Algoritmos de Criptografia Leve

Henrique Macedo
henriquebmacedo@gmail.com

Introdução

- Dispositivos caracterizados por dimensões físicas reduzidas e por recursos computacionais limitados estão cada vez mais presentes em diversas áreas das atividades humanas.

- Estes dispositivos coletam, processam, armazenam e compartilham informações sensíveis sobre seus usuários.

- Criptografia leve é um algoritmo de criptografia adaptado para implementação em ambientes restritos, incluindo etiquetas RFID, sensores, cartões inteligentes, dispositivos de cuidados de saúde e assim por diante.

Introdução

- Um fator adicional de segurança em sistemas embarcados é que os aplicativos geralmente incluem interação direta com o mundo físico. Consequentemente, um incidente de segurança pode levar a danos em posses valiosas ou até mesmo lesões pessoais ou morte.
- O uso de técnicas criptográficas tradicionais é inviável.
- A implementação da lógica desses dispositivos é feita de 2 formas: Hardware e Software.

Introdução

- **Software**

  - Limitação na memória tanto dinâmica (RAM) quanto só de leitura (ROM, flash).

  - Preocupação com o tamanho do código.

  - Métrica: *Combined Metric* (CM). Quanto menor o CM, melhor.

$$CM = (code\ size\ [bits] * encryption\ cycle\ count\ [cycles])/block\ size\ [bits]$$

Introdução

- **Hardware**

  - Limitação de tamanho físico do circuito integrado.

  - Preocupação com consumo de energia.

  - Unidade de medida padrão: *gate equivalent* (GE).

  - Métrica: Figure of Merit (FOM). Quanto maior o valor, melhor.

$$FOM = throughput\ [Kbps]/area\ squared\ [GE^2]$$

Mecanismos de Criptografia Leve

- **Tamanho**
- **Custo**
- **Velocidade**

  - Código otimizado gera resultado mais rápido.

- **Consumo de energia**

  - O mais rápido que um conjunto de instruções for executado, mais rápido o dispositivo pode voltar para um estado de repouso ou entrar em *sleep mode*.

Criptografia Leve e Algoritmos

- Nível de segurança requerido pelas aplicações de algoritmos leves é menor se comparado com aplicações dos algoritmos tradicionais de criptografia.

- Assumem uma abordagem mais pragmática e menos conservadora, e escolhem defender seletivamente seus algoritmos apenas contra ataques considerados práticos e realistas.

Cifradores Leves de Bloco

- **PRESENT**

  - Projetado para uma implementação eficiente em hardware.

  - Tamanho de bloco: 64 bits.

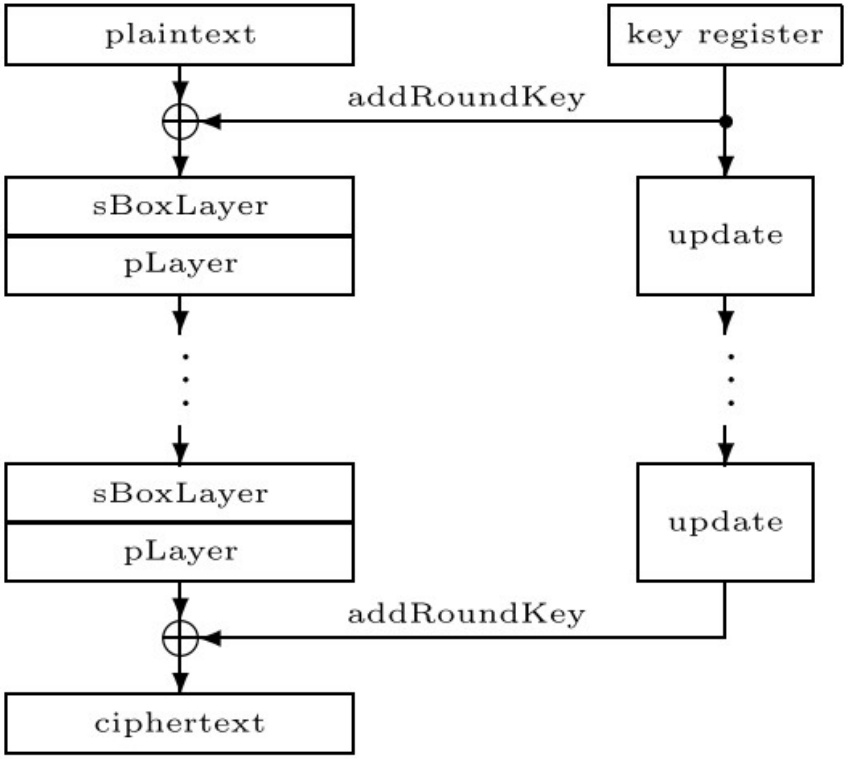
  - Tamanho das chaves: 80 ou 128 bits.

