Error Detection by Parity Modification for the 128-bit Serpent Encryption Algorithm

Grigori Kuznetsov, Ramesh Karri and Michael Gössel
Polytechnical University Brooklyn, USA
University of Potsdam, Germany

Modified input parity

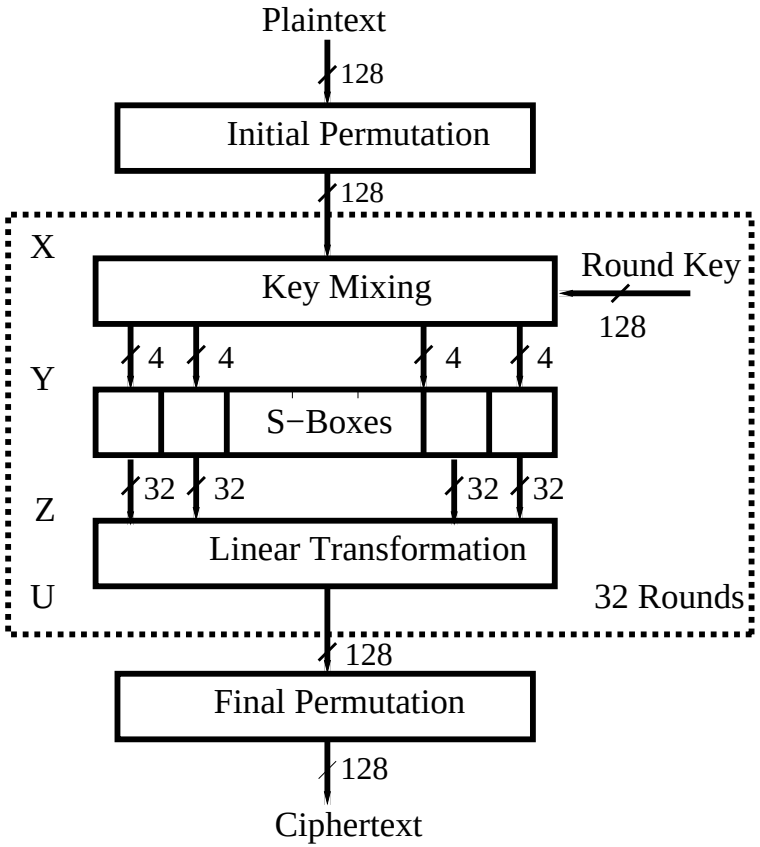
Essence of the proposed method

- The input-parity is determined and modified step by step in accordance with the processing steps of the cryptographic algorithm into the output-parity and compared with the actual output parity
- Often the same hardware can be used for the computation of the input-parity and of the output-parity

Serpent Encryption Algorithm

- Initial permutation
- 32 rounds
  - componentwise key-addition
  - 32 nonlinear substitution boxes
  - linear transformation
- Final permutation

