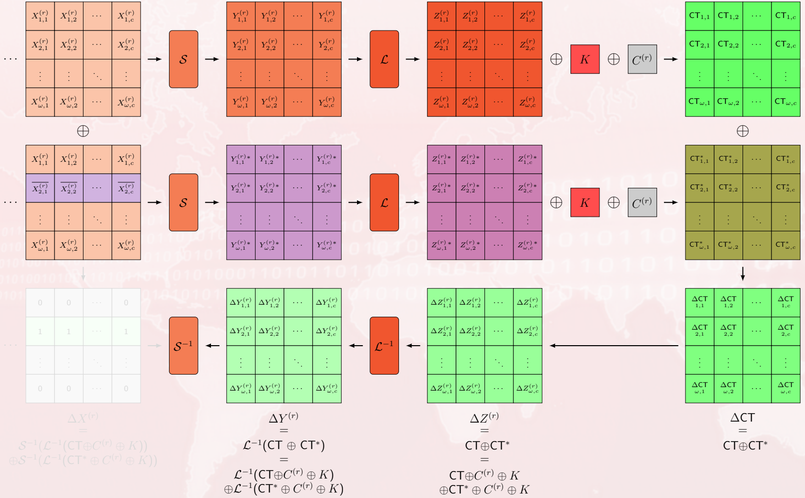
General principle



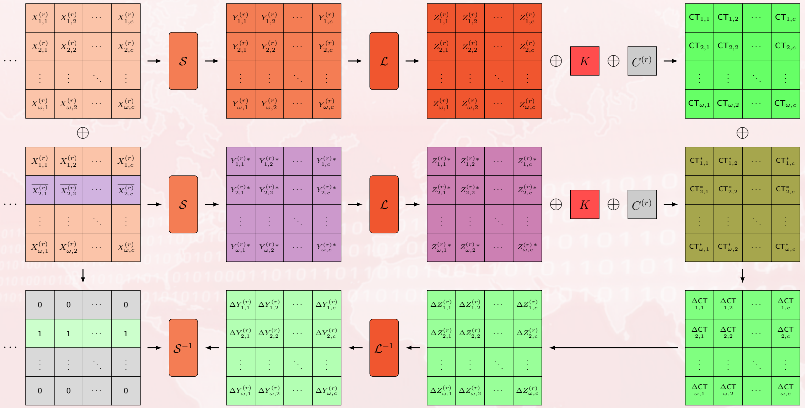
General principle



General principle



General principle



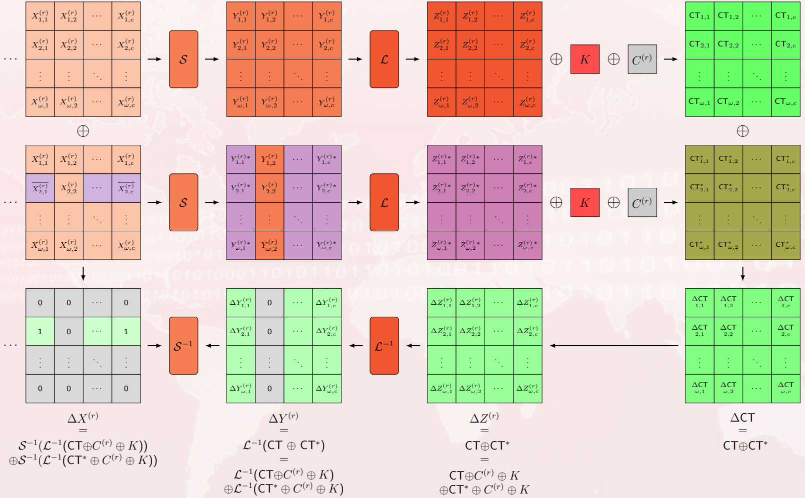
$\Delta X^{(r)} = \mathcal{S}^{-1}(\mathcal{L}^{-1}(\mathsf{CT} \oplus C^{(r)} \oplus K)) \oplus \mathcal{S}^{-1}(\mathcal{L}^{-1}(\mathsf{CT}^* \oplus C^{(r)} \oplus K))$

$\Delta Y^{(r)} = \mathcal{L}^{-1}(\mathsf{CT} \oplus \mathsf{CT}^*) = \mathcal{L}^{-1}(\mathsf{CT} \oplus C^{(r)} \oplus K) \oplus \mathcal{L}^{-1}(\mathsf{CT}^* \oplus C^{(r)} \oplus K)$

$\Delta Z^{(r)} = \mathsf{CT} \oplus \mathsf{CT}^* = \mathsf{CT} \oplus C^{(r)} \oplus K \oplus \mathsf{CT}^* \oplus C^{(r)} \oplus K$

$\Delta \mathsf{CT} = \mathsf{CT} \oplus \mathsf{CT}^*$

General principle



Proposition

Let $\mathcal{S}$ be an n-bit S-box. Let (a_1, b_1) and (a_2, b_2) be two differentials with $a_1 \neq a_2$ such that the system of two equations

$$\mathcal{S}(x \oplus a_1) \oplus \mathcal{S}(x) = b_1 \tag{1}$$

$$\mathcal{S}(x \oplus a_2) \oplus \mathcal{S}(x) = b_2 \tag{2}$$

has at least two solutions. Then, each of the three equations (1), (2) and

$$\mathcal{S}(x \oplus a_1 \oplus a_2) \oplus \mathcal{S}(x) = b_1 \oplus b_2 \tag{3}$$

has at least four solutions.

Mathematical exploited relations

Obtaining information on the key is possible from each ω-bit word $1 \leq i \leq c$:

$x = \mathcal{L}^{-1}(\mathrm{CT} \oplus C^{(r)} \oplus K)[i]$ and $y = \mathcal{L}^{-1}(\mathrm{CT}^* \oplus C^{(r)} \oplus K)[i]$

satisfy

$$x \oplus y = \mathcal{L}^{-1}(\mathrm{CT} \oplus \mathrm{CT}^*) = \Delta Y^{(r)}[i] = a_1$$

and

$$\mathcal{S}^{-1}(x) \oplus \mathcal{S}^{-1}(y) = \Delta X^{(r)}[i] = 2^{\omega - j} = b_1.$$

Proposition

Let $\mathcal{S}$ be an n-bit S-box. Let (a_1, b_1) and (a_2, b_2) be two differentials with $a_1 \neq a_2$ such that the system of two equations

$$\mathcal{S}(x \oplus a_1) \oplus \mathcal{S}(x) = b_1 \quad (1)$$

$$\mathcal{S}(x \oplus a_2) \oplus \mathcal{S}(x) = b_2 \quad (2)$$

has at least two solutions. Then, each of the three equations (1), (2) and

$$\mathcal{S}(x \oplus a_1 \oplus a_2) \oplus \mathcal{S}(x) = b_1 \oplus b_2 \quad (3)$$

has at least four solutions.

Mathematical exploited relations

Obtaining information on the key is possible from each ω-bit word $1 \leq i \leq c$:

$x = \mathcal{L}^{-1}(\mathsf{CT} \oplus C^{(r)} \oplus K)[i]$ and $y = \mathcal{L}^{-1}(\mathsf{CT}^* \oplus C^{(r)} \oplus K)[i]$

satisfy

$$x \oplus y = \mathcal{L}^{-1}(\mathsf{CT} \oplus \mathsf{CT}^*) = \Delta Y^{(r)}[i] = a_1$$

and

$$\mathcal{S}^{-1}(x) \oplus \mathcal{S}^{-1}(y) = \Delta X^{(r)}[i] = 2^{\omega - j} = b_1.$$

Proposition

Let $\mathcal{S}$ be an n-bit S-box. Let (a_1, b_1) and (a_2, b_2) be two differentials with $a_1 \neq a_2$ such that the system of two equations

$$\mathcal{S}(x \oplus a_1) \oplus \mathcal{S}(x) = b_1 \tag{1}$$

$$\mathcal{S}(x \oplus a_2) \oplus \mathcal{S}(x) = b_2 \tag{2}$$

has at least two solutions. Then, each of the three equations (1), (2) and

$$\mathcal{S}(x \oplus a_1 \oplus a_2) \oplus \mathcal{S}(x) = b_1 \oplus b_2 \tag{3}$$

has at least four solutions.

Mathematical exploited relations

Obtaining information on the key is possible from each ω-bit word $1 \leq i \leq c$:

$$x = \mathcal{L}^{-1}(\mathrm{CT} \oplus C^{(r)} \oplus K)[i] \text{ and } y = \mathcal{L}^{-1}(\mathrm{CT}^* \oplus C^{(r)} \oplus K)[i]$$

satisfy

$$x \oplus y = \mathcal{L}^{-1}(\mathrm{CT} \oplus \mathrm{CT}^*) = \Delta Y^{(r)}[i] = a_1$$

and

$$\mathcal{S}^{-1}(x) \oplus \mathcal{S}^{-1}(y) = \Delta X^{(r)}[i] = 2^{\omega - j} = b_1.$$

Proposition

Let $\mathcal{S}$ be an n-bit S-box. Let (a_1, b_1) and (a_2, b_2) be two differentials with $a_1 \neq a_2$ such that the system of two equations

$$\mathcal{S}(x \oplus a_1) \oplus \mathcal{S}(x) = b_1 \quad (1)$$

$$\mathcal{S}(x \oplus a_2) \oplus \mathcal{S}(x) = b_2 \quad (2)$$

has at least two solutions. Then, each of the three equations (1), (2) and

$$\mathcal{S}(x \oplus a_1 \oplus a_2) \oplus \mathcal{S}(x) = b_1 \oplus b_2 \quad (3)$$

has at least four solutions.