  - Eficiente em hardware e software.

  - PRESENT-80 usa 1030 GEs.

PRESENT - Algoritmo

generateRoundKeys()
for $i = 1$ to 31 **do**
  addRoundKey(STATE,K_i)
  sBoxLayer(STATE)
  pLayer(STATE)
end for
addRoundKey(STATE,K_{32})



PRESENT-80: Geração de *roundKeys*

- A chave fornecida pelo usuário é armazenada no *key register* K e representada como k79k78 . . . k0. No round i a 64-bit *round key* Ki = κ63κ62 . . .κ0 consiste dos 64 bits mais à esquerda do atual valor do registrador K. Então, num round i temos:

$$Ki = \kappa 63 \kappa 62 \ldots \kappa 0 = k79k78 \ldots k16$$

PRESENT-80: Geração de *roundKeys*

- Após extrair a round key Ki, o registrador K = k79k78 . . . k0 é atualizado conforme os seguintes passos:

1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \texttt{round\_counter}$

PRESENT – S-Box

- sBoxLayer denota o uso de uma *4-bit to 4-bit* S-Box.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

PRESENT - Permutação

- pLayer é uma permutação de bit dada pela seguinte tabela:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Cifradores Leves de Bloco

- **LED**

  - Objetivo de possibilitar uma implementação mais eficiente em hardware mas também razoavelmente eficiente em software. Porém consome alta energia por bit e é ineficiente.

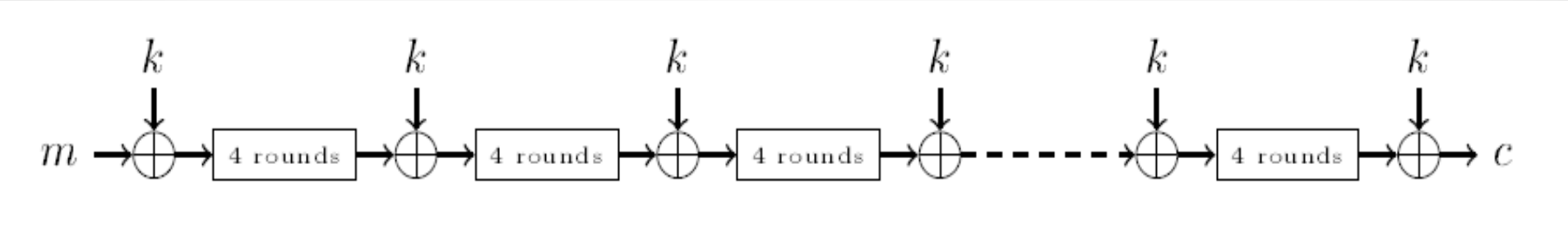
  - Bloco de 64 bits.

  - Tamanhos variáveis de chaves de 64 a 128 bits.

  - LED-80 usa 1040 GEs.

  - Já foram detectadas falhas de segurança.

LED-64



Cifradores Leves de Bloco

- **SIMON E SPECK**

  - Famílias de algoritmos ultraleves de cifragem de blocos desenvolvidos pela NSA (*National Security Agency*, E.U.A)

  - Blocos de 32, 48, 64, 96 ou 128 bits.