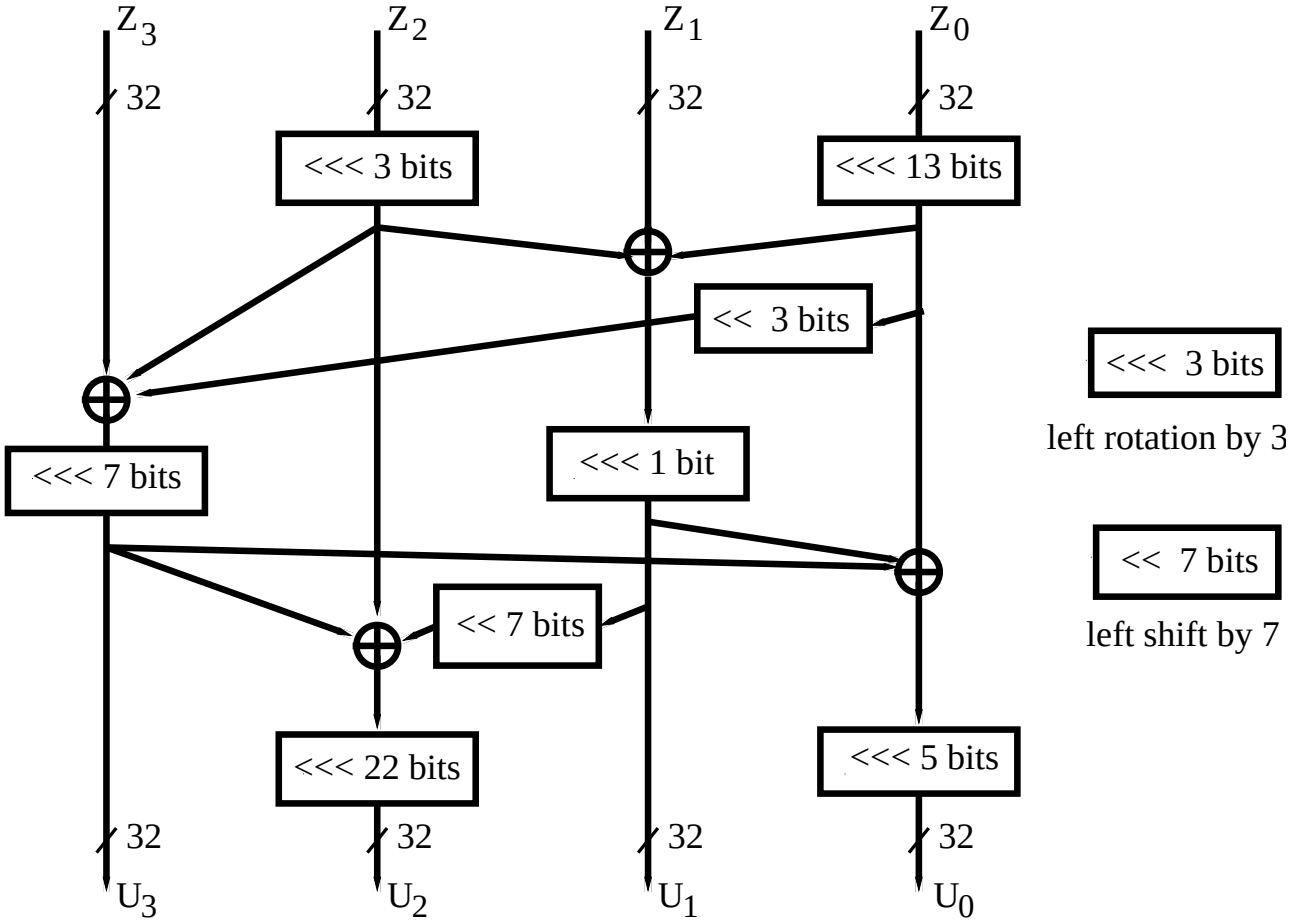
Serpent Encryption Algorithm



Serpent Encryption Algorithm: S-boxes

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Y	3	8	15	1	10	6	5	11	14	13	4	2	7	0	9	12