Mathematical exploited relations

Obtaining information on the key is possible from each ω-bit word $1 \leq i \leq c$:

$x = \mathcal{L}^{-1}(\text{CT} \oplus C^{(r)} \oplus K)[i]$ and $y = \mathcal{L}^{-1}(\text{CT}^* \oplus C^{(r)} \oplus K)[i]$

satisfy

$$x \oplus y = \mathcal{L}^{-1}(\text{CT} \oplus \text{CT}^*) = \Delta Y^{(r)}[i] = a_1$$

and

$$\mathcal{S}^{-1}(x) \oplus \mathcal{S}^{-1}(y) = \Delta X^{(r)}[i] = 2^{\omega - j} = b_1.$$

Ideal fault model

- Find $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$ with $1 \leq i_1 < i_2 \leq \omega$ such that (a_1, b_1) and (a_2, b_2) are simultaneously satisfied for a single element for all a_1 and a_2.

Flip the row i_1 then the row i_2 of the state before the last substitution layer with two successive fault injections in order to retrieve the complete secret key.

Exploitable differential pairs

Cipher	Pair
PRIDE	$(a_1$,0x1), $(a_2$,0x8)
Robin	$(a_1$,0x01), $(a_2$,0x40)
Fantomas	$(a_1$,0x01), $(a_2$,0x80)
Scream	$(a_1$,0x01), $(a_2$,0x02)
iScream	$(a_1$,0x02), $(a_2$,0x80)

Ideal fault model

- Find $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$ with $1 \leq i_1 < i_2 \leq \omega$ such that (a_1, b_1) and (a_2, b_2) are simultaneously satisfied for a single element for all a_1 and a_2.
- Flip the row i_1 then the row i_2 of the state before the last substitution layer with two successive fault injections in order to retrieve the complete secret key.

Exploitable differential pairs

Cipher	Pair
PRIDE	$(a_1$,0x1), $(a_2$,0x8)
Robin	$(a_1$,0x01), $(a_2$,0x40)
Fantomas	$(a_1$,0x01), $(a_2$,0x80)
Scream	$(a_1$,0x01), $(a_2$,0x02)
iScream	$(a_1$,0x02), $(a_2$,0x80)

Ideal fault model

- Find $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$ with $1 \leq i_1 < i_2 \leq \omega$ such that (a_1, b_1) and (a_2, b_2) are simultaneously satisfied for a single element for all a_1 and a_2.
- Flip the row i_1 then the row i_2 of the state before the last substitution layer with two successive fault injections in order to retrieve the complete secret key.

Exploitable differential pairs

Cipher	Pair
PRIDE	(a_1,0x1), (a_2,0x8)
Robin	(a_1,0x01), (a_2,0x40)
Fantomas	(a_1,0x01), (a_2,0x80)
Scream	(a_1,0x01), (a_2,0x02)
iScream	(a_1,0x02), (a_2,0x80)

- **Random fault model**
  - Target the same $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$.

Let A_1 (resp. A_2) be the average number of remaining candidates for a ω-bit key word obtained from a fault on row i_1 (resp. i_2).

Let m_1 (resp. m_2) denote the number of obtained faults on row i_1 (resp. i_2). Then, the number N of remaining candidates for the key is:

$$\left(\frac{2^\omega}{2^{m_1+m_2}} + \sum_{i=1}^{m_1} \frac{A_1}{2^{i+m_2}} + \sum_{i=1}^{m_2} \frac{A_2}{2^{i+m_1}} + \left(\sum_{i=1}^{m_1} \frac{1}{2^i} \right) \left(\sum_{i=1}^{m_2} \frac{1}{2^i} \right) \right)^c$$

Indeed, from m faults, an attacker obtains no difference on a ω-bit word with probability $1/2^m$, she obtains at least one difference with probability $\sum_{i=1}^{m} 1/2^i = (2^m - 1)/2^m$. We then deduce that N is equal to:

$$\left(\frac{2^\omega + A_1(2^{m_1} - 1) + A_2(2^{m_2} - 1) + (2^{m_1} - 1)(2^{m_2} - 1)}{2^{m_1+m_2}} \right)^c$$

- **Random fault model**
  - Target the same $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$.
  - Let A_1 (resp. A_2) be the average number of remaining candidates for a ω-bit key word obtained from a fault on row i_1 (resp. i_2).

  Let m_1 (resp. m_2) denote the number of obtained faults on row i_1 (resp. i_2).

Then, the number N of remaining candidates for the key is:

$$\left(\frac{2^\omega}{2^{m_1+m_2}} + \sum_{i=1}^{m_1} \frac{A_1}{2^{i+m_2}} + \sum_{i=1}^{m_2} \frac{A_2}{2^{i+m_1}} + \left(\sum_{i=1}^{m_1} \frac{1}{2^i}\right)\left(\sum_{i=1}^{m_2} \frac{1}{2^i}\right)\right)^c$$

Indeed, from m faults, an attacker obtains no difference on a ω-bit word with probability $1/2^m$, she obtains at least one difference with probability $\sum_{i=1}^{m} 1/2^i = (2^m - 1)/2^m$. We then deduce that N is equal to:

$$\left(\frac{2^\omega + A_1(2^{m_1} - 1) + A_2(2^{m_2} - 1) + (2^{m_1} - 1)(2^{m_2} - 1)}{2^{m_1+m_2}}\right)^c$$

- **Random fault model**
  - Target the same $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$.
  - Let A_1 (resp. A_2) be the average number of remaining candidates for a ω-bit key word obtained from a fault on row i_1 (resp. i_2).
  - Let m_1 (resp. m_2) denote the number of obtained faults on row i_1 (resp. i_2).

Then, the number N of remaining candidates for the key is:

$$\left(\frac{2^\omega}{2^{m_1+m_2}} + \sum_{i=1}^{m_1}\frac{A_1}{2^{i+m_2}} + \sum_{i=1}^{m_2}\frac{A_2}{2^{i+m_1}} + \left(\sum_{i=1}^{m_1}\frac{1}{2^i}\right)\left(\sum_{i=1}^{m_2}\frac{1}{2^i}\right)\right)^c$$

Indeed, from m faults, an attacker obtains no difference on a ω-bit word with probability $1/2^m$, she obtains at least one difference with probability $\sum_{i=1}^{m} 1/2^i = (2^m - 1)/2^m$. We then deduce that N is equal to:

$$\left(\frac{2^\omega + A_1(2^{m_1}-1) + A_2(2^{m_2}-1) + (2^{m_1}-1)(2^{m_2}-1)}{2^{m_1+m_2}}\right)^c$$

- **Random fault model**
  - Target the same $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$.
  - Let A_1 (resp. A_2) be the average number of remaining candidates for a ω-bit key word obtained from a fault on row i_1 (resp. i_2).
  - Let m_1 (resp. m_2) denote the number of obtained faults on row i_1 (resp. i_2).

Then, the number N of remaining candidates for the key is:

$$\left(\frac{2^\omega}{2^{m_1+m_2}} + \sum_{i=1}^{m_1} \frac{A_1}{2^{i+m_2}} + \sum_{i=1}^{m_2} \frac{A_2}{2^{i+m_1}} + \left(\sum_{i=1}^{m_1} \frac{1}{2^i} \right) \left(\sum_{i=1}^{m_2} \frac{1}{2^i} \right) \right)^c$$

Indeed, from m faults, an attacker obtains no difference on a ω-bit word with probability $1/2^m$, she obtains at least one difference with probability $\sum_{i=1}^{m} 1/2^i = (2^m - 1)/2^m$. We then deduce that N is equal to:

$$\left(\frac{2^\omega + A_1(2^{m_1} - 1) + A_2(2^{m_2} - 1) + (2^{m_1} - 1)(2^{m_2} - 1)}{2^{m_1+m_2}} \right)^c$$

- **Random fault model**
  - Target the same $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$.
  - Let A_1 (resp. A_2) be the average number of remaining candidates for a ω-bit key word obtained from a fault on row i_1 (resp. i_2).
  - Let m_1 (resp. m_2) denote the number of obtained faults on row i_1 (resp. i_2).