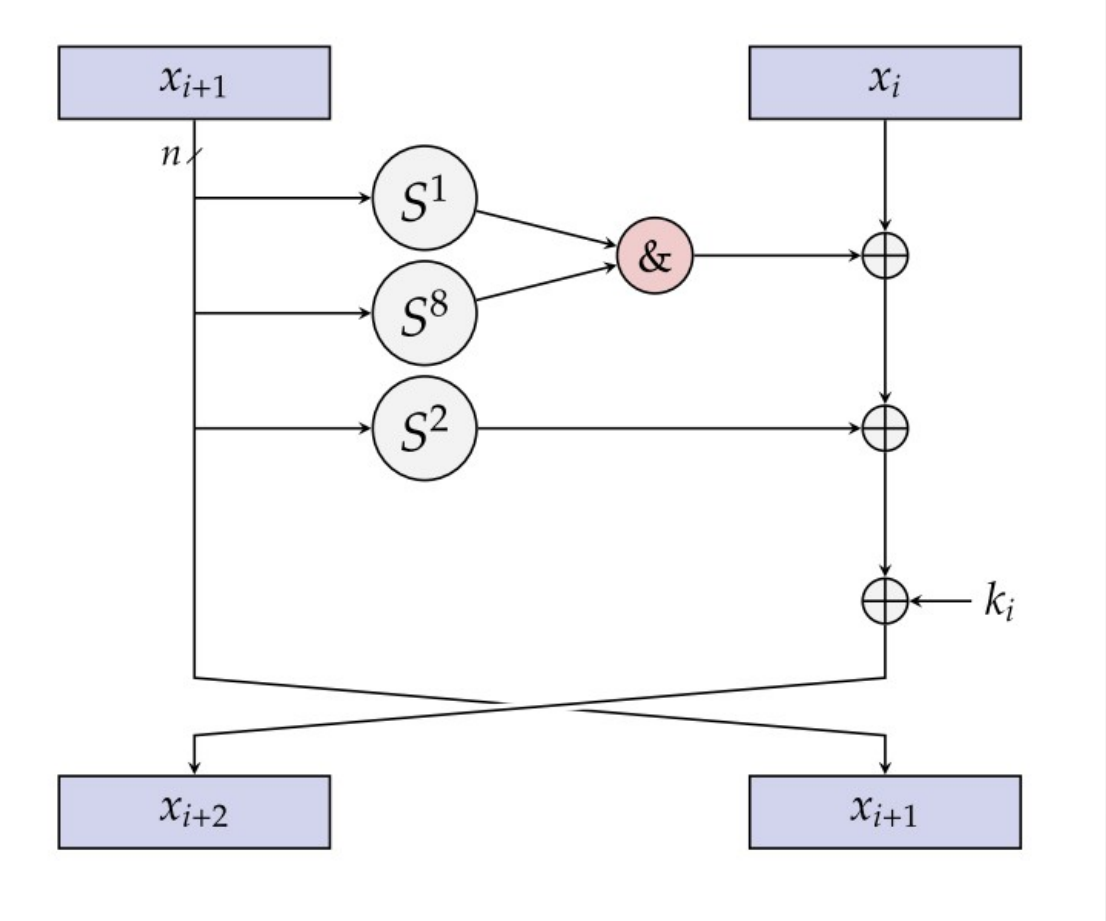
  - Para cada tamanho de bloco são suportados até 3 tamanhos de chave, que podem ser de 64, 72, 96, 128, 144, 192 ou 256 bits.

  - Implementações em hardware do SIMON e do SPECK com blocos de 64 bits e chaves de 96 bits precisam respectivamente de 838 e 984 GEs de área de circuito integrado, 35% e 41% respectivamente da área requerida pelo AES.

SIMON

block size $2n$	key size mn	word size n	key words m	const seq	rounds T
32	**64**	16	4	z_0	32
48	**72**	24	3	z_0	36
	96		4	z_1	36
64	**96**	32	3	z_2	42
	128		4	z_3	44
96	**96**	48	2	z_2	52
	144		3	z_3	54
128	**128**	64	2	z_2	68
	192		3	z_3	69
	256		4	z_4	72