Serpent algorithm: Linear Transformation



Ordinary Parity Prediction

$y_1(x) = f_1(x), \cdots, y_m(x) = f_m(x)$

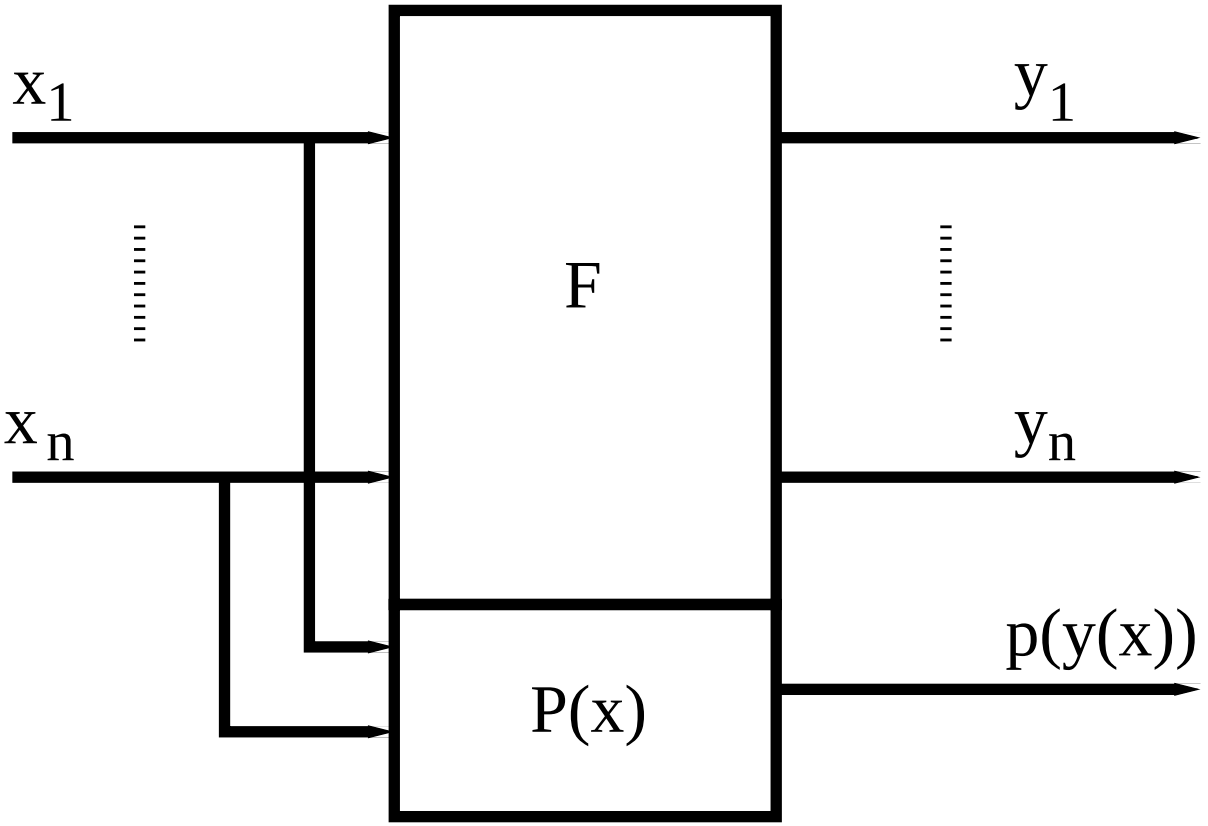
- $P(y(x)) = \underbrace{y_1(x) \oplus \cdots \oplus y_m(x)}_{optimized}$,
- $P(y(x))$ is determined as a function of the inputs

Ordinary Parity Prediction

$y_1(x) = f_1(x), \cdots, y_m(x) = f_m(x)$

- $P(y) = y_1 \oplus \cdots y_n$,
  $P(y)$ is determined as the XOR-sum of the circuit-outputs
- $P(y(x))$ is compared with $P(y)$.

Ordinary Parity Prediction



Proposed Method: Modified Input Parity

- Input-parity $\oplus$ Output-parity: $\underbrace{P(x) \oplus P(y(x))}_{optimized} =$

$$\underbrace{x_1 \oplus \cdots \oplus x_n \oplus y_1(x) \oplus \cdots \oplus y_n(x)}_{optimized}$$