Then, the number N of remaining candidates for the key is:

$$\left(\frac{2^\omega}{2^{m_1+m_2}} + \sum_{i=1}^{m_1} \frac{A_1}{2^{i+m_2}} + \sum_{i=1}^{m_2} \frac{A_2}{2^{i+m_1}} + \left(\sum_{i=1}^{m_1} \frac{1}{2^i} \right) \left(\sum_{i=1}^{m_2} \frac{1}{2^i} \right) \right)^c$$

Indeed, from m faults, an attacker obtains no difference on a ω-bit word with probability $1/2^m$, she obtains at least one difference with probability $\sum_{i=1}^{m} 1/2^i = (2^m - 1)/2^m$.
We then deduce that N is equal to:

$$\left(\frac{2^\omega + A_1(2^{m_1} - 1) + A_2(2^{m_2} - 1) + (2^{m_1} - 1)(2^{m_2} - 1)}{2^{m_1+m_2}} \right)^c$$

Random fault model

- Target the same $b_1 = \Delta X_1^{(r)} = 2^{\omega - i_1}$ and $b_2 = \Delta X_2^{(r)} = 2^{\omega - i_2}$.
- Let A_1 (resp. A_2) be the average number of remaining candidates for a ω-bit key word obtained from a fault on row i_1 (resp. i_2).
- Let m_1 (resp. m_2) denote the number of obtained faults on row i_1 (resp. i_2).

Then, the number N of remaining candidates for the key is:

$$\left(\frac{2^\omega}{2^{m_1+m_2}} + \sum_{i=1}^{m_1} \frac{A_1}{2^{i+m_2}} + \sum_{i=1}^{m_2} \frac{A_2}{2^{i+m_1}} + \left(\sum_{i=1}^{m_1} \frac{1}{2^i} \right) \left(\sum_{i=1}^{m_2} \frac{1}{2^i} \right) \right)^c$$

Indeed, from m faults, an attacker obtains no difference on a ω-bit word with probability $1/2^m$, she obtains at least one difference with probability $\sum_{i=1}^{m} 1/2^i = (2^m - 1)/2^m$. We then deduce that N is equal to:

$$\left(\frac{2^\omega + A_1(2^{m_1} - 1) + A_2(2^{m_2} - 1) + (2^{m_1} - 1)(2^{m_2} - 1)}{2^{m_1+m_2}} \right)^c$$

- **The design of the linear layer**
  - Flip a c-bit row of the state before the substitution layer activates all S-boxes at its input.
  - Use this property on the last substitution layer allows the attacker to recover information on all ω-bit words of K.
  - The number of remaining candidates is at most δ^c, where δ is the differential-uniformity of the S-box.
- **The differential properties of the S-box**
  - The number of inputs which satisfy two valid differentials simultaneously is usually reduced to a single element.
  - It is therefore sufficient to find two differences $2^{\omega-i}$ and $2^{\omega-j}$ which verify it and to flip the i-th row then the j-th row before the last substitution layer.

- **The design of the linear layer**
  - Flip a c-bit row of the state before the substitution layer activates all S-boxes at its input.
  - Use this property on the last substitution layer allows the attacker to recover information on all ω-bit words of K.
  - The number of remaining candidates is at most δ^c, where δ is the differential-uniformity of the S-box.
- **The differential properties of the S-box**
  - The number of inputs which satisfy two valid differentials simultaneously is usually reduced to a single element.
  - It is therefore sufficient to find two differences $2^{\omega-i}$ and $2^{\omega-j}$ which verify it and to flip the i-th row then the j-th row before the last substitution layer.

- **The design of the linear layer**
  - Flip a c-bit row of the state before the substitution layer activates all S-boxes at its input.
  - Use this property on the last substitution layer allows the attacker to recover information on all ω-bit words of K.
  - The number of remaining candidates is at most δ^c, where δ is the differential-uniformity of the S-box.
- **The differential properties of the S-box**
  - The number of inputs which satisfy two valid differentials simultaneously is usually reduced to a single element.
  - It is therefore sufficient to find two differences $2^{\omega-i}$ and $2^{\omega-j}$ which verify it and to flip the i-th row then the j-th row before the last substitution layer.

- **The design of the linear layer**
  - Flip a c-bit row of the state before the substitution layer activates all S-boxes at its input.
  - Use this property on the last substitution layer allows the attacker to recover information on all ω-bit words of K.
  - The number of remaining candidates is at most δ^c, where δ is the differential-uniformity of the S-box.
- **The differential properties of the S-box**
  - The number of inputs which satisfy two valid differentials simultaneously is usually reduced to a single element.

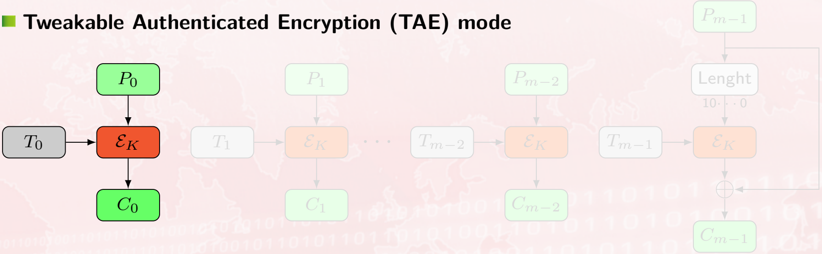
  It is therefore sufficient to find two differences $2^{\omega - i}$ and $2^{\omega - j}$ which verify it and to flip the i-th row then the j-th row before the last substitution layer.

- **The design of the linear layer**
  - Flip a c-bit row of the state before the substitution layer activates all S-boxes at its input.
  - Use this property on the last substitution layer allows the attacker to recover information on all ω-bit words of K.
  - The number of remaining candidates is at most δ^c, where δ is the differential-uniformity of the S-box.
- **The differential properties of the S-box**
  - The number of inputs which satisfy two valid differentials simultaneously is usually reduced to a single element.
  - It is therefore sufficient to find two differences $2^{\omega-i}$ and $2^{\omega-j}$ which verify it and to flip the i-th row then the j-th row before the last substitution layer.

Practical implementation of the DFA on SCREAM

1. LS-Designs
2. Applying DFA on LS-Designs
   - General principle
   - Depending on the fault model
3. Practical implementation of the DFA on SCREAM
   - The TAE mode SCREAM
   - DFA on SCREAM
   - Practical implementation
4. Countermeasures
   - Modes of operation
   - Masking
   - Internal Redundancy Countermeasure
5. Conclusion and perspectives

Tweakable Authenticated Encryption (TAE) mode

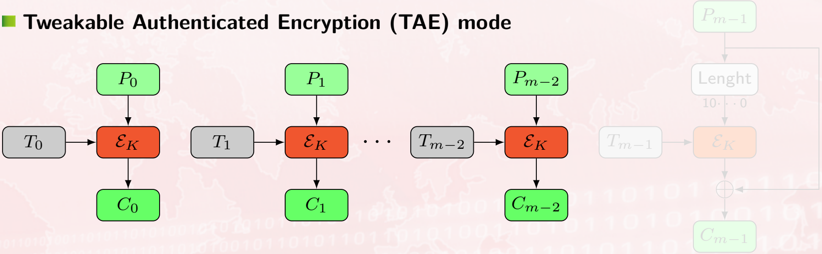


Tweakey scheduling algorithm of Scream

Scream is an iterative block cipher composed of N_s steps, each of them made of N_r rounds, introduced by Grosso in 2014. It takes as inputs a 128-bit block, a 128-bit key K and a 128-bit tweak $T = t_0||t_1$. The tweak is used as a "lightweight key schedule". The output of the step s is added by an XOR to a subkey equal to:

$$
\begin{array}{ll}
K \oplus (t_0||t_1) & \text{if } s = 3i, \\
K \oplus (t_0 \oplus t_1||t_0) & \text{if } s = 3i + 1, \\
K \oplus (t_1||t_0 \oplus t_1) & \text{if } s = 3i + 2.
\end{array}
$$

Tweakable Authenticated Encryption (TAE) mode

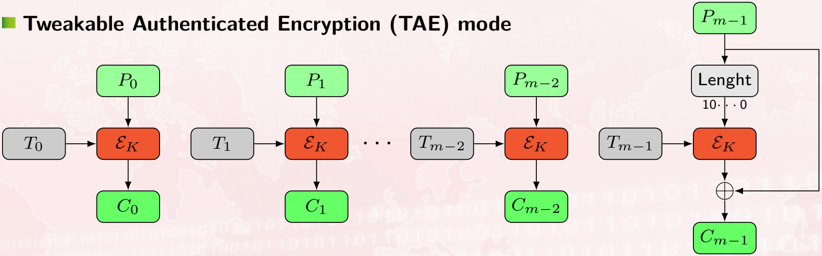


Tweakey scheduling algorithm of Scream

Scream is an iterative block cipher composed of N_s steps, each of them made of N_r rounds, introduced by Grosso in 2014. It takes as inputs a 128-bit block, a 128-bit key K and a 128-bit tweak $T = t_0||t_1$. The tweak is used as a "lightweight key schedule". The output of the step s is added by an XOR to a subkey equal to:

$$\begin{cases} K \oplus (t_0||t_1) & \text{if } s = 3i, \\ K \oplus (t_0 \oplus t_1||t_0) & \text{if } s = 3i+1, \\ K \oplus (t_1||t_0 \oplus t_1) & \text{if } s = 3i+2. \end{cases}$$