SIMON – Função de Round



SIMON – Geração de Chaves

z0	11111010001001010110000111001 10
z1	10001110111110010011000010110 10
z2	10101111011100000011010010011 00 01010000100011111100101101 10011
z3	11011011101011000110010111100 00 00100100010100111001101000 01111
z4	11010001111001101011011000100 00 00101110000110010100100111 01111

SIMON – Geração de chaves

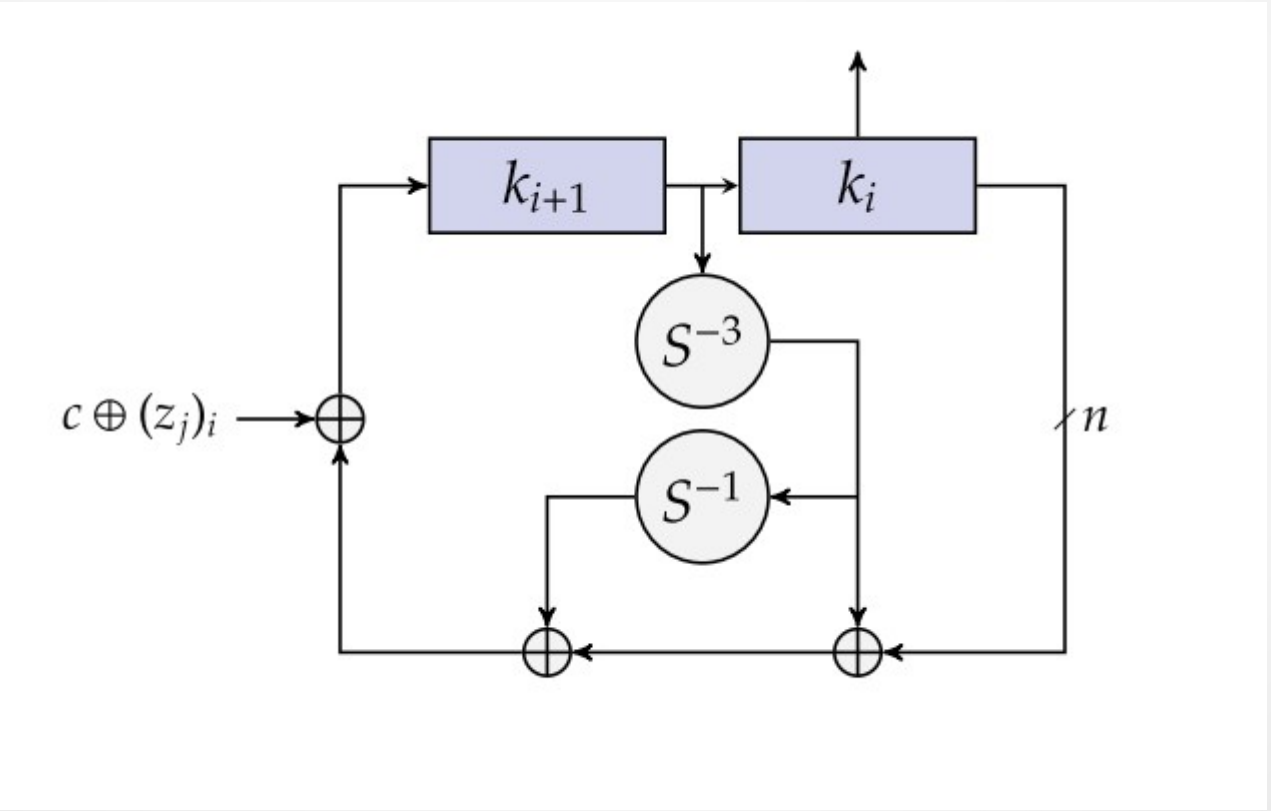
$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1}, & \text{if } m = 2, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2}, & \text{if } m = 3, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}), & \text{if } m = 4, \end{cases}$$

