Aula 1

GB-501: Programação em Criptografia

Operadores Lógicos

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

Operadores Lógicos

- Muitos algoritmos simétricos utilizam o modelo ARX: Addition, Rotation and XOR: *FEAL*, *Threefish*, *Salsa20*, *ChaCha*, *HC-128*, *BLAKE*, *Skein*
- Normalmente a adição é computada módulo 2^N.
- Este módulo poderá ser substituido por *and* lógico.

Operadores Lógicos

- Normalmente utilizados em criptografia simétrica e cifra de fluxo.
- Operadores rápidos: *shift* e *xor*.
- Outros operadores importantes: *and*,*or* e *not*.

Operadores Lógicos em C

- *shift left:* << (multiplicação por potência de 2)
- *shift right:* >> (divisão por potência de 2)
- *xor:* ^ (bit flip)
- *and:* &
- *or:* ||
- *not:* ~

Operadores Lógicos em C

Operações comuns:
Selecionar n bits a direita

- $(n = 4)$ 11011101 & 00001111 $=$ 00001101

Criar a máscara

- $(n = 4)$ 00001111 $=$ (1 << 4) - 1

'Flipar' todos os 8 bits

- 10101100 ^ ((1 << 8) - 1) $=$ 01010011

Operadores Lógicos em C

Exemplos:

- `void setbit( uint * v, uint n )`
- `void resetbit( uint * v, uint n )`
- `void flipbit( uint  * v, uint n )`
- `uchar parity( uint v)`

Rotate

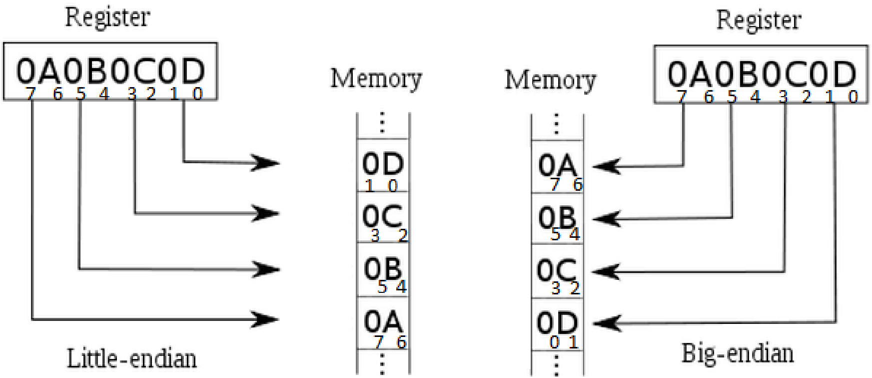
- Rotate normalmente denotado por >>> e <<<.
- Não possui operador nativo em C.
- Vamos implementar em C?

Finalista AES: Serpent

Estado: (X_3, X_2, X_1, X_0) X_i tem 32 bits

$$X_0 = X_0 <<< 13 \quad (1)$$
$$X_2 = X_2 <<< 3 \quad (2)$$
$$X_1 = X_1 \oplus X_0 \oplus X_2 \quad (3)$$
$$X_3 = X_3 \oplus X_2 \oplus (X_0 <<< 3) \quad (4)$$
$$X_1 = X_1 <<< 1 \quad (5)$$
$$X_3 = X_3 <<< 7 \quad (6)$$
$$X_0 = X_0 \oplus X_1 \oplus X_3 \quad (7)$$
$$X_2 = X_2 \oplus X_3 \oplus (X_1 <<< 7) \quad (8)$$
$$X_0 = X_0 <<< 5 \quad (9)$$
$$X_2 = X_2 <<< 22 \quad (10)$$

Little Endian vs Big Endian



Último Slide

- Perguntas?

Aula 2

GB-501: Programação em Criptografia
Cifras de Fluxo

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

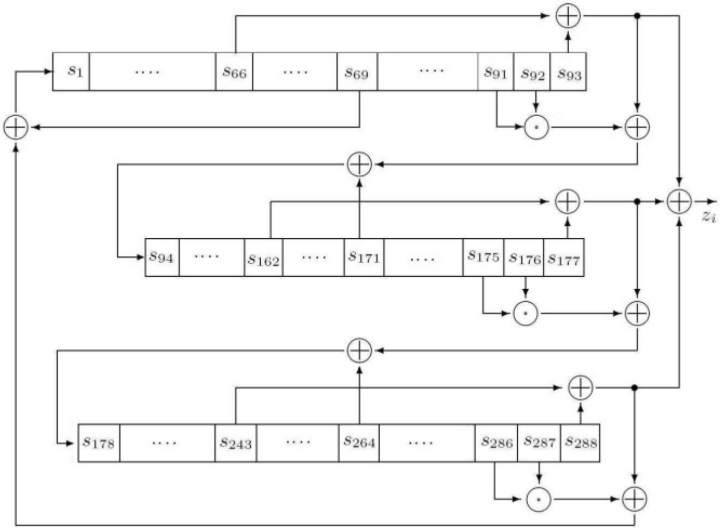
April 15, 2020

LSFR

- Método utilizado para gerar sequências pseudo aleatória.
- Extremamente rápido para implementação em *hardware*.
- Exemplos: A5/1, Grain, E0, Trivium, ...

NLSFR

Trivium



Algoritmo RC4

Key Schedule Algorithm (KSA)

```
for i from 0 to 255
  S[i] := i
endfor
j := 0
for i from 0 to 255
  j := (j + S[i] + key[i mod keysize]) mod 256
  swap values of S[i] and S[j]
endfor
```

Algoritmo RC4

Pseudo-random generation algorithm (PRGA)

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```

Aula 3

GB-501: Programação em Criptografia
Funções de Hash

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

Relembrando...

Tipo	#define	Tam. bits	Tam. bytes
unsigned char	uchar	8 bits	1 byte
unsigned short	ushort	16 bits	2 bytes
unsigned int	uint	32 bits	4 bytes
unsigned long int	ulint	64 bits	8 bytes

Relembrando...