Tweakable Authenticated Encryption (TAE) mode

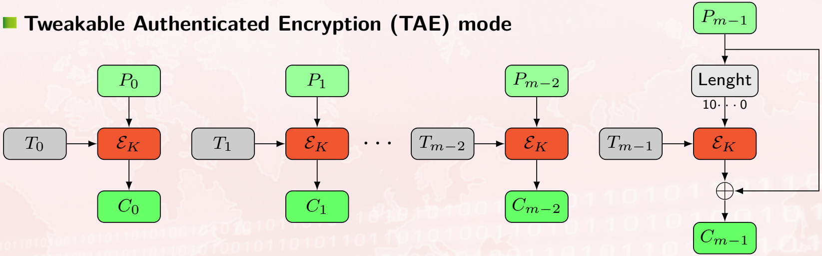


Tweakey scheduling algorithm of Scream

Scream is an iterative block cipher composed of N_s steps, each of them made of N_r rounds, introduced by Grosso in 2014. It takes as inputs a 128-bit block, a 128-bit key K and a 128-bit tweak $T = t_0||t_1$. The tweak is used as a "lightweight key schedule". The output of the step s is added by an XOR to a subkey equal to:

$$\begin{cases} K \oplus (t_0||t_1) & \text{if } s = 3i, \\ K \oplus (t_0 \oplus t_1||t_0) & \text{if } s = 3i+1, \\ K \oplus (t_1||t_0 \oplus t_1) & \text{if } s = 3i+2. \end{cases}$$

- **Tweakable Authenticated Encryption (TAE) mode**

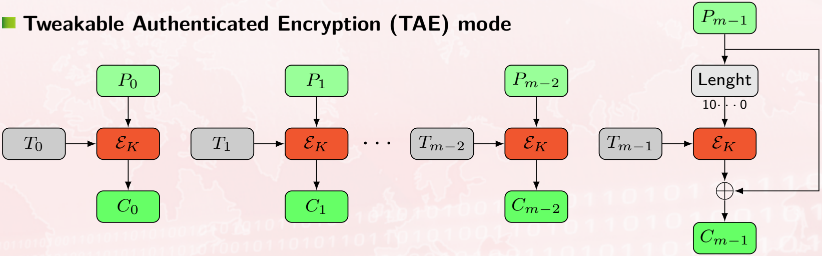


- **Tweakey scheduling algorithm of Scream**

Scream is an iterative block cipher composed of N_s steps, each of them made of N_r rounds, introduced by Grosso in 2014. It takes as inputs a 128-bit block, a 128-bit key K and a 128-bit tweak $T = t_0||t_1$. The tweak is used as a "lightweight key schedule". The output of the step s is added by an XOR to a subkey equal to:

$$\begin{cases} K \oplus (t_0||t_1) & \text{if } s = 3i, \\ K \oplus (t_0 \oplus t_1||t_0) & \text{if } s = 3i+1, \\ K \oplus (t_1||t_0 \oplus t_1) & \text{if } s = 3i+2. \end{cases}$$

Tweakable Authenticated Encryption (TAE) mode



Tweakey scheduling algorithm of Scream

Scream is an iterative block cipher composed of N_s steps, each of them made of N_r rounds, introduced by Grosso in 2014. It takes as inputs a 128-bit block, a 128-bit key K and a 128-bit tweak $T = t_0||t_1$. The tweak is used as a “lightweight key schedule”. The output of the step s is added by an XOR to a subkey equal to:

$$
\begin{array}{ll}
K \oplus (t_0||t_1) & \text{if } s = 3i, \\
K \oplus (t_0 \oplus t_1||t_0) & \text{if } s = 3i+1, \\
K \oplus (t_1||t_0 \oplus t_1) & \text{if } s = 3i+2.
\end{array}
$$

DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.

The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

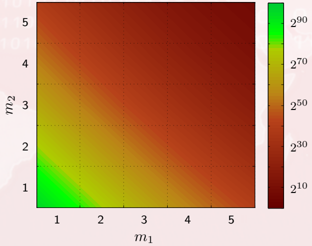
$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

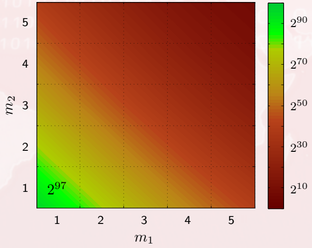


DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

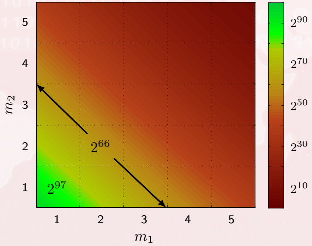


DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

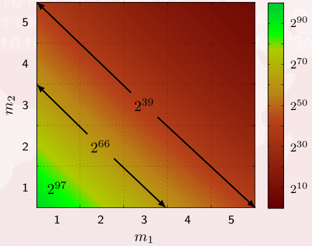


DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

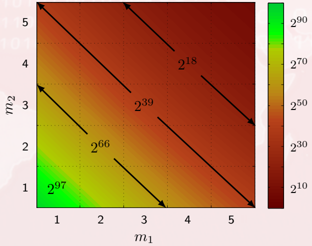


DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$

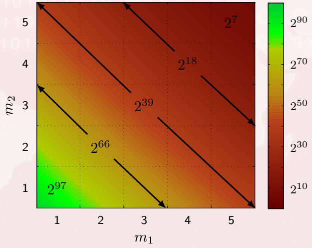


DFA on SCREAM

- Target the last two rows to obtain differentials (a_1,0x01) (resp. (a_2,0x02)) which allows to obtain $A_1 \approx 2.286$ (resp. $A_2 \approx 2.639$) candidates on some key bytes.
- The inner state is represented as a 8×16 bit array.

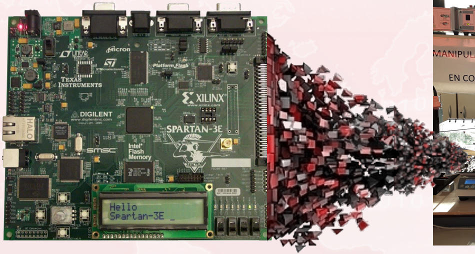
Therefore, the average number of remaining candidates for the key from m_1 (resp. m_2) random faults on the last (resp. penultimate) row is approximately:

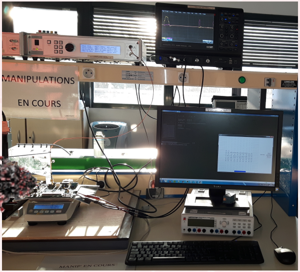
$$\left(\frac{252.075}{2^{m_1+m_2}} + \frac{1.286}{2^{m_2}} + \frac{1.639}{2^{m_1}} + 1\right)^{16}$$



The faults injection device

We used electromagnetic pulses to disrupt SCREAM execution. This approach requires no decapsulation of the chip and allows to precisely target a given time.





The SCREAM implementation used

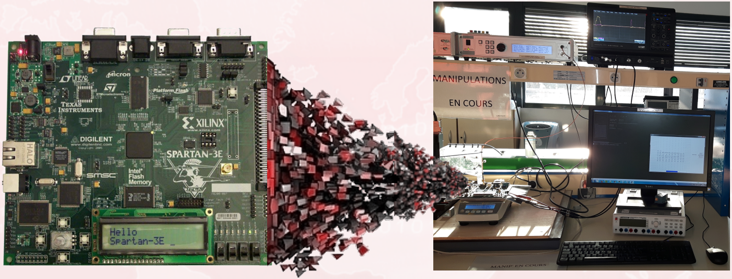
We have implemented and run the 128-bit reference VHDL code of SCREAM.

The FPGA die was composed of components CRYPTO, UART and FSM.

The input parameters were a 88-bit nonce, 2 ass. data blocks and 3 data blocks.

The faults injection device

We used electromagnetic pulses to disrupt SCREAM execution. This approach requires no decapsulation of the chip and allows to precisely target a given time.



The SCREAM implementation used

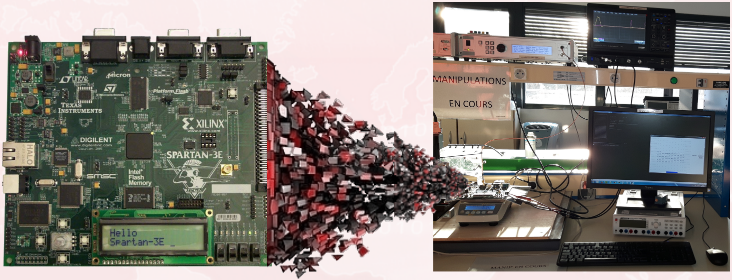
- We have implemented and run the 128-bit reference VHDL code of SCREAM.

  The FPGA die was composed of components CRYPTO, UART and FSM.

  The input parameters were a 88-bit nonce, 2 ass. data blocks and 3 data blocks.

The faults injection device

We used electromagnetic pulses to disrupt SCREAM execution. This approach requires no decapsulation of the chip and allows to precisely target a given time.