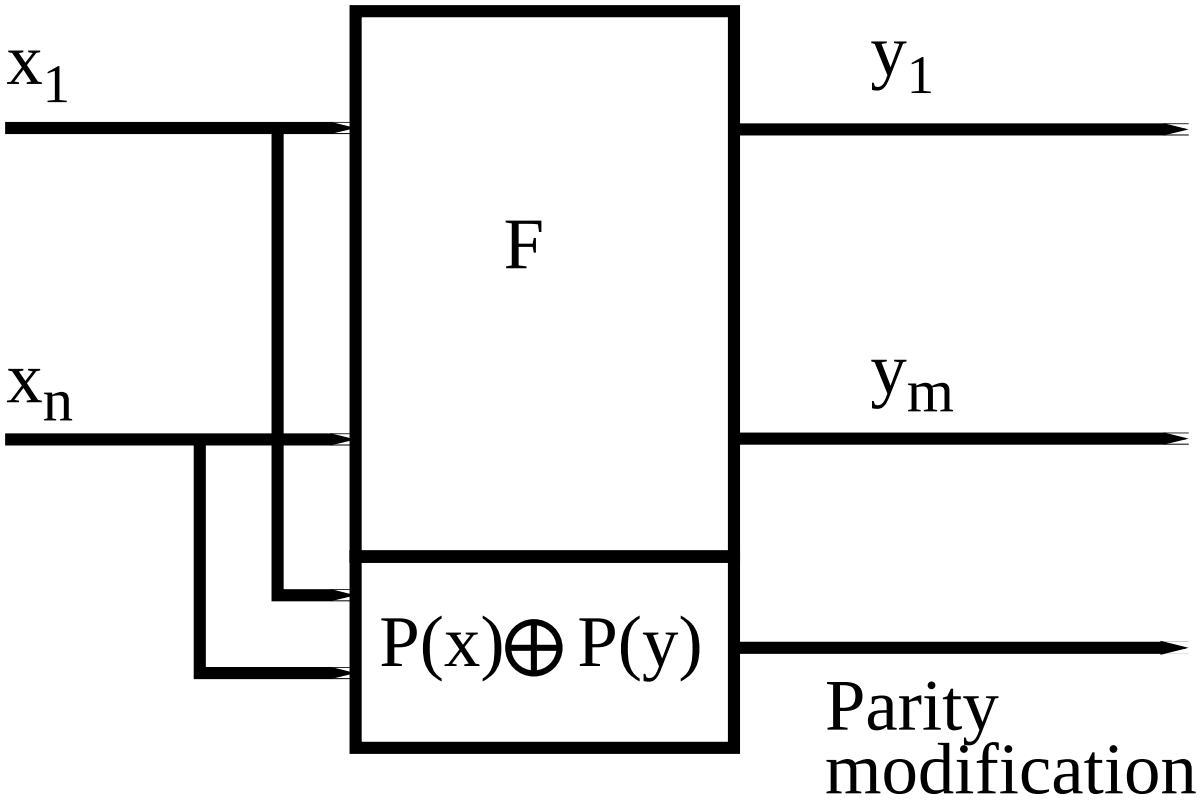
Proposed Method: Modified Parity

- Input parity P(x) $= x_1 \oplus \ldots \oplus x_n$
- The input-parity $P(x)$ is modified by adding modulo 2 $P(x) \oplus P(y(x))$ to the output parity $P(y)$