Operador	Definição
$<<$	Shift binário à esquerda
$>>$	Shift binário à direita
$<<<$	Rotate binário à esquerda
$>>>$	Rotate binário à direita
&	AND
$\vert$	OR
$\sim$	NOT
$\oplus$	XOR
$\boxplus$	Soma módulo 2^N
$\boxminus$	Subtração módulo 2^N

Relembrando...

Uso de macros

```c
#define rotl(x,n) ((x << n) | (x >> (32-n))

uint rotl( uint x, uint x )  {
    return (x << n) | (x >> (32-n));
}

#define parity32(r, x) r = v; \
                       r = r ^ (r >> 16); \
                       r = r ^ (r >> 8); \
                       r = r ^ (r >> 4); \
                       r = r ^ (r >> 2); \
                       r = (r ^ (r >> 1)) & 1;
```

Relembrando

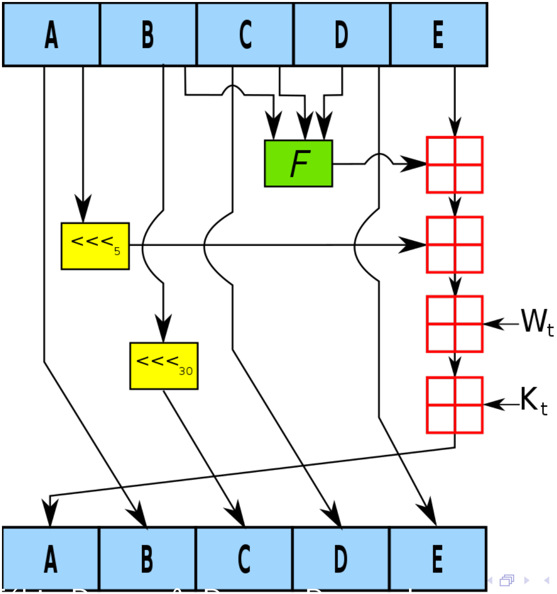
Criptografia de Fluxo

- Hadware: normalmente usa LFSR e NFSR.
  - Para implementar em software considerar o algoritmo *XorShift*.
- Software: ARX (Addiction, Rotate and XOR) veja o Salsa20.
  - Normalmente as implementações rápidas são feitas em *Assembly*.

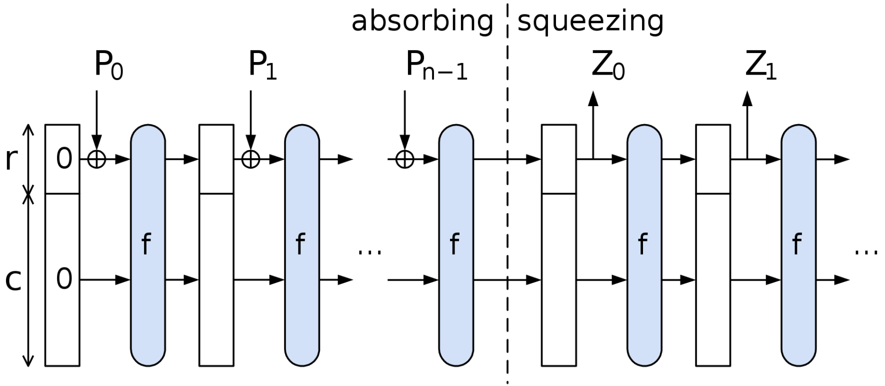
Funções de Hash

- São funções que recebem entradas de tamanhos arbitrários (até alguns GB) e possuem saídas de tamanho fixo. Por exemplo, 256 bits.
- Deve ser difícil encontrar uma entrada x dado um valor de hash h tal que $h = H(x)$.
- Deve ser difícil encontrar um valor de hash h_2 dado um valor h_1 e a entrada x tal que $h_1 = H(x)$.

SHA-1



Construção Moderna (Função Esponja)



Construção Moderna (Função Esponja)

Exemplo

- r 32 bits (1 uint: a).
- c 96 bits (3 uints: b, c, d).
- Saída 256 bits (8 aplicações de f).

```
f(a,b,c,d)
    tmp = (a ^ d) ^ (b <<< 10);
    (a,b,c,d) = (tmp,a,b,c);
```

Aula 4

GB-501: Programação em Criptografia
Corpo de Galois

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

Exemplos de Algoritmos que usam o GF

- AES
- Curvas Elípticas
- Whirpool
- HFE
- Raibow
- Isogeny
- ...

Corpo de Galois $GF(p^m)$

- Corpo Finito de ordem p^m (possui p^m elementos).
- É necessário a caracterização de um polinômio irredutível de grau m. Todas operações são feitas módulo este polinômio.

Corpo de Galois $GF(p^m)$

m	polinômios irredutíveis
1	$X+1,\ X$
2	X^2+X+1
3	$X^3+X+1,\ X^3+X^2+1$
4	$X^4+X+1,\ X^4+X^3+X^2+X+1,\ X^4+X^3+1$

Polinômios irredutíveis até o grau 4.

Exemplo GF

Representação de $GF(2^3)$

$$GF(2^3) = \{0, 1, X, X+1, X^2, X^2+1, X^2+X, X^2+X+1\}.$$

$$GF(2^3) = \{(000), (001), (010), (011), (100), (101), (110), (111)\}.$$

$$GF(2^3) = \{0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7\}$$

Multiplicação em $GF(2^m)$

Gerador de $GF(2^4)$.

$$
\begin{aligned}
(g(X))^1 &= X^2+1 &&= (0101)\\
(g(X))^2 &= X &&= (0010)\\
(g(X))^3 &= X^3+X &&= (1010)\\
(g(X))^4 &= X^2 &&= (0100)\\
(g(X))^5 &= X^2+X+1 &&= (0111)\\
(g(X))^6 &= X^3 &&= (1000)\\
(g(X))^7 &= X^3+X^2+X &&= (1110)\\
(g(X))^8 &= X+1 &&= (0011)\\
(g(X))^9 &= X^3+X^2+X+1 &&= (1111)\\
(g(X))^{10} &= X^2+X &&= (0110)\\
(g(X))^{11} &= X^3+X^2+1 &&= (1101)\\
(g(X))^{12} &= X^3+X^2 &&= (1100)\\
(g(X))^{13} &= X^3+1 &&= (1001)\\
(g(X))^{14} &= X^3+X+1 &&= (1011)\\
(g(X))^{15} &= 1 &&= (0001).
\end{aligned}
$$