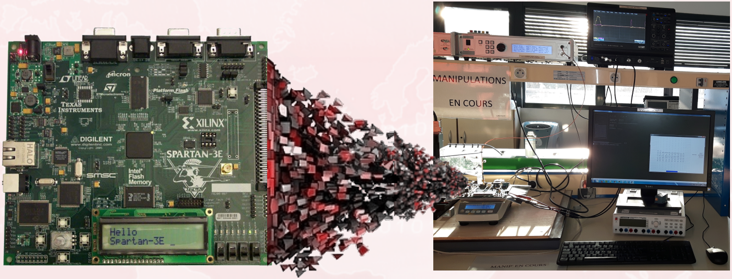
The SCREAM implementation used

- We have implemented and run the 128-bit reference VHDL code of SCREAM.
- The FPGA die was composed of components CRYPTO, UART and FSM.

The input parameters were a 88-bit nonce, 2 ass. data blocks and 3 data blocks.

The faults injection device

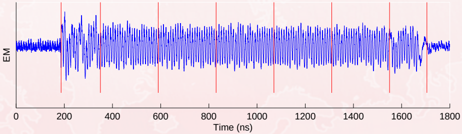
We used electromagnetic pulses to disrupt SCREAM execution. This approach requires no decapsulation of the chip and allows to precisely target a given time.



The SCREAM implementation used

- We have implemented and run the 128-bit reference VHDL code of SCREAM.
- The FPGA die was composed of components CRYPTO, UART and FSM.
- The input parameters were a 88-bit nonce, 2 ass. data blocks and 3 data blocks.

Electromagnetic radiations analysis of SCREAM



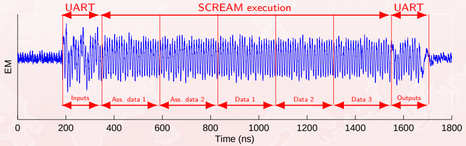
Cartography of the obtained faults on the full chip

The pulses were injected on 100 spatial positions distributed on a 10×10 grid.

On each, we tested 11 different temporal positions, 4 different voltages and we injected 2 pulses.

On the 8800 injections, we obtained 465 faults of which at most 88 to one spatial position.

Electromagnetic radiations analysis of SCREAM



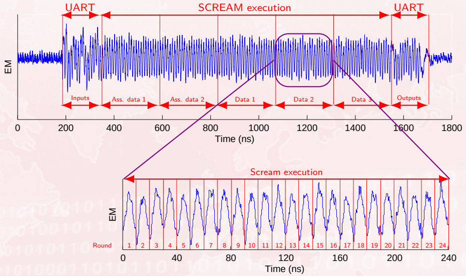
Cartography of the obtained faults on the full chip

The pulses were injected on 100 spatial positions distributed on a 10×10 grid.

On each, we tested 11 different temporal positions, 4 different voltages and we injected 2 pulses.

On the 8800 injections, we obtained 465 faults of which at most 88 to one spatial position.

Electromagnetic radiations analysis of SCREAM



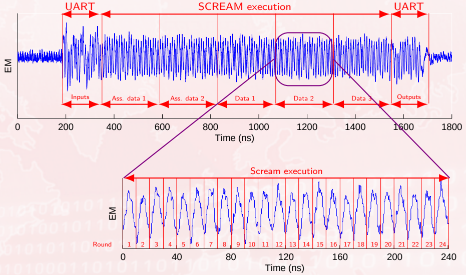
Cartography of the obtained faults on the full chip

The pulses were injected on 100 spatial positions distributed on a 10×10 grid.

On each, we tested 11 different temporal positions, 4 different voltages and we injected 2 pulses.

On the 8800 injections, we obtained 465 faults of which at most 88 to one spatial position.

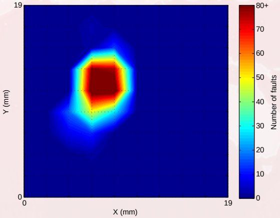
- **Electromagnetic radiations analysis of SCREAM**



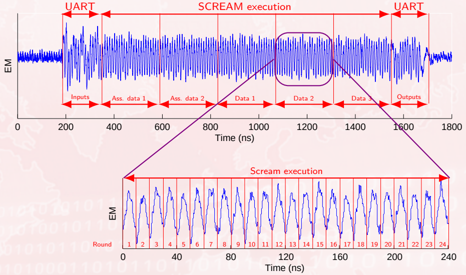
- **Cartography of the obtained faults on the full chip**
  - The pulses were injected on 100 spatial positions distributed on a 10×10 grid.

  On each, we tested 11 different temporal positions, 4 different voltages and we injected 2 pulses.

  On the 8800 injections, we obtained 465 faults of which at most 88 to one spatial position.



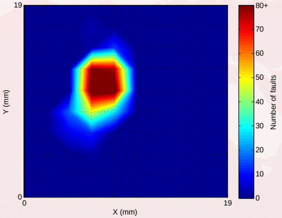
Electromagnetic radiations analysis of SCREAM



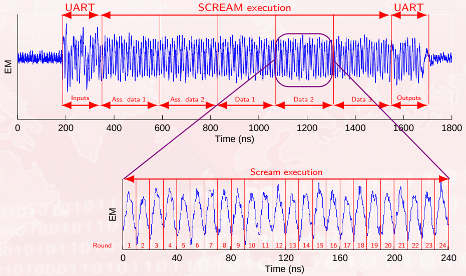
Cartography of the obtained faults on the full chip

- The pulses were injected on 100 spatial positions distributed on a 10×10 grid.
- On each, we tested 11 different temporal positions, 4 different voltages and we injected 2 pulses.

On the 8800 injections, we obtained 465 faults of which at most 88 to one spatial position.

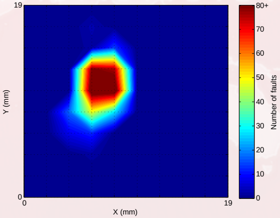


Electromagnetic radiations analysis of SCREAM



Cartography of the obtained faults on the full chip

- The pulses were injected on 100 spatial positions distributed on a 10×10 grid.
- On each, we tested 11 different temporal positions, 4 different voltages and we injected 2 pulses.
- On the 8800 injections, we obtained 465 faults of which at most 88 to one spatial position.



All obtained faults

- A total of 69250 pulses were injected on the sensitive area of the chip. We obtained 2482 faults, among which 937 were different.

For each fault, we verified if each byte could have been obtained by the same difference equal to 2^j with $0 \leq j \leq 7$ in input of the last substitution layer.

A total of 36 different faults complied with this property.

Gained knowledge

		ΔIn			
		0x01	0x04	0x08	$\mathcal{L}^{-1}(\mathrm{CT} \oplus K \oplus T)[i] \oplus C^{(23)}[i]$
$\Delta Y^{(24)}[i]$	$i = 1$	0xb8	$\emptyset$	0xb9	0x03
	$i = 2$	0x33	$\emptyset$	0x88	0x0e
	$i = 3$	0x47	0x4b	$\emptyset$	0x2e
	$i = 4$	0xca	0x54	0x3a	0xef
	$i = 5$	0x19	$\emptyset$	$\emptyset$	0x2b, 0x32, 0x4f, 0x56, 0x65 or 0x7c
	$i = 6$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	$i = 7$	0x2a	$\emptyset$	$\emptyset$	0xd1 or 0xfb
	$i = 8$	0xd5	$\emptyset$	0x1a	0xcb
	$i = 9$	0xd9	$\emptyset$	$\emptyset$	0x02 or 0xdb
	$i = 10$	0x5d	$\emptyset$	0x58	0x3f
	$i = 11$	0xfd	$\emptyset$	0xf9	0x48
	$i = 12$	0xcc	$\emptyset$	0xbc	0x59
	$i = 13$	0x2a	$\emptyset$	0x1a	0xd1
	$i = 14$	0x54	$\emptyset$	0xee	0xe4
	$i = 15$	0x8a	$\emptyset$	0x46	0x9e
	$i = 16$	0xc2	$\emptyset$	0xe9	0x97

We eventually obtained $6144 \approx 2^{12.58}$ candidates for $\mathcal{L}^{-1}(\mathrm{CT} \oplus K \oplus T) \oplus C^{(23)}$.

All obtained faults

- A total of 69250 pulses were injected on the sensitive area of the chip. We obtained 2482 faults, among which 937 were different.
- For each fault, we verified if each byte could have been obtained by the same difference equal to 2^j with $0 \leq j \leq 7$ in input of the last substitution layer.

A total of 36 different faults complied with this property.

Gained knowledge

		ΔIn			
		0x01	0x04	0x08	$\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T)[i] \oplus C^{(23)}[i]$
$\Delta Y^{(24)}[i]$	$i = 1$	0xb8	$\emptyset$	0xb9	0x03
	$i = 2$	0x33	$\emptyset$	0x88	0x0e
	$i = 3$	0x47	0x4b	$\emptyset$	0x2e
	$i = 4$	0xca	0x54	0x3a	0xef
	$i = 5$	0x19	$\emptyset$	$\emptyset$	0x2b, 0x32, 0x4f, 0x56, 0x65 or 0x7c
	$i = 6$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	$i = 7$	0x2a	$\emptyset$	$\emptyset$	0xd1 or 0xfb
	$i = 8$	0xd5	$\emptyset$	0x1a	0xcb
	$i = 9$	0xd9	$\emptyset$	$\emptyset$	0x02 or 0xdb
	$i = 10$	0x5d	$\emptyset$	0x58	0x3f
	$i = 11$	0xfd	$\emptyset$	0xf9	0x48
	$i = 12$	0xcc	$\emptyset$	0xbc	0x59
	$i = 13$	0x2a	$\emptyset$	0x1a	0xd1
	$i = 14$	0x54	$\emptyset$	0xee	0xe4
	$i = 15$	0x8a	$\emptyset$	0x46	0x9e
	$i = 16$	0xc2	$\emptyset$	0xe9	0x97

We eventually obtained $6144 \approx 2^{12.58}$ candidates for $\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T) \oplus C^{(23)}$.