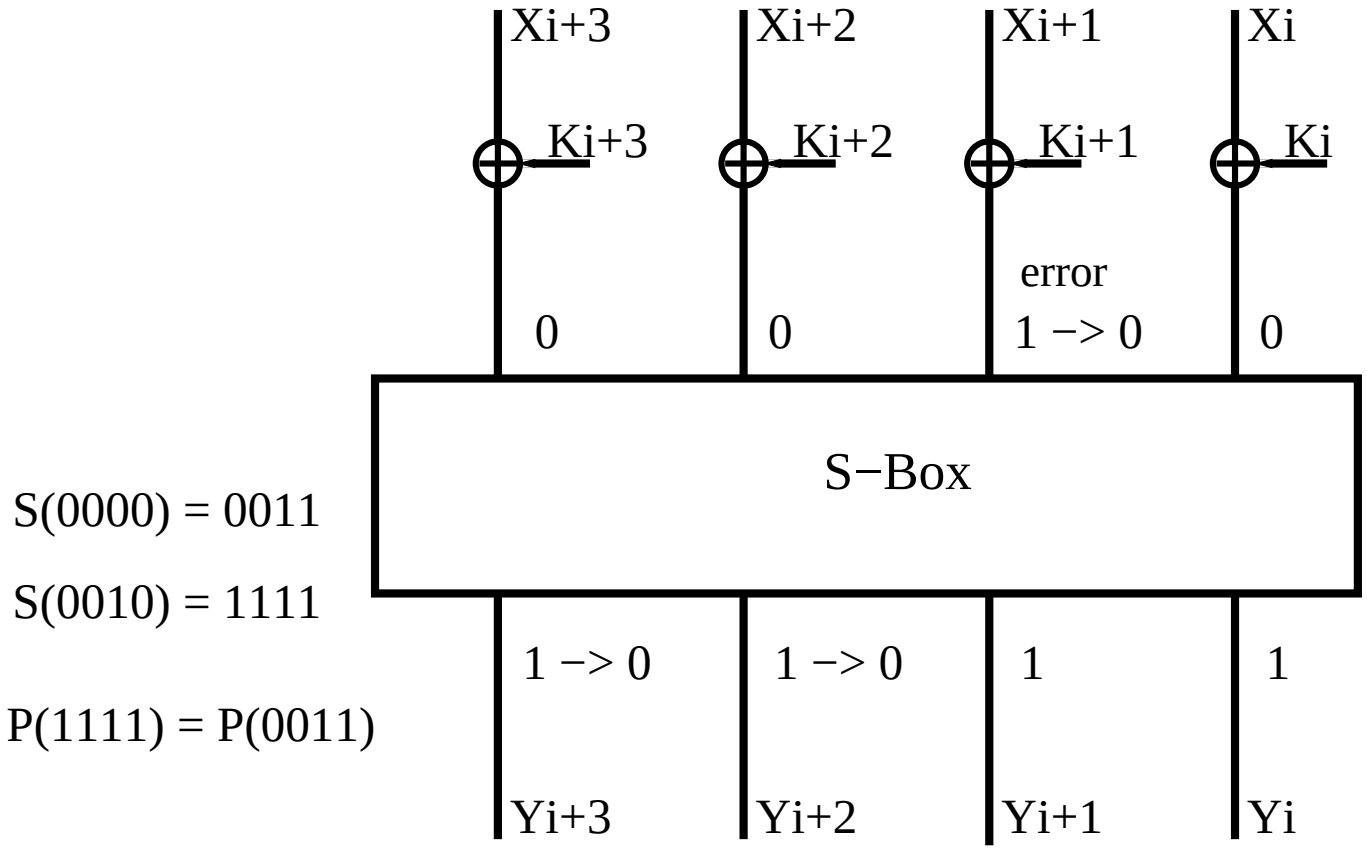
Proposed Method: Modified Parity

- Output parity P(y) $= y_1 \oplus \ldots \oplus y_n$
- $P(x) \oplus [P(x) \oplus P(y(x))]$ is compared with $P(y)$

Modified Parity



Example with Serpent S-Box

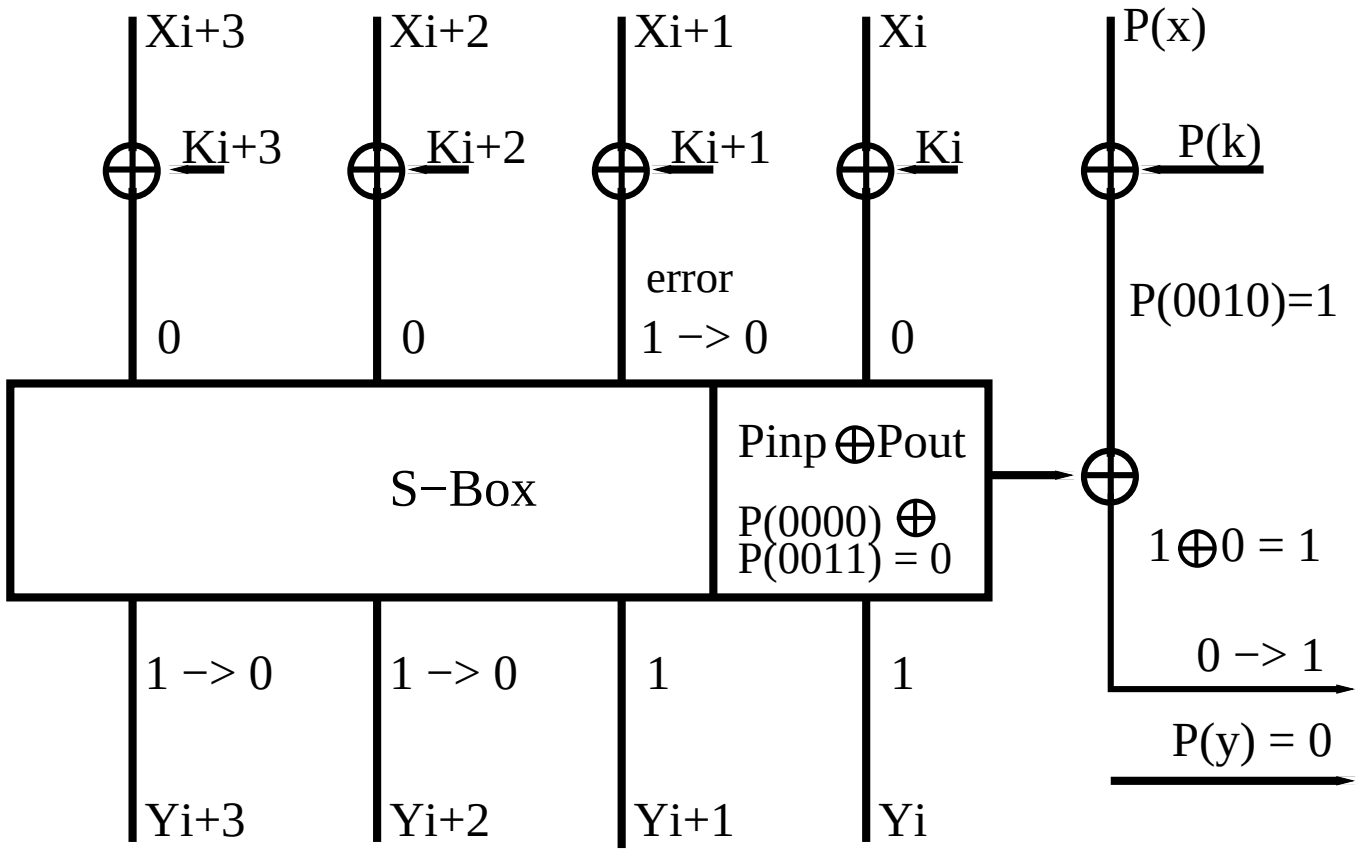


S(0000) = 0011

S(0010) = 1111

P(1111) = P(0011)