Multiplicação em $GF(2^m)$

Multiplicação em $GF(2^8)$.

$$f(X) \cdot g(X) = (X^5 + X^3 + X^2 + X) \cdot (X^4 + X^2 + X + 1) =$$

$$= X^9 + 2X^7 + 2X^6 + 3X^5 + 2X^4 + 3X^3 + 2X^2 + X =$$

$$g(X) = X^9 + X^5 + X^3 + X.$$

Multiplicação em $GF(2^m)$

Seja

$$a(X) = a_0 + a_1 X + \ldots + a_{m-1} X^{m-1}$$

e

$$b(X) = b_0 + b_1 X + \ldots + b_{m-1} X^{m-1}$$

$$a(X) \cdot b(X) = a_0 b(X) + a_1 X b(b) + \ldots + a_{m-1} X^{m-1} b(X)$$

Multiplicação em $GF(2^m)$

$$a(X) \cdot b(X) = a_0 b(X) + a_1 X b(X) + \ldots + a_{m-1} X^{m-1} b(X)$$

Podemos criar um algoritmo com a seguinte ideia:

- Na iteração i podemos calcular

$$b(X) = X^i b(X) \mod i(X)$$

.

- Se $a_i = 1$ podemos acumular o valor em um polinômio $c(X) = c(X) \oplus b(X)$.

Redução modular em $GF(2^8)$.

$$
\begin{array}{l|l}
X^9 \qquad\qquad + X^5 \qquad + X^3 \qquad + X & X^8 + X^4 + X^3 + X + 1 \\
\underline{X^9 \qquad\qquad + X^5 + X^4 \qquad + X^2 + X} & \qquad\qquad\qquad\qquad X \\
\qquad\qquad\qquad X^4 + X^3 + X^2 &
\end{array}
$$

Redução modular em $GF(2^m)$

Redução modular em $GF(2^8)$.
Calcular `a(X)` mod `i(X)`.

```
while( deg(a(X)) >= m ) {
    j = deg(a(X)) - deg(i(X))
    a = a XOR (X^j * i(x))
}
return a;
```

Aula 5

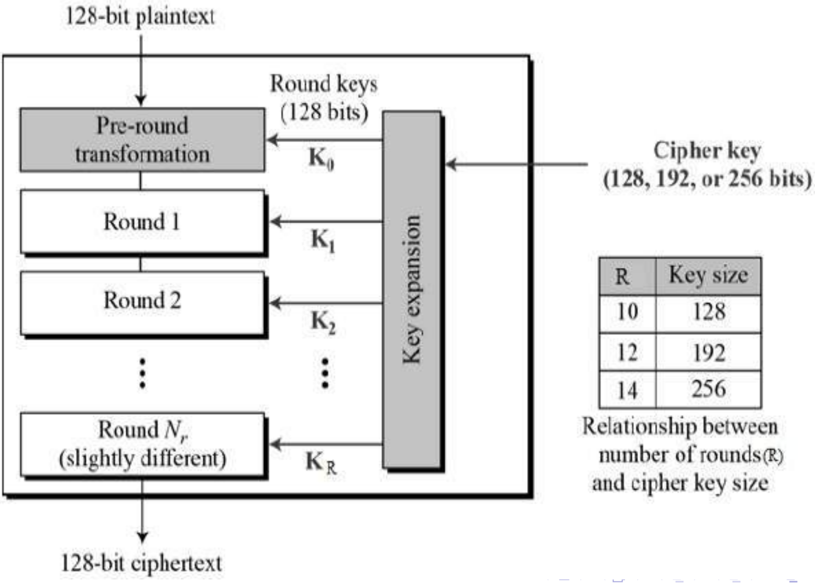
GB-501: Programação em Criptografia
Criptografia Simétrica de Bloco

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

Cifra de Bloco: AES



R	Key size
10	128
12	192
14	256

Relationship between number of rounds (R) and cipher key size

Padding

- Padding permite 'completar' a mensagem com o objetivo de tornar o tamanho da mensagem possível para o algoritmo.
- Aumentar a segurança de alguns métodos.
- O padding deve ser identificável (exceto em funções de hash)

Padding: Bit Padding

Consite completar a mensagem com um bit igual a 1 e o restante com 0. Por exemplo:

1011 1001 1101 0100 0010 011**1 0000 0000**

Padding: Byte Padding

ANSI X9.23

Consite completar a mensagem com um 1 a 8 bytes (podem ser zeros) de forma que o último byte contém o número de bytes adicionados.

01 10 55 44 11 8D D1 DD | CC 1D D0 A9 **00 00 00 04**

Padding: Byte Padding

ISO 10126

Consite completar a mensagem com um 1 a 8 bytes (bytes aleatórios) de forma que o último byte contém o número de bytes adicionados.

01 10 55 44 11 81 D1 DD | 11 1D 10 **14 7D A2 9F 05**

Padding: Byte Padding

PKCS#7 descrito na RFC 5652

Consiste completar a mensagem repetindo o tamanho do padding. O pad, neste caso pode ser:

01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
06 06 06 06 06 06
...

01 10 55 44 11 8D D1 DD | CC 1D D0 **05 05 05 05 05**

Padding: Byte Padding

ISO/IEC 7816-4:2005
Igual ao bit padding, com a diferença que o o tamanho mínimo é 1 byte

01 10 55 44 11 8D D1 DD | CC 1D D0 **80 00 00 00 00**

Padding: Byte Padding

Zero Padding
Consiste em completar a mensagem apenas com zeros. Este não é reversível. Somente usado em Hash e MACs

01 10 55 44 11 8D D1 DD | CC 1D D0 **00 00 00 00 00**

Padding

Para calcular o tamanho da mensagem com padding precisamos:

- Tamanho do Bloco em bytes: B.
- Tamanho da Mensagem em bytes: M.