All obtained faults

- A total of 69250 pulses were injected on the sensitive area of the chip. We obtained 2482 faults, among which 937 were different.
- For each fault, we verified if each byte could have been obtained by the same difference equal to 2^j with $0 \leq j \leq 7$ in input of the last substitution layer.
- A total of 36 different faults complied with this property.

Gained knowledge

		ΔIn			$\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T)[i] \oplus C^{(23)}[i]$
		0x01	0x04	0x08	
$\Delta Y^{(24)}[i]$	$i = 1$	0xb8	$\emptyset$	0xb9	0x03
	$i = 2$	0x33	$\emptyset$	0x88	0x0e
	$i = 3$	0x47	0x4b	$\emptyset$	0x2e
	$i = 4$	0xca	0x54	0x3a	0xef
	$i = 5$	0x19	$\emptyset$	$\emptyset$	0x2b, 0x32, 0x4f, 0x56, 0x65 or 0x7c
	$i = 6$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	$i = 7$	0x2a	$\emptyset$	$\emptyset$	0xd1 or 0xfb
	$i = 8$	0xd5	$\emptyset$	0x1a	0xcb
	$i = 9$	0xd9	$\emptyset$	$\emptyset$	0x02 or 0xdb
	$i = 10$	0x5d	$\emptyset$	0x58	0x3f
	$i = 11$	0xfd	$\emptyset$	0xf9	0x48
	$i = 12$	0xcc	$\emptyset$	0xbc	0x59
	$i = 13$	0x2a	$\emptyset$	0x1a	0xd1
	$i = 14$	0x54	$\emptyset$	0xee	0xe4
	$i = 15$	0x8a	$\emptyset$	0x46	0x9e
	$i = 16$	0xc2	$\emptyset$	0xe9	0x97

We eventually obtained $6144 \approx 2^{12.58}$ candidates for $\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T) \oplus C^{(23)}$.

All obtained faults

- A total of 69250 pulses were injected on the sensitive area of the chip. We obtained 2482 faults, among which 937 were different.
- For each fault, we verified if each byte could have been obtained by the same difference equal to 2^j with $0 \leq j \leq 7$ in input of the last substitution layer.
- A total of 36 different faults complied with this property.

Gained knowledge

		ΔIn			$\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T)[i] \oplus C^{(23)}[i]$
		0x01	0x04	0x08	
$\Delta Y^{(24)}[i]$	$i = 1$	0xb8	∅	0xb9	0x03
	$i = 2$	0x33	∅	0x88	0x0e
	$i = 3$	0x47	0x4b	∅	0x2e
	$i = 4$	0xca	0x54	0x3a	0xef
	$i = 5$	0x19	∅	∅	0x2b, 0x32, 0x4f, 0x56, 0x65 or 0x7c
	$i = 6$	∅	∅	∅	∅
	$i = 7$	0x2a	∅	∅	0xd1 or 0xfb
	$i = 8$	0xd5	∅	0x1a	0xcb
	$i = 9$	0xd9	∅	∅	0x02 or 0xdb
	$i = 10$	0x5d	∅	0x58	0x3f
	$i = 11$	0xfd	∅	0xf9	0x48
	$i = 12$	0xcc	∅	0xbc	0x59
	$i = 13$	0x2a	∅	0x1a	0xd1
	$i = 14$	0x54	∅	0xee	0xe4
	$i = 15$	0x8a	∅	0x46	0x9e
	$i = 16$	0xc2	∅	0xe9	0x97

We eventually obtained $6144 \approx 2^{12.58}$ candidates for $\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T) \oplus C^{(23)}$.

All obtained faults

- A total of 69250 pulses were injected on the sensitive area of the chip. We obtained 2482 faults, among which 937 were different.
- For each fault, we verified if each byte could have been obtained by the same difference equal to 2^j with $0 \leq j \leq 7$ in input of the last substitution layer.
- A total of 36 different faults complied with this property.

Gained knowledge

		ΔIn			$\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T)[i] \oplus C^{(23)}[i]$
		0x01	0x04	0x08	
$\Delta Y^{(24)}[i]$	$i = 1$	0xb8	∅	0xb9	0x03
	$i = 2$	0x33	∅	0x88	0x0e
	$i = 3$	0x47	0x4b	∅	0x2e
	$i = 4$	0xca	0x54	0x3a	0xef
	$i = 5$	0x19	∅	∅	0x2b, 0x32, 0x4f, 0x56, 0x65 or 0x7c
	$i = 6$	∅	∅	∅	∅
	$i = 7$	0x2a	∅	∅	0xd1 or 0xfb
	$i = 8$	0xd5	∅	0x1a	0xcb
	$i = 9$	0xd9	∅	∅	0x02 or 0xdb
	$i = 10$	0x5d	∅	0x58	0x3f
	$i = 11$	0xfd	∅	0xf9	0x48
	$i = 12$	0xcc	∅	0xbc	0x59
	$i = 13$	0x2a	∅	0x1a	0xd1
	$i = 14$	0x54	∅	0xee	0xe4
	$i = 15$	0x8a	∅	0x46	0x9e
	$i = 16$	0xc2	∅	0xe9	0x97

We eventually obtained $6144 \approx 2^{12.58}$ candidates for $\mathcal{L}^{-1}(\text{CT} \oplus K \oplus T) \oplus C^{(23)}$.

Countermeasures

1. LS-Designs
2. Applying DFA on LS-Designs
   - General principle
   - Depending on the fault model
3. Practical implementation of the DFA on SCREAM
   - The TAE mode SCREAM
   - DFA on SCREAM
   - Practical implementation
4. Countermeasures
   - Modes of operation
   - Masking
   - Internal Redundancy Countermeasure
5. Conclusion and perspectives

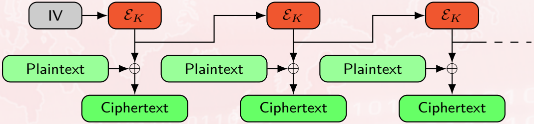
Cipher not applied to data

In order to encrypt data, a block cipher is always used with a mode of operation. It turns out that some well-known modes - standardized and already used in practice - thwart our DFA. It is the case for the modes which use an nonce to encrypt data.

OFB mode encryption

Cipher not applied to data

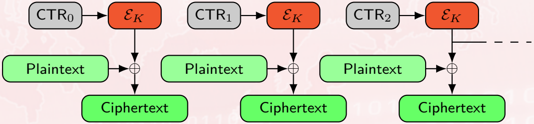
In order to encrypt data, a block cipher is always used with a mode of operation. It turns out that some well-known modes - standardized and already used in practice - thwart our DFA. It is the case for the modes which use an nonce to encrypt data.



OFB mode encryption

Cipher not applied to data

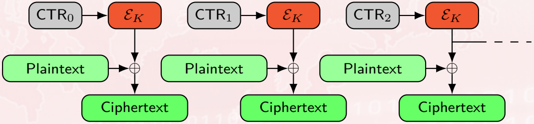
In order to encrypt data, a block cipher is always used with a mode of operation. It turns out that some well-known modes - standardized and already used in practice - thwart our DFA. It is the case for the modes which use an nonce to encrypt data.



CTR mode encryption

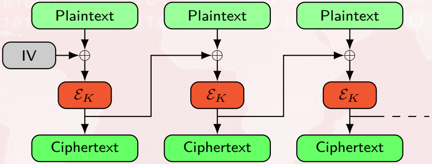
Cipher not applied to data

In order to encrypt data, a block cipher is always used with a mode of operation. It turns out that some well-known modes - standardized and already used in practice - thwart our DFA. It is the case for the modes which use an nonce to encrypt data.



CTR mode encryption

Cipher applied to data



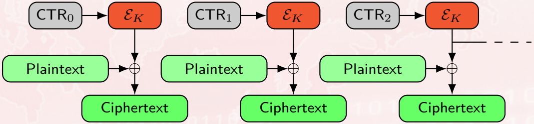
CBC mode encryption

In this case, the IV must be unpredictable by the attacker in advance, otherwise:

$$\forall (IV_1, IV_2), \exists (P_1, P_2) \text{ such that } P_1 \oplus IV_1 = P_2 \oplus IV_2.$$

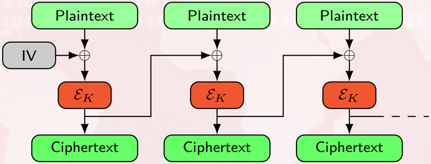
Cipher not applied to data

In order to encrypt data, a block cipher is always used with a mode of operation. It turns out that some well-known modes - standardized and already used in practice - thwart our DFA. It is the case for the modes which use an nonce to encrypt data.