Example with Serpent S-Box



Parity-modifications for :Key-addition modulo 2

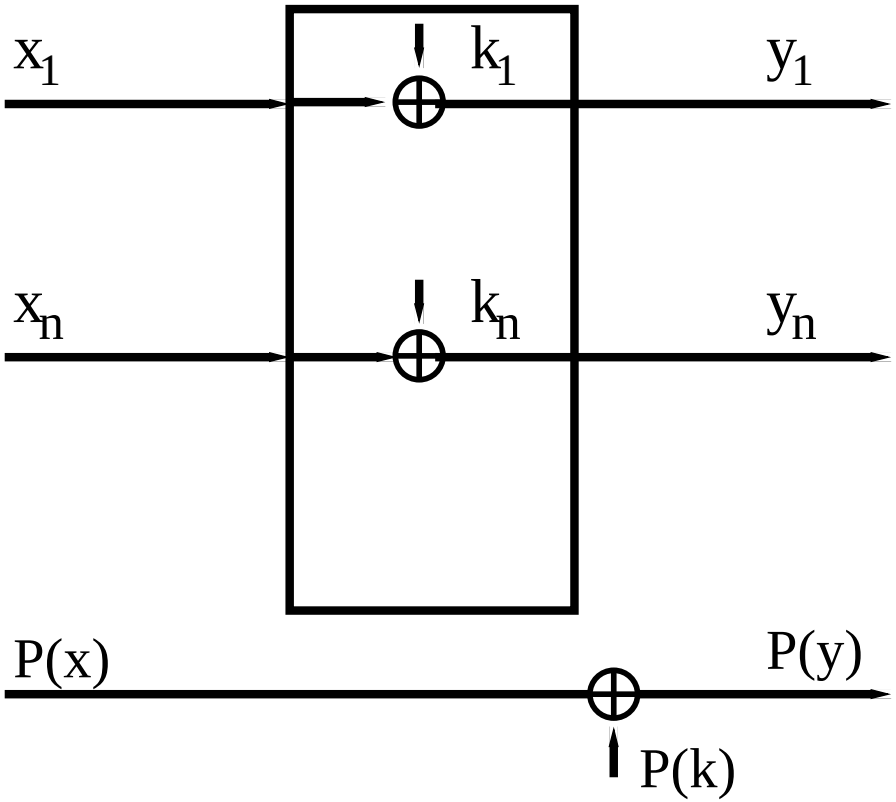
$y_1 = x_1 \oplus k_1, \cdots, y_m = x_m \oplus k_m$

$P(y(x)) = x_1 \oplus k_1 \oplus \cdots \oplus y_m \oplus k_m = P(x) \oplus P(k)$