O tamanho da mensagem com padding é dado pelo próximo múltiplo de B maior que M. Temos a seguinte equação para o tamanho da mensagem após o padding:

$$P = (\lfloor M/B \rfloor + 1) \cdot B$$

AES S-Box

S-Box AES

$$s = b \oplus (b \lll 1) \oplus (b \lll 2) \oplus (b \lll 3) \oplus (b \lll 4) \oplus 63_{16}$$

Inverse S-Box AES

$$b = (s \lll 1) \oplus (s \lll 3) \oplus (s \lll 6) \oplus 5_{16}$$

Key Schedule RC6

Key schedule for RC6-$w/r/b$

Input: User-supplied b byte key preloaded into the c-word array $L[0,\ldots,c-1]$
Number r of rounds

Output: w-bit round keys $S[0,\ldots,2r+3]$

Procedure: $S[0] = P_w$

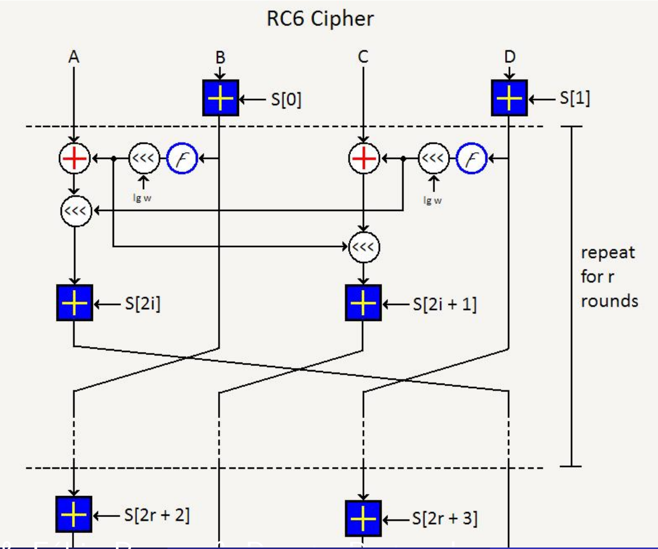
for $i = 1$ **to** $2r+3$ **do**
$\quad S[i] = S[i-1] + Q_w$

$A = B = i = j = 0$

$v = 3 \times \max\{c, 2r+4\}$
for $s = 1$ **to** v **do**
{
$\quad A = S[i] = (S[i] + A + B) \lll 3$
$\quad B = L[j] = (L[j] + A + B) \lll (A + B)$
$\quad i = (i+1) \bmod (2r+4)$
$\quad j = (j+1) \bmod c$
}

RC6

Finalista do AES



Aula 6

GB-501: Programação em Criptografia
Modos de Criptografia

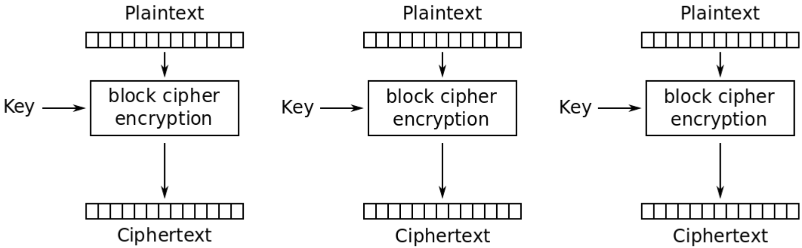
Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

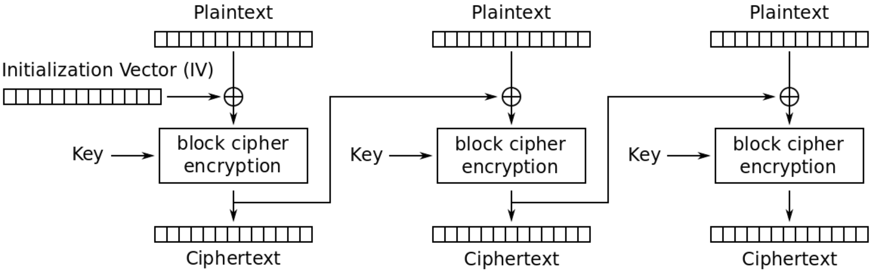
Modos de Criptografia

ECB (Electronic Code Book)



Electronic Codebook (ECB) mode encryption

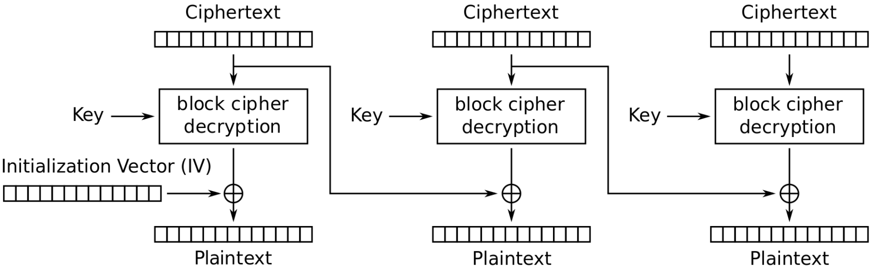
CBC (Cipher Chaining Block)



Cipher Block Chaining (CBC) mode encryption

Modos de Criptografia

CBC (Cipher Chaining Block)