CTR mode encryption

Cipher applied to data



CBC mode encryption

In this case, the IV must be unpredictable by the attacker in advance, otherwise:

$$\forall(IV_1, IV_2),\ \exists(P_1, P_2) \text{ such that } P_1 \oplus IV_1 = P_2 \oplus IV_2.$$

Description

Add a random value RV to the state SM in the middle of the encryption $\mathcal{E}_K$.

Then, to mount a DFA on the encryption, an attacker must obtain a correct ciphertext $C = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_1) \oplus RV_1)$ and a faulty one $C^* = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_2) \oplus RV_2)$ such that:

$$\mathcal{E}_K^{(0)}(P_1) \oplus RV_1 = \mathcal{E}_K^{(0)}(P_2) \oplus RV_2.$$

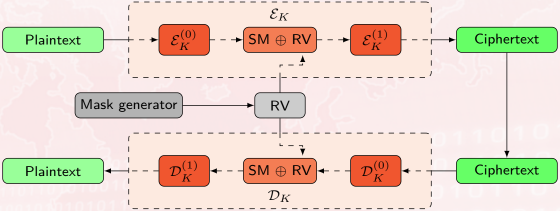
From the birthday paradox, this requires $2^{n/2}$ fault injections where n is the block size.

Cost

The cost depends on the choice of the random mask generation. A simple LFSR implemented in hardware has a low cost with respect to IoT constraints.

Description

Add a random value RV to the state SM in the middle of the encryption $\mathcal{E}_K$.



Then, to mount a DFA on the encryption, an attacker must obtain a correct ciphertext $C = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_1) \oplus RV_1)$ and a faulty one $C^* = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_2) \oplus RV_2)$ such that:

$$\mathcal{E}_K^{(0)}(P_1) \oplus RV_1 = \mathcal{E}_K^{(0)}(P_2) \oplus RV_2.$$

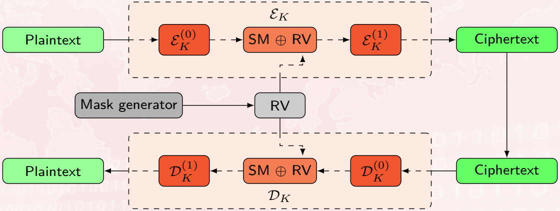
From the birthday paradox, this requires $2^{n/2}$ fault injections where n is the block size.

Cost

The cost depends on the choice of the random mask generation. A simple LFSR implemented in hardware has a low cost with respect to IoT constraints.

Description

Add a random value RV to the state SM in the middle of the encryption $\mathcal{E}_K$.



Then, to mount a DFA on the encryption, an attacker must obtain a correct ciphertext $C = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_1) \oplus RV_1)$ and a faulty one $C^* = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_2) \oplus RV_2)$ such that:

$$\mathcal{E}_K^{(0)}(P_1) \oplus RV_1 = \mathcal{E}_K^{(0)}(P_2) \oplus RV_2.$$

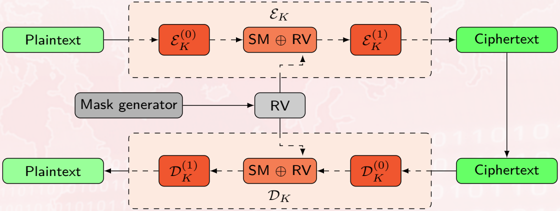
From the birthday paradox, this requires $2^{n/2}$ fault injections where n is the block size.

Cost

The cost depends on the choice of the random mask generation. A simple LFSR implemented in hardware has a low cost with respect to IoT constraints.

Description

Add a random value RV to the state SM in the middle of the encryption $\mathcal{E}_K$.



Then, to mount a DFA on the encryption, an attacker must obtain a correct ciphertext $C = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_1) \oplus RV_1)$ and a faulty one $C^* = \mathcal{E}_K^{(1)}(\mathcal{E}_K^{(0)}(P_2) \oplus RV_2)$ such that:

$$\mathcal{E}_K^{(0)}(P_1) \oplus RV_1 = \mathcal{E}_K^{(0)}(P_2) \oplus RV_2.$$

From the birthday paradox, this requires $2^{n/2}$ fault injections where n is the block size.

Cost

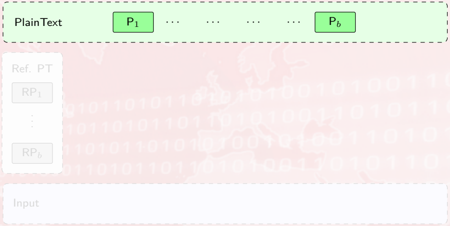
The cost depends on the choice of the random mask generation. A simple LFSR implemented in hardware has a low cost with respect to IoT constraints.

Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.

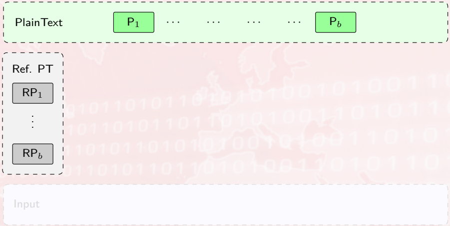
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



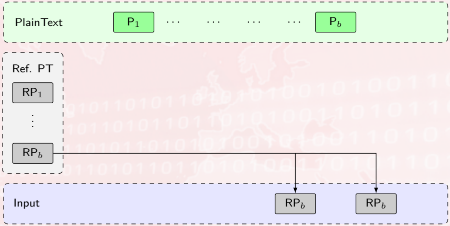
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



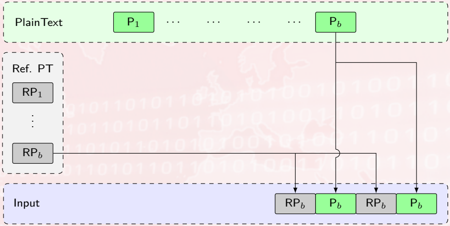
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



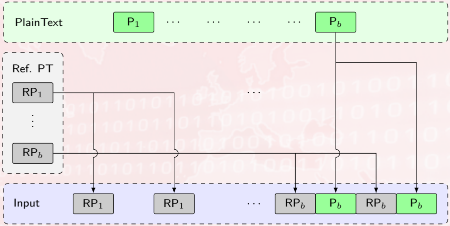
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



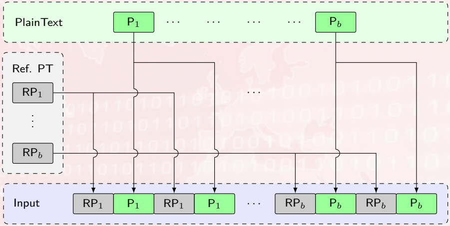
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



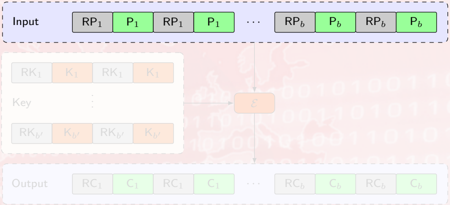
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



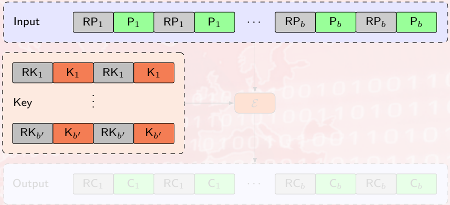
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



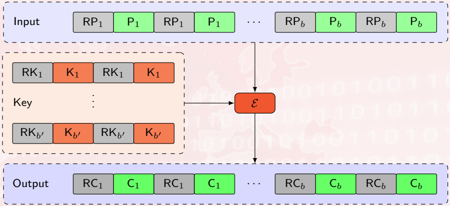
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



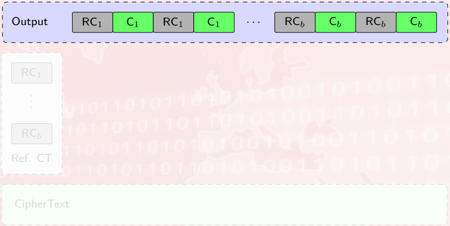
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



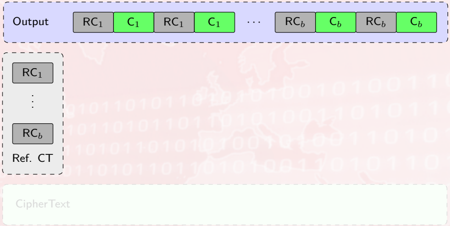
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



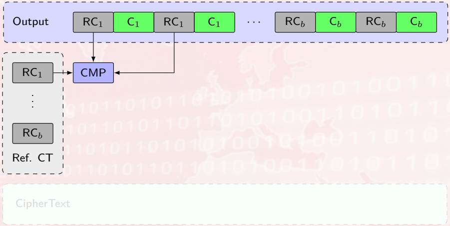
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