with $P(k) = k_1 \oplus \cdots \oplus k_m$

Modification: $\oplus P(k)$, 1-bit operation

Key addition



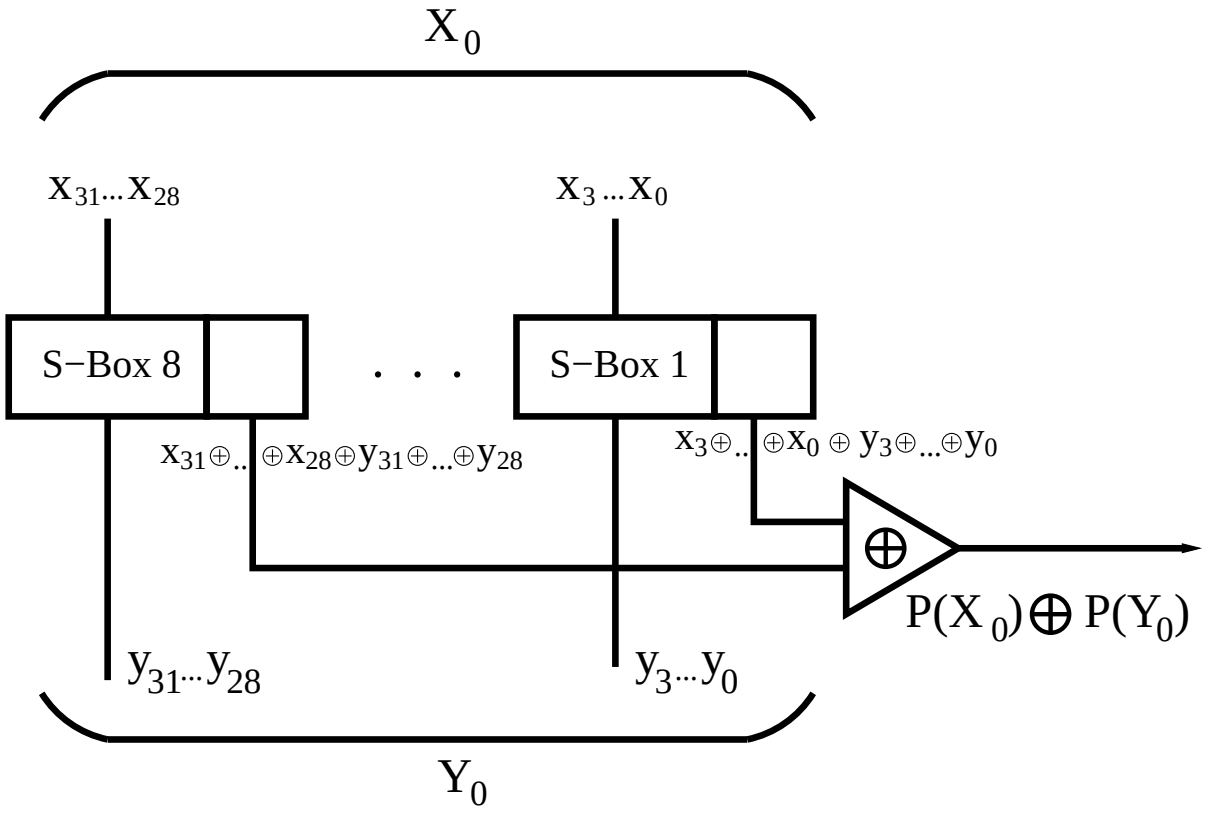
$$P(x) \oplus P(y) = P(k) = k_1 \oplus \ldots \oplus k_n$$

Parity-modifications for :Non-linear Transformation by an S-box

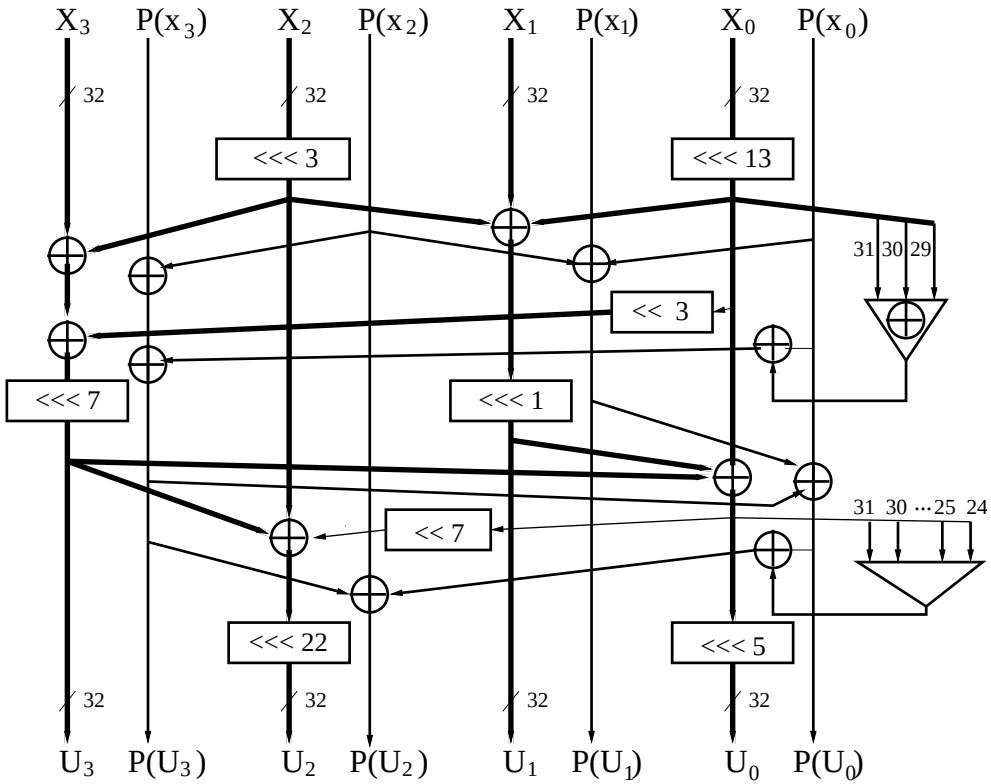
Example: S-box with four inputs $x_1, x_2, x_3, x_4 = x$ and four outputs y_1, y_2, y_3, y_4 with
$y_1(x) = S_1(x), \cdots, y_4(x) = S_4(x)$