Cipher Block Chaining (CBC) mode decryption

CBC (Cipher Chaining Block)
Criptografar

$$C_0 = IV$$

$$C_i = E_K(P_i \oplus C_{i-1}), i \in \{1, 2, 3, \ldots\}$$

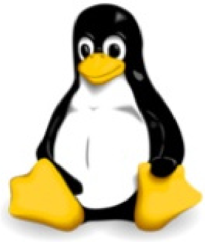
Decifrar

$$C_0 = IV.$$

$$P_i = D_K(C_i) \oplus C_{i-1}, i \in \{1, 2, 3, \ldots\}$$

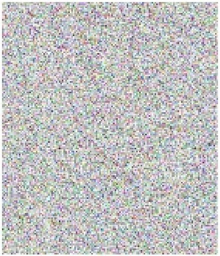
Modos de Criptografia

ECB vs CBC



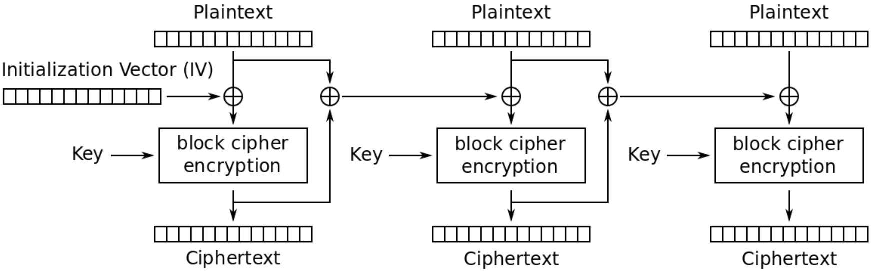
Original

ECB



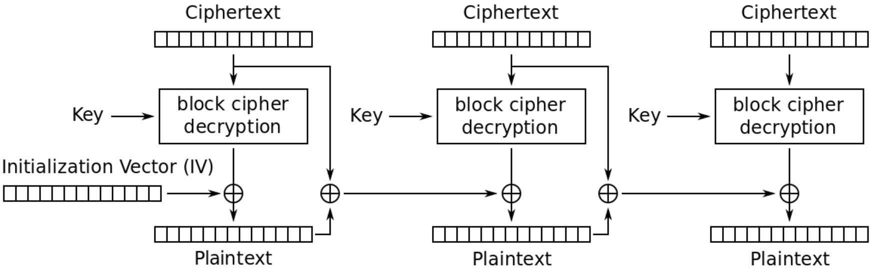
CBC

PCBC (Propagating Cipher Block Chaining)



Propagating Cipher Block Chaining (PCBC) mode encryption

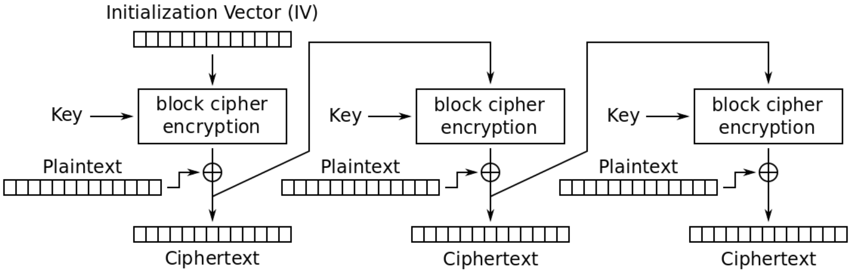
PCBC (Propagating Cipher Block Chaining)



Propagating Cipher Block Chaining (PCBC) mode decryption

Modos de Criptografia

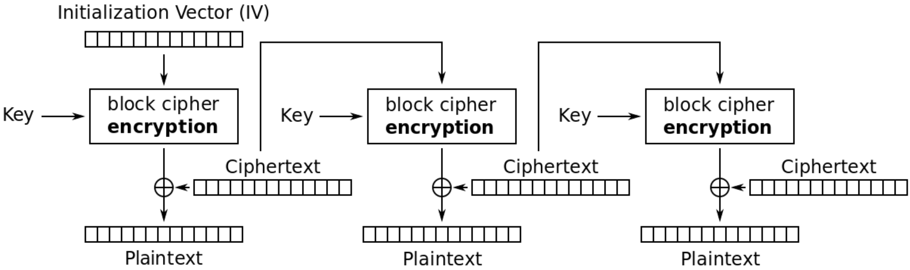
CFB (Cipher Feedback)



Cipher Feedback (CFB) mode encryption

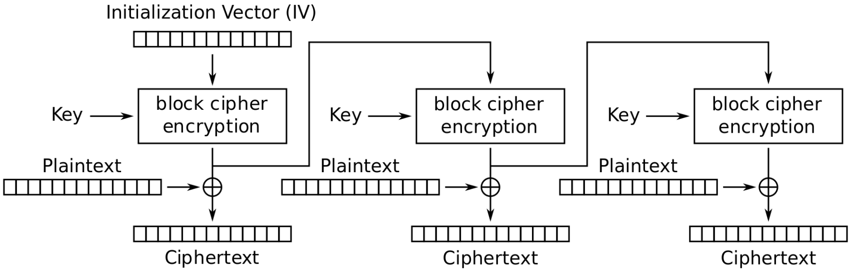
Modos de Criptografia

CFB (Cipher Feedback)



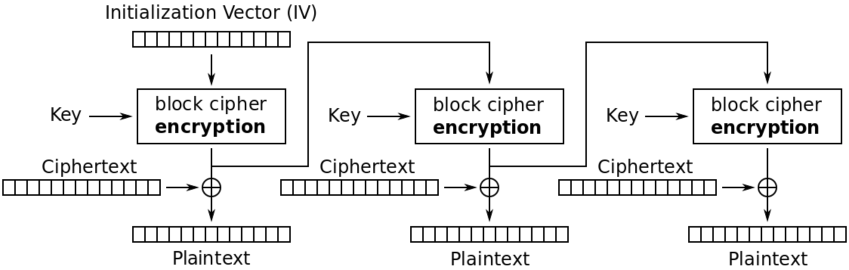
Cipher Feedback (CFB) mode decryption

OFB (Output Feedback)



Output Feedback (OFB) mode encryption

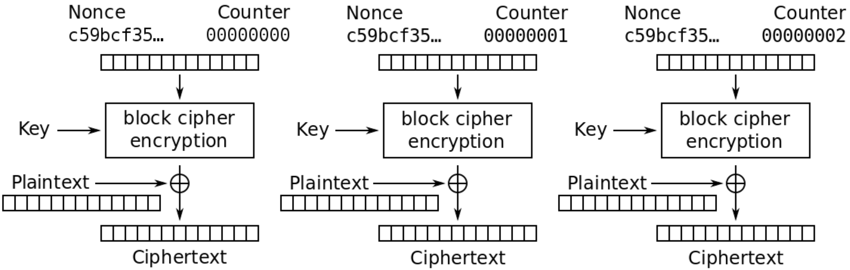
OFB (Output Feedback)



Output Feedback (OFB) mode decryption

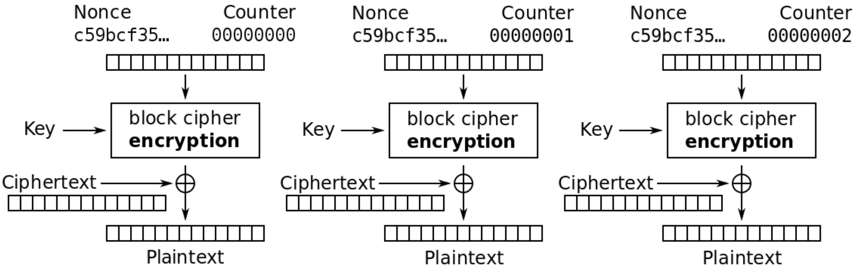
Modos de Criptografia

CTR (Counter Mode)