for $0 \leq i < T - m$

$$c = 2^n - 4 = 0\,xff\ldots fc$$

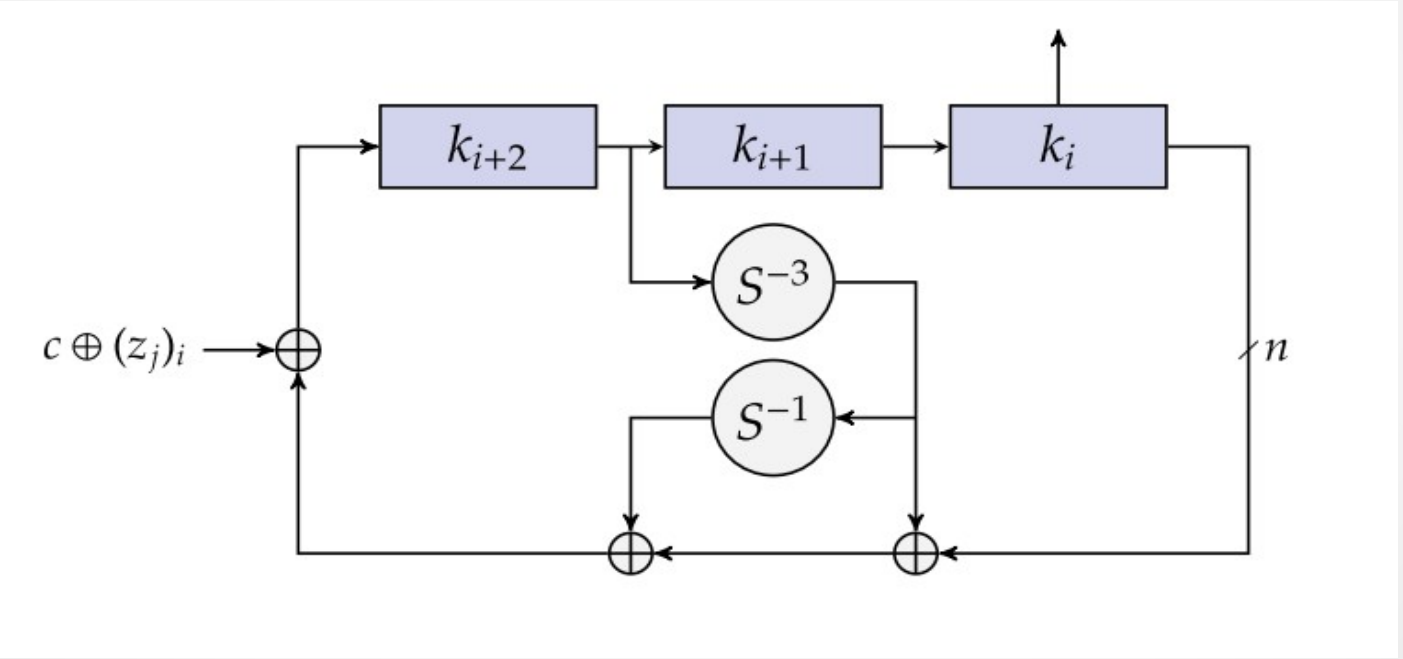
SIMON – Geração de Chaves

- m = 2



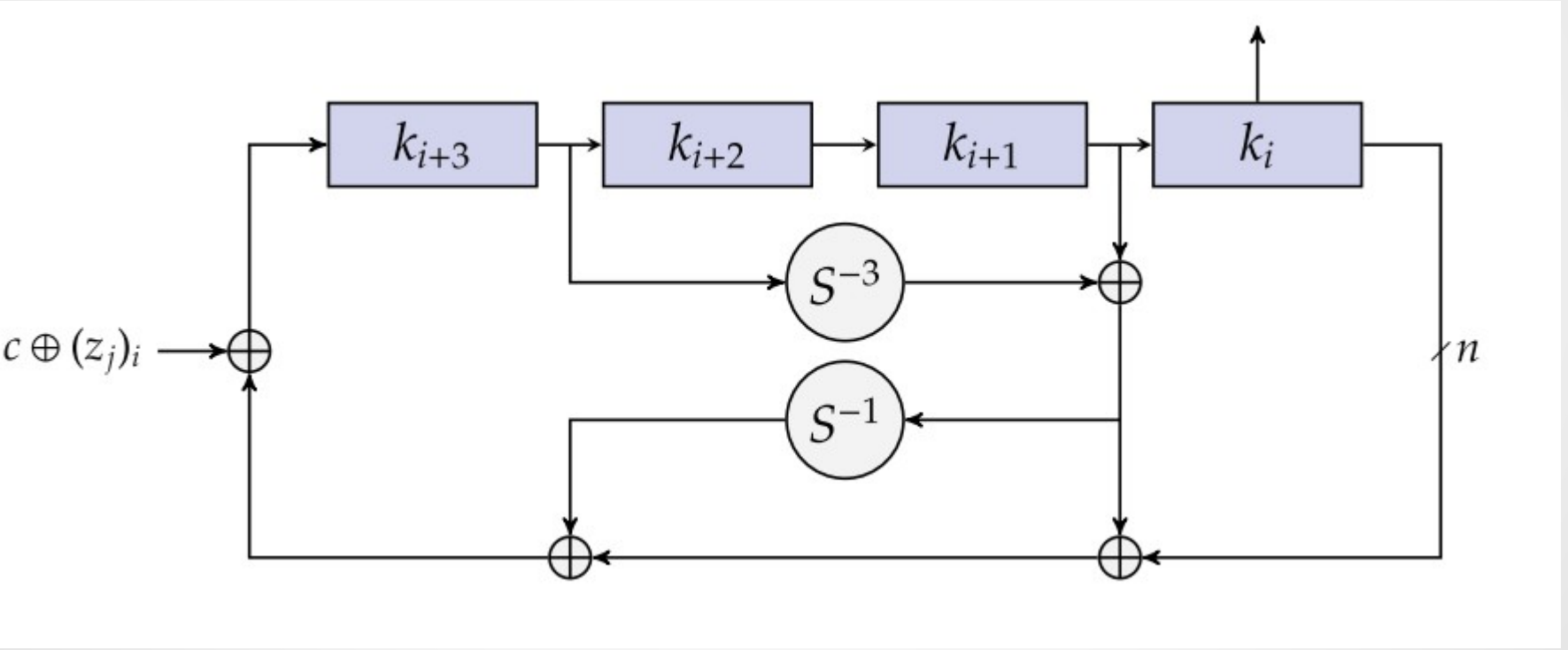
SIMON – Geração de Chaves

- m = 3

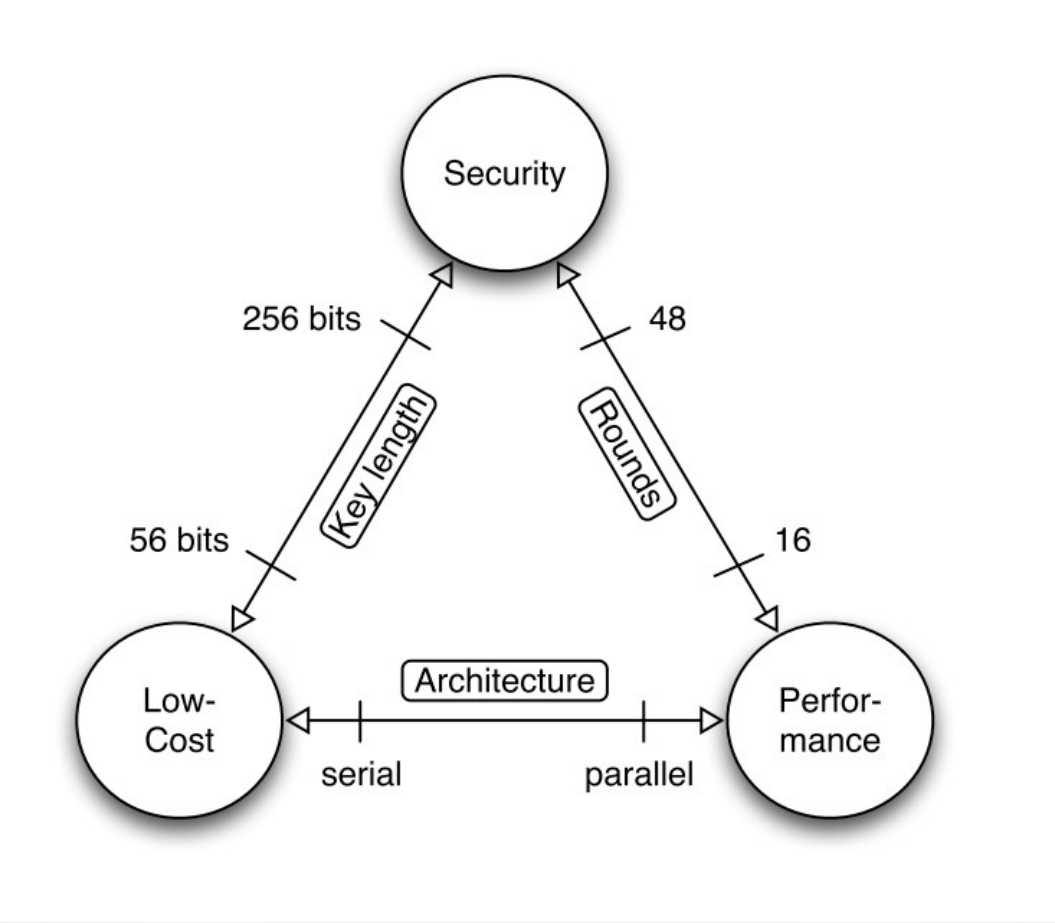


SIMON – Geração de Chaves

- m = 4



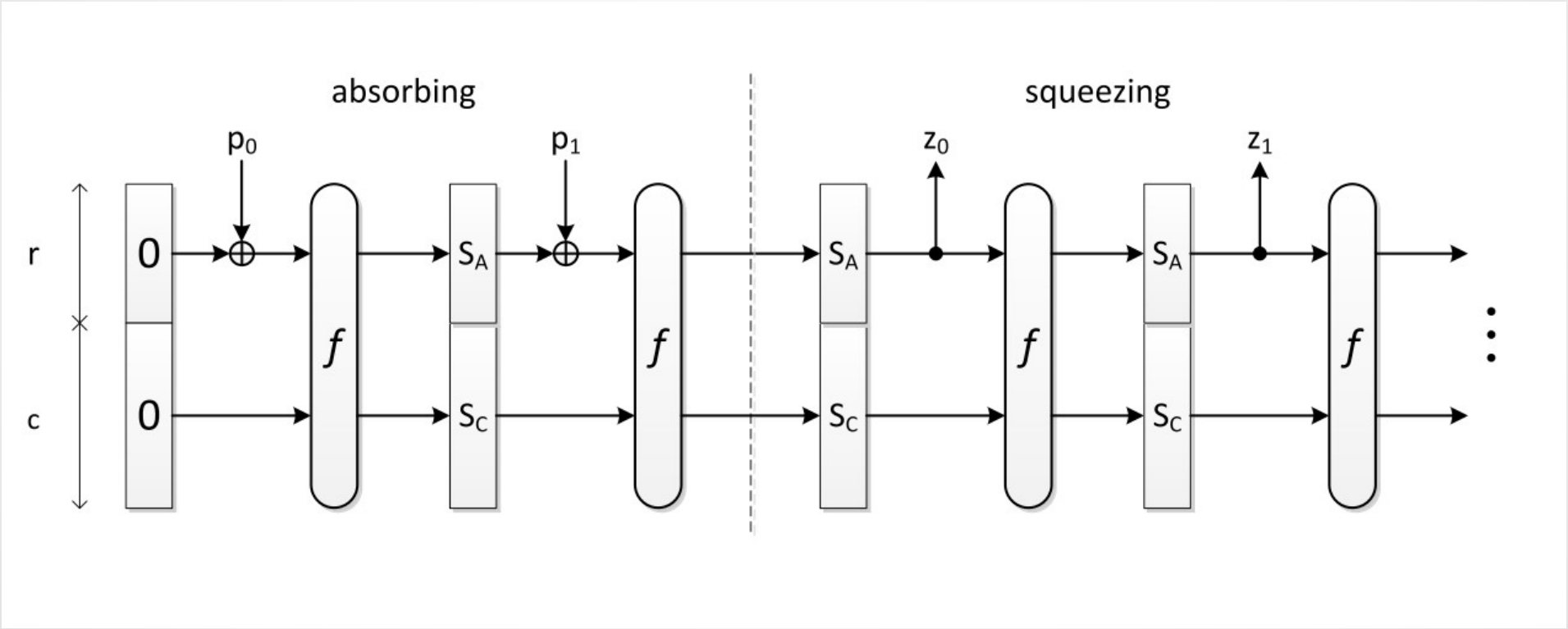
Trade-offs para Criptografia Leve



Cifradores Leves de Fluxo

- Uma alternativa para cifradores de chave simétrica.
- Inferiores aos cifradores leves de bloco.
- Ainda estão em “primeiro plano” devido à sua velocidade e simplicidade em hardware.
- Tradicionais cifradores de fluxo como: RC4, A5/1 e E0 são considerados inseguros e não devem ser usados em novas aplicações.

Algoritmos com construção esponja



Algoritmos Criptográficos Leves de Hash

- **SPONGENT**

  - Família de algoritmos leves de hash criptográfico propostos por alguns dos mesmos autores do cifrador leve de blocos PRESENT.