$$y_5 = x_1 \oplus \cdots \oplus x_4 \oplus y_1(x_{1-4}) \oplus \cdots \oplus y_4(x_{1-4})$$

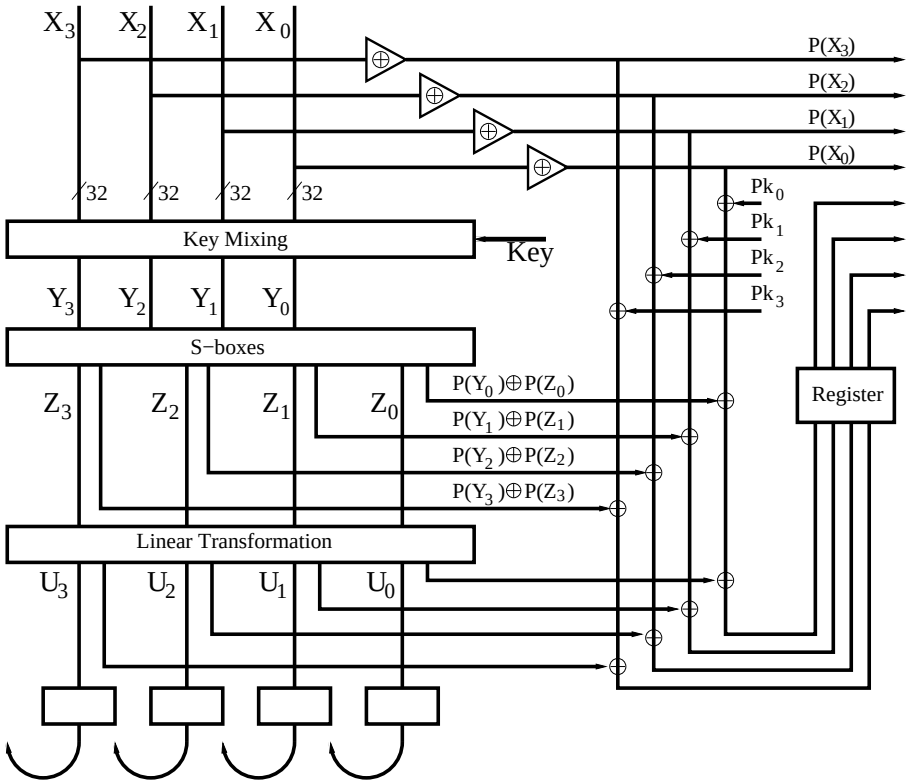
S-boxes


X0
x31...x28
x3 ...x0
S−Box 8
. . .
S−Box 1
x31⊕..⊕x28⊕y31⊕...⊕y28
x3⊕..⊕x0 ⊕ y3⊕...⊕y0
⊕
P(X0)⊕ P(Y0)
y31...y28
y3...y0
Y0

Parity-modifications for the Linear Transformation



Serpent algorithm with error detection



Error detection capability and Area overhead

- If outputs of the s-boxes are independently implemented then we can detect all single stuck-at faults
- The area overhead for the S-boxes is 8% compared to the s-boxes checked by ordinary parity.

Conclusions

- A new method of error detection (due to technical faults and due to attacs) for the Serpent Encryption Algorithm has been proposed.

- The input parity is modified according to the processing steps of the encryption algorithm into the output parity and compared with the actual output parity.

Conclusions

- the propagation of a single-bit error at the outputs of a processing step ito a multi-bit error in the sucessive processing steps , which is typical for encryption algorithms, does not reduce the error detection capability of the proposed method.