Counter (CTR) mode encryption

CTR (Counter Mode)



Counter (CTR) mode decryption

Aula 7

GB-501: Programação em Criptografia

Aritmética de Precisão Múltipla

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

Aritmética de Precisão Múltipla

- Muitos algoritmos assimétricos e algumas funções de *hash* utilizam aritmética em $\mathbb{Z}_m$ onde m é um número grande (por exemplo 2048 bits).
- Temos que prover mecanismos computacionais eficientes para prover a aritmética eficiente neste conjunto.

Aritmética de Precisão Múltipla

Representação
O inteiro a será representado por t elementos de W bits da seguinte forma:

$$a = (A[t-1], A[t-2], \ldots, A[2], A[1], A[0])$$

Onde $A[i]$ possui W bits (por exemplo, 64 bits)

Aritmética de Precisão Múltipla

Representação
Podemos escrever o valor de a como

$$a = 2^{(t-1)W}A[t-1] + 2^{(t-2)W}A[t-2] + \ldots + 2^{2W}A[2] + 2^{W}A[1] + A[0]$$

W normalmente é a metade da palavra do processador. Em uma máquina de 64 bits então temos $W = 32$.

Soma

Algorithm 2.5 Multiprecision addition

INPUT: Integers $a, b \in [0, 2^{Wt})$.
OUTPUT: (ε, c) where $c = a + b \bmod 2^{Wt}$ and ε is the carry bit.

1. $(\varepsilon, C[0]) \leftarrow A[0] + B[0]$.
2. For i from 1 to $t-1$ do
   2.1 $(\varepsilon, C[i]) \leftarrow A[i] + B[i] + \varepsilon$.
3. Return(ε, c).

Subtração

Algorithm 2.6 Multiprecision subtraction

INPUT: Integers $a, b \in [0, 2^{Wt})$.
OUTPUT: (ε, c) where $c = a - b \bmod 2^{Wt}$ and ε is the borrow.

1. $(\varepsilon, C[0]) \leftarrow A[0] - B[0]$.
2. For i from 1 to $t-1$ do
   2.1 $(\varepsilon, C[i]) \leftarrow A[i] - B[i] - \varepsilon$.
3. Return(ε, c).

Multiplicação

Algorithm 2.9 Integer multiplication (operand scanning form)

INPUT: Integers $a, b \in [0, p-1]$.
OUTPUT: $c = a \cdot b$.

1. Set $C[i] \leftarrow 0$ for $0 \leq i \leq t-1$.
2. For i from 0 to $t-1$ do
   - 2.1 $U \leftarrow 0$.
   - 2.2 For j from 0 to $t-1$ do:
     - $(UV) \leftarrow C[i+j] + A[i] \cdot B[j] + U$.
     - $C[i+j] \leftarrow V$.
   - 2.3 $C[i+t] \leftarrow U$.
3. Return(c).

Aritmética de Precisão Múltipla

Divisão

Input: Numerador $N = (N_{n-1} \dots N_0)_2$ e denominador D com $N > D$

Output: Resto R e quociente Q

$Q \leftarrow 0$
$R \leftarrow 0$
for $i \leftarrow n-1 \dots 0$ **do**
    $R \leftarrow R << 1$
    $R_0 \leftarrow N_i$
    if $R \geq D$ **then**
        $R \leftarrow R - D$
        $Q_i \leftarrow 1$
end
return Q, R

Aula 8

GB-501: Programação em Criptografia

Introdução à Criptografia Assimétrica

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

GMP

Gnu Multiple Precision

- Biblioteca portável para inteiros de precisão múltipla.
- Escrita em C.
- Eficiente.
- Muito utilizada em criptografia.

Diffie Hellman

- Em 1976, W. Diffie e M. Hellman propuseram uma solução para o estabelecimento de chaves criptográficas em um canal de comunicação inseguro.
- Cada usuário possui duas chaves: pública e privada.
- Para criptografar utiliza-se a chave pública do usuário remetente.
- Para decifrar utiliza-se a sua chave privada e somente essa chave pode decifrar a mensagem.

Diffie Hellman

Algoritmo 1: Etapas da Alice

$s_A \leftarrow s(\mathbb{Z}_p)$

$q_A \leftarrow g^{s_A} \mod p$

Eviar q_A para Bob.

$r_A \leftarrow q_B^{s_A} \mod p$

return r_A

Algoritmo 2: Etapas da Bob

$s_B \leftarrow s(\mathbb{Z}_p)$

$q_B \leftarrow g^{s_B} \mod p$

Eviar q_B para Alice.

$r_B \leftarrow q_A^{s_B} \mod p$

return r_B

RSA

Processo de geração de chaves.

Algoritmo 2: Geração de chaves RSA.

Saída: A chave pública (n, e) e a chave privada (n, d)

início

Escolher dois primos grandes, digamos p e q;

Calcular $n = p \cdot q$;

Calcular a função $\varphi(n) = (p-1)(q-1)$;

Escolher um inteiro e tal que $1 < e < \varphi(n)$ e $mdc(e, \varphi(n)) = 1$;

Calcular d, tal que $d \cdot e \equiv 1 \pmod{\varphi(n)}$;

retorna chave pública: (n, e), chave privada: (n, d);

fim

RSA

Processo de geração de chaves.

Algoritmo 2: Geração de chaves RSA.

Saída: A chave pública (n, e) e a chave privada (n, d)

início

- Escolher dois primos grandes, digamos p e q;
- Calcular $n = p \cdot q$;
- Calcular a função $\varphi(n) = (p-1)(q-1)$;
- Escolher um inteiro e tal que $1 < e < \varphi(n)$ e $mdc(e, \varphi(n)) = 1$;
- Calcular d, tal que $d \cdot e \equiv 1 \pmod{\varphi(n)}$;
- **retorna** chave pública: (n, e), chave privada: (n, d);

fim

RSA

Criptografia

$$c = m^e \mod n$$

Decifragem

$$m = c^d \mod n$$

Okamoto–Uchiyama

Geração de Chaves

- Generate two large primes p and q.
- Compute $n = p^2 q$.
- Choose a random integer $g \in \{2 \ldots n-1\}$ such that $g^{p-1} \not\equiv 1 \mod p^2$.
- Compute $h = g^n \bmod n$.

The public key is then(n, g, h) and the private key is (p, q).

Okamoto–Uchiyama

A message $m < p$ can be encrypted with the public key (n, g, h) as follows.

- Choose a random integer $r \in \{1 \ldots n-1\}$.
- Compute $c = g^m h^r \bmod n$.

Okamoto–Uchiyama

An encrypted message c can be decrypted with the private key (p, q) as follows.

- Compute
$$a = \frac{(c^{p-1} \bmod p^2) - 1}{p}.$$
- Compute
$$b = \frac{(g^{p-1} \bmod p^2) - 1}{p}.$$
- Compute $b' = b^{-1} \bmod p$.
- Compute $m = ab' \bmod p$.

Aula 9

GB-501: Programação em Criptografia

Criptografia Baseada em Curvas Elípticas

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

Curvas Elípticas

- Proposto independentemente por Victor S. Miller e Neal Koblitz em 1985.
- Vantagem que permite o uso de chaves menores que o RSA.
- Baseado no problema do logaritmo discreto sobre curvas elípticas (PLDCE).

Definição Uma curva elíptica E é o conjunto dos pontos (x, y) que satisfazem a seguinte lei de definição

$$y^2 = x^3 + ax + b \tag{12}$$

e mais um ponto, o ponto no infinito, denotado por ∞.

Curvas Elípticas

$$y^2 = x^3 + ax + b \tag{13}$$

- $x, y, a, b \in \mathbb{F}_q$.
- $4a^3 + 27b^2 \neq 0$.

Curvas Elípticas

Por exemplo, considere a curva $E : y^2 = x^3 + 5x + 1$ e o corpo $\mathbb{Z}_{11}$. Desta forma, a curva elíptica $E(\mathbb{Z}_{11})$ é formada pelos seguintes pontos:

$$\infty, (0,1), (0,10), (6,4), (6,7), (7,4), (7,7),$$

$$(8,5), (8,6), (9,4), (9,7).$$

Curvas Elípticas

Podemos definir uma operação $(+)$ sobre os pontos de uma curva elíptica E. O ponto no infinito ∞ atua como elemento neutro.

$$P + \infty = \infty + P = P.$$

$$P + (-P) = (-P) + P = \infty.$$

Soma entre dois pontos $P = (x_1, y_1)$ e $Q = (x_2, y_2)$, ambos pertencentes a uma curva $E(\mathbb{Z}_p)$ de forma que $P \neq \pm Q$ e seja $R = (x_r, y_r)$ onde $R = P + Q$, então

$$x_r = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$

e

$$y_r = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_r) - y_1.$$

Curvas Elípticas

Duplicação de pontos Se o objetivo é somar um ponto $P = (x_1, y_1)$ a ele mesmo, então as equações são:

$$x_r = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$$

e

$$y_r = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_r) - y_1.$$

Curvas Elípticas

Definição [PLDCE]: Dados dois pontos P e Q de uma curva elíptica, o problema consiste encontrar um inteiro k tal que $P = kQ$.

Algoritmo 3: Algoritmo Binário para Multiplicação por Escalar

Input: $k = (b_0 b_1 \ldots b_{w-1})_2 \in \mathbb{Z}^+,\ P \in E$

Output: $Q = kP$

```
Q ← ∞
for i ← 0 ... w − 1 do
    if b_i = 1 then
        Q ← Q + P
    P = 2P
end
return Q
```

Curvas Elípticas

A partir de um ponto gerador, por exemplo, P é possível gerar todos os pontos da curva elíptica E, em outras palavras

$$E : \{\infty, P, 2P, 3P, 4P, 5P, \ldots\}.$$

Curvas Elípticas - Diffie-Hellman

Algoritmo 4: Etapas da Alice

$s_A \leftarrow s(\mathbb{F}_q)$
$Q_A \leftarrow s_A P$
Eviar Q_A para Bob.
$R_A \leftarrow s_A Q_B$
return R_A

Algoritmo 5: Etapas de Bob

$s_B \leftarrow s(\mathbb{F}_q)$
$Q_B \leftarrow s_B P$
Eviar Q_A para Alice.
$R_B \leftarrow s_B Q_A$
return R_B

Aula 10

GB-501: Programação em Criptografia
Números Aleatórios

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

April 15, 2020

Números Aleatórios

Em criptografia, frequentemente precisamos de números aleatórios para:

- Chaves Criptográficas.
- NONCE.
- IV.
- Salts.
- OTP.

Números Aleatórios

NIST SP 800-90A (Dual_EC_DRBG)

Algorithm 1 Dual Elliptic Curve pseudorandom generator

Input: $s_0 \in \{0, 1, \ldots, \#\mathrm{E}(\mathbb{F}_p) - 1\}$, $k > 0$
Output: $240k$ bits
  for $i = 1$ to k **do**
    Set $s_i \leftarrow \mathrm{x}(s_{i-1}P)$
    Set $r_i \leftarrow \mathrm{lsb}_{240}(\mathrm{x}(s_iQ))$
  end for
  Return $r_1, \ldots, r_k$

The New York Times

N.S.A. Able to Foil Basic Safeguards of Privacy on Web

By Nicole Perlroth, Jeff Larson and Scott Shane

Sept. 5, 2013

1465

The National Security Agency is winning its long-running secret war on encryption, using supercomputers, technical trickery, court orders and behind-the-scenes persuasion to undermine the major tools protecting the privacy of everyday communications in the Internet age, according to newly disclosed documents.

The agency has circumvented or cracked much of the encryption, or digital scrambling, that guards global commerce and banking systems, protects sensitive data like trade secrets and medical records, and automatically secures the e-mails, Web searches, Internet chats and phone calls of Americans and others around the world, the documents show.

Números Aleatórios

Support The Guardian

Available for everyone, funded by readers

Contribute → Subscribe →

Search jobs | Sign in | Search | International edition

The Guardian

News | Opinion | Sport | Culture | Lifestyle | More

World ▸ Europe US Americas Asia Australia Middle East Africa Inequality Global development

Glenn Greenwald on security and liberty

The NSA files

This article is more than **6 years old**

Revealed: how US and UK spy agencies defeat internet privacy and security

- NSA and GCHQ unlock encryption used to protect emails, banking and medical records
- $250m-a-year US program works covertly with tech companies to insert weaknesses into products
- Security experts say programs 'undermine the fabric of the internet'
- Q&A: submit your questions for our privacy experts

Advertisement

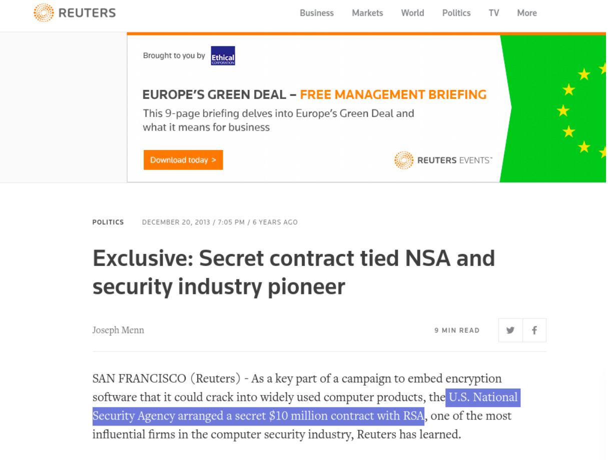
The radiation guide you've been searching for

Expert insight into radiation's effects on space, industrial and terrestrial applications in our "Radiation handbook for electronics"

Read e-book

Texas Instruments

Números Aleatórios



REUTERS

Business Markets World Politics TV More

Brought to you by Ethical

EUROPE'S GREEN DEAL – FREE MANAGEMENT BRIEFING

This 9-page briefing delves into Europe's Green Deal and what it means for business

Download today >

REUTERS EVENTS

POLITICS DECEMBER 20, 2013 / 7:05 PM / 6 YEARS AGO

Exclusive: Secret contract tied NSA and security industry pioneer

Joseph Menn 9 MIN READ

SAN FRANCISCO (Reuters) - As a key part of a campaign to embed encryption software that it could crack into widely used computer products, the U.S. National Security Agency arranged a secret $10 million contract with RSA, one of the most influential firms in the computer security industry, Reuters has learned.

Números Aleatórios

An algorithm called Dual Elliptic Curve, developed inside the agency, was on the road to approval by the National Institutes of Standards and Technology as one of four acceptable methods for generating random numbers. NIST's blessing is required for many products sold to the government and often sets a broader de facto standard.

RSA adopted the algorithm even before NIST approved it. The NSA then cited the early use of Dual Elliptic Curve inside the government to argue successfully for NIST approval, according to an official familiar with the proceedings.

RSA's contract made Dual Elliptic Curve the default option for producing random numbers in the RSA toolkit. No alarms were raised, former employees said, because the deal was handled by business leaders rather than pure technologists.

Números Aleatórios

Cryptanalysis of the Dual Elliptic Curve Pseudorandom Generator

Berry Schoenmakers and Andrey Sidorenko
Dept. of Mathematics and Computer Science, TU Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
berry@win.tue.nl, a.sidorenko@tue.nl

29 May 2006

1 Introduction

The Dual Elliptic Curve Pseudorandom Generator (DEC PRG) is proposed by Barker and Kelsey [2]. It is claimed (see Section 10.3.1 of [2]) that the pseudorandom generator is secure unless the adversary can solve the elliptic curve discrete logarithm problem (ECDLP) for the corresponding elliptic curve. The claim is supported only by an informal discussion. No security reduction is given, that is, it is not shown that an adversary that breaks the pseudorandom generator implies a solver for the ECDLP.

Números Aleatórios

IT Forum 365 | A VOZ DA TI

TODAS COLUNAS

Home > Notícias

RSA Security e Maximus oferecem segurança para governo

Editorial IT Forum 365
23/05/2011 às 21h41

Foto:

https://www.itforum365.com.br/rsa-

A Maximus deverá trabalhar com a RSA Security para treinar e certificar seus consultores para integrar o RSA SecurID (autenticação forte) e a família de smart cards, o software de autorização RSA ClearTrust e o programa de certificação digital RSA Keon. A iniciativa das empresas está relacionada ao esforço das organizações governamentais norte-americanas para proteger informações críticas, com integridade, autenticidade e controle de acesso, sistemas e transporte dos dados.

O RSA SecurID Passage smart card suporta o Departamento de Defesa (DoD) Common Acess Card (CAC), que fornece assinatura digital, autenticação de usuários e certificação baseada em logon para redes e sistemas de computador. A solução combina recursos de segurança dos smart cards com a certificação digital utilizada para acessar redes, aplicações e dados.

Números Aleatórios

- Cryptographically Secure Pseudorandom Number Generator (CSPRNG)
- Cryptographic Pseudorandom Number Generator (CPRNG).
- Cryptographic Random Number Generator (CRNG)

Números Aleatórios

Os CSPRNG pode ser divididos em 3 classes:

- Baseados em primitivas criptográficas (Cifra de Bloco, Cifra de Fluxo, Hash,...).
- Baseado em problemas matemáticos.
- Projetos de propósito específico.

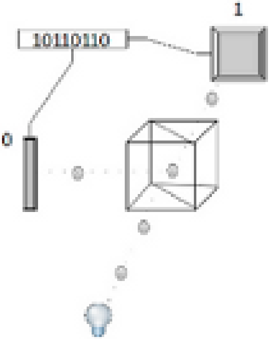
CSPRNG: Semente (seed)

- Temos que ter acesso a uma fonte de entropia.
- Linux temos o `/dev/random` que coleta uma série de ruídos do [1]computador.
- O `/dev/random` possui baixa largura de banda. Então o sistema possui o `/dev/urandom` que usa `/dev/random` como seed e gera uma sequência usando primitivas criptográficas.