  - Entrada de tamanho variável da qual produzem saída de tamanho fixo (88, 128, 160, 224 ou 256 bits).

  - Resistência à colisão variando de 40 a 128 bits.

  - SPONGENT com saída de 128 bits e resistência à colisão de 64 bits requer 1060 GEs.

Variantes do SPONGENT

	n (bit)	b (bit)	c (bit)	r (bit)	R number of rounds
SPONGENT-88	88	88	80	8	45
SPONGENT-128	128	136	128	8	70
SPONGENT-160	160	176	160	16	90
SPONGENT-224	224	240	224	16	120
SPONGENT-256	256	272	256	16	140

SPONGENT – Permutação Interna

- $for\ i = 1\ to\ R\ do$
  $State \leftarrow lCounter_{reversed\ bit-order} \oplus State \oplus lCounter$
  $State \leftarrow sBoxLayer(State)$
  $State \leftarrow pLayer(State)$
- $end\ for$

SPONGENT – Permutação Interna

- *lCounter* é o estado de um LFSR (*Linear Feedback Shift Register*) dependente de b que produz uma constante de round que é adicionada ao bit mais à direita do estado.
- *lCounter reversed-bit-order* é o valor de *lCounter* com os bits em ordem reversa e sua constante de round é adicionada ao bit mais à esquerda do estado.

	LFSR bit size	Valor inicial (X)	Primitiva Polinomial
SPONGENT-88	6	000101	X^6+X^5+1
SPONGENT-128	7	1111010	X^7+X^6+1
SPONGENT-160	7	1000101	X^7+X^6+1
SPONGENT-224	7	0000001	X^7+X^6+1
SPONGENT-256	8	10011110	$X^8+X^4+X^3+X^2+1$

SPONGENT – Permutação Interna

- SboxLayer(state) denota o uso de uma *4-bit to 4-bit* S-Box.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	E	D	B	0	2	1	4	F	7	A	8	5	9	C	3	6

SPONGENT – Permutação Interna

- pLayer(state) é uma permutação de bits.

$$P_b(j) = \begin{cases} j \cdot b/4 \mod b-1, & \text{if } j \in \{0,\ldots,b-2\} \\ b-1, & \text{if } j = b-1. \end{cases}$$

Algoritmos Criptográficos Leves de Hash

- **PHOTON**

  - Família de algoritmos leves de hash criptográfico desenvolvidos por alguns dos mesmos autores do cifrador de blocos LED.

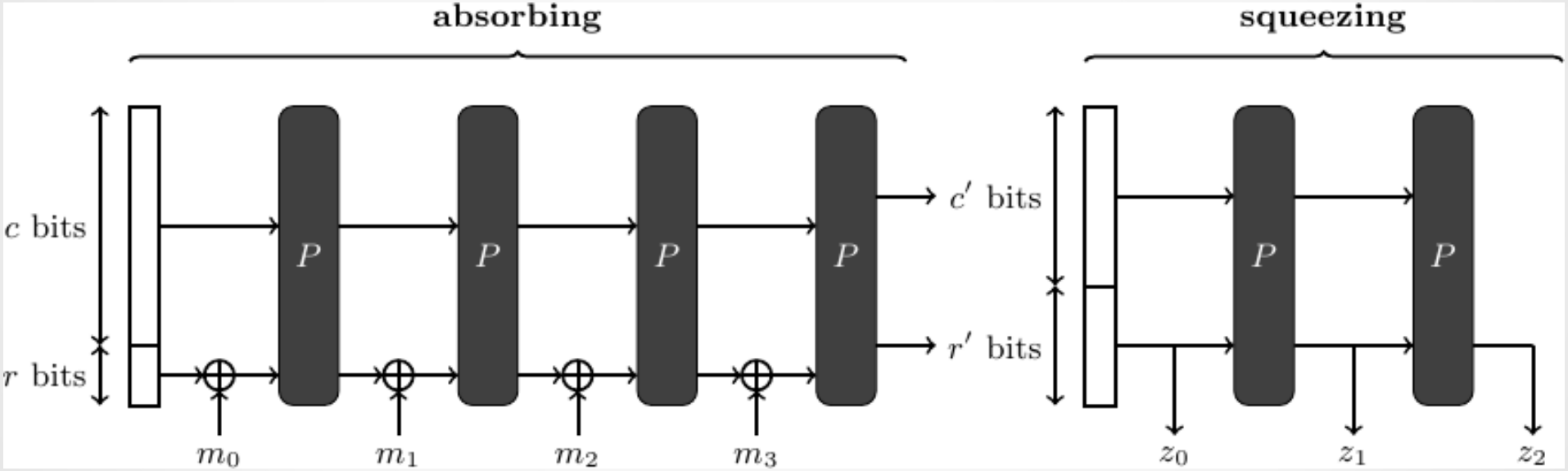
  - Sua estrutura é de uma construção esponja.

  - Geram saídas de tamanho de 80, 128, 160, 224 ou 256 bits.

  - Possuem uma resistência à colisão entre 40 e 128 bits, dependendo do tamanho da saída.

  - A implementação em hardware do PHOTON com saída de 128 bits e resistência à colisão de 64 bits, precisa de 1122 GEs.

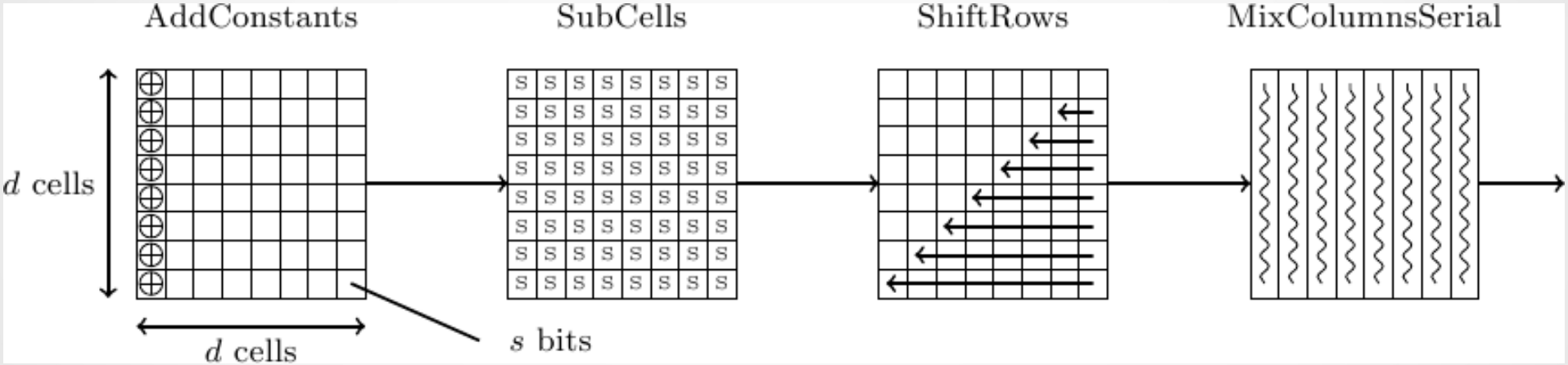
Modo de operação do PHOTON



Variantes do PHOTON

	Internal Permutation	t	n	c	r	r'	d	s
PHOTON-80/20/16	P_{100}	100	80	80	20	16	5	4
PHOTON-128/16/16	P_{144}	144	128	128	16	16	6	4
PHOTON-160/36/36	P_{196}	196	160	160	36	36	7	4
PHOTON-224/32/32	P_{256}	256	224	224	32	32	8	4
PHOTON-256/32/32	P_{288}	288	256	256	32	32	6	8

Permutação Interna do PHOTON



PHOTON - AddConstants

	t	d	s	N_r	$IC_d(\cdot)$
P_{100}	100	5	4	12	$[0, 1, 3, 6, 4]$
P_{144}	144	6	4	12	$[0, 1, 3, 7, 6, 4]$
P_{196}	196	7	4	12	$[0, 1, 2, 5, 3, 6, 4]$
P_{256}	256	8	4	12	$[0, 1, 3, 7, 15, 14, 12, 8]$
P_{288}	288	6	8	12	$[0, 1, 3, 7, 6, 4]$

$$RC(v) = [1, 3, 7, 14, 13, 11, 6, 12, 9, 2, 5, 10]$$

Fim

Obrigado!