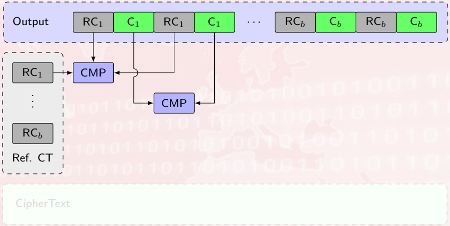
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



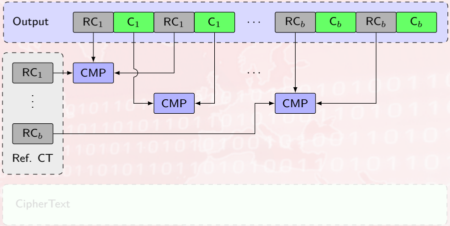
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



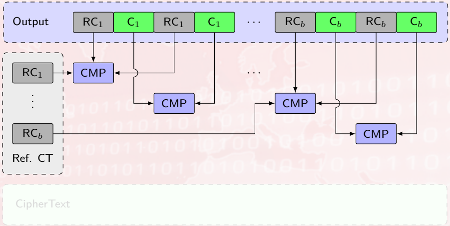
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



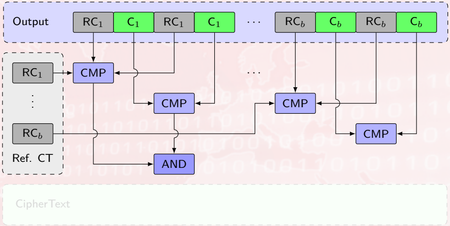
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



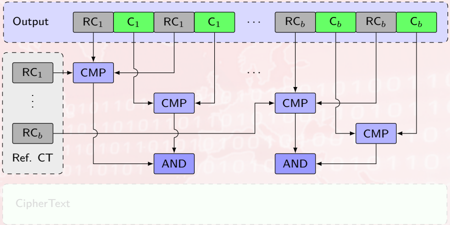
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



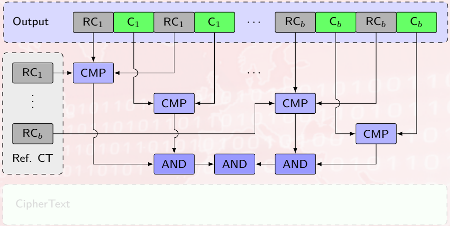
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



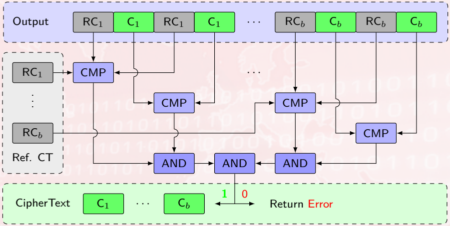
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



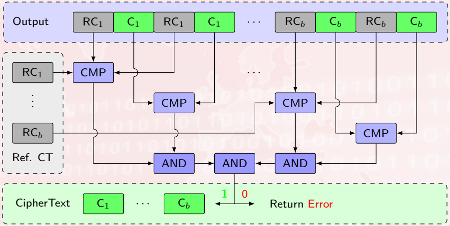
Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



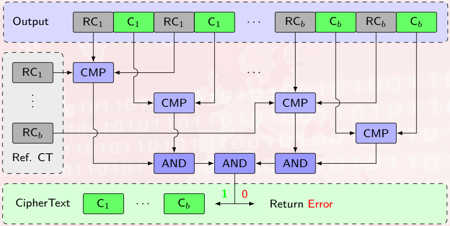
Cost

IRC simply uses bitwise operators on 32-bit and use SIMD instructions - or masks depending on the targeted device - to replace nonlinear operations.

Therefore, we obtain performances close to those on an 8-bit architecture while having a structure that intrinsically protects against DFA.

Description

IRC consists in using efficient 8-bit implementations but from 32-bit instructions systematically operating as a whole on the 4 bytes of a 32-bit word. It uses reference blocks to thwart fault attacks, especially skip instruction for which it is very effective.



Cost

IRC simply uses bitwise operators on 32-bit and use SIMD instructions - or masks depending on the targeted device - to replace nonlinear operations.
Therefore, we obtain performances close to those on an 8-bit architecture while having a structure that intrinsically protects against DFA.

Conclusion and perspectives

1 LS-Designs

2 Applying DFA on LS-Designs
- General principle
- Depending on the fault model

3 Practical implementation of the DFA on SCREAM
- The TAE mode SCREAM
- DFA on SCREAM
- Practical implementation

4 Countermeasures
- Modes of operation
- Masking
- Internal Redundancy Countermeasure

5 Conclusion and perspectives

Conclusion

- General method for Differential Fault Analysis on any block cipher based on LS-designs and other families of SPN with similar structures from only 2 faults in the best cases.
- Successfully perform such an attack against a hardware implementation of SCREAM, using the TLS-Design Scream with a fixed tweak.
- Faults were injected using EM pulses, which constitutes a low-cost means of injection.
- Resistance against DFA is important for an LS-Design, which will be dedicated to low-end devices thanks to its lightness.
- Some countermeasures which leave the cipher still efficient for IoT devices, especially a new kind of countermeasure: the so-called Internal Redundancy Countermeasure.

Perspectives

- Apply IRC on other block ciphers and also propose a generic method to deploy it on stream ciphers: will be studied in future work.

Conclusion and perspectives

Conclusion

- General method for Differential Fault Analysis on any block cipher based on LS-designs and other families of SPN with similar structures from only 2 faults in the best cases.
- Successfully perform such an attack against a hardware implementation of SCREAM, using the TLS-Design Scream with a fixed tweak.
- Faults were injected using EM pulses, which constitutes a low-cost means of injection.
- Resistance against DFA is important for an LS-Design, which will be dedicated to low-end devices thanks to its lightness.
- Some countermeasures which leave the cipher still efficient for IoT devices, especially a new kind of countermeasure: the so-called Internal Redundancy Countermeasure.

Perspectives

- Apply IRC on other block ciphers and also propose a generic method to deploy it on stream ciphers: will be studied in future work.

Conclusion and perspectives

Conclusion

- General method for Differential Fault Analysis on any block cipher based on LS-designs and other families of SPN with similar structures from only 2 faults in the best cases.
- Successfully perform such an attack against a hardware implementation of SCREAM, using the TLS-Design Scream with a fixed tweak.
- Faults were injected using EM pulses, which constitutes a low-cost means of injection.
- Resistance against DFA is important for an LS-Design, which will be dedicated to low-end devices thanks to its lightness.
- Some countermeasures which leave the cipher still efficient for IoT devices, especially a new kind of countermeasure: the so-called Internal Redundancy Countermeasure.

Perspectives

- Apply IRC on other block ciphers and also propose a generic method to deploy it on stream ciphers: will be studied in future work.

Conclusion and perspectives

Conclusion

- General method for Differential Fault Analysis on any block cipher based on LS-designs and other families of SPN with similar structures from only 2 faults in the best cases.
- Successfully perform such an attack against a hardware implementation of SCREAM, using the TLS-Design Scream with a fixed tweak.
- Faults were injected using EM pulses, which constitutes a low-cost means of injection.
- Resistance against DFA is important for an LS-Design, which will be dedicated to low-end devices thanks to its lightness.
- Some countermeasures which leave the cipher still efficient for IoT devices, especially a new kind of countermeasure: the so-called Internal Redundancy Countermeasure.

Perspectives

- Apply IRC on other block ciphers and also propose a generic method to deploy it on stream ciphers: will be studied in future work.

Conclusion

- General method for Differential Fault Analysis on any block cipher based on LS-designs and other families of SPN with similar structures from only 2 faults in the best cases.
- Successfully perform such an attack against a hardware implementation of SCREAM, using the TLS-Design Scream with a fixed tweak.
- Faults were injected using EM pulses, which constitutes a low-cost means of injection.
- Resistance against DFA is important for an LS-Design, which will be dedicated to low-end devices thanks to its lightness.
- Some countermeasures which leave the cipher still efficient for IoT devices, especially a new kind of countermeasure: the so-called Internal Redundancy Countermeasure.

Perspectives

Apply IRC on other block ciphers and also propose a generic method to deploy it on stream ciphers: will be studied in future work.

- **Conclusion**
  - General method for Differential Fault Analysis on any block cipher based on LS-designs and other families of SPN with similar structures from only 2 faults in the best cases.
  - Successfully perform such an attack against a hardware implementation of SCREAM, using the TLS-Design Scream with a fixed tweak.
  - Faults were injected using EM pulses, which constitutes a low-cost means of injection.
  - Resistance against DFA is important for an LS-Design, which will be dedicated to low-end devices thanks to its lightness.
  - Some countermeasures which leave the cipher still efficient for IoT devices, especially a new kind of countermeasure: the so-called Internal Redundancy Countermeasure.

- **Perspectives**
  - Apply IRC on other block ciphers and also propose a generic method to deploy it on stream ciphers: will be studied in future work.

DE LA RECHERCHE À L'INDUSTRIE

THANKS FOR YOUR ATTENTION

Commissariat à l'énergie atomique et aux énergies alternatives
Benjamin Lac | CEA-Tech/DPACA/LSAS

Public Industrial and Commercial Establishment | RCS Paris B 775 685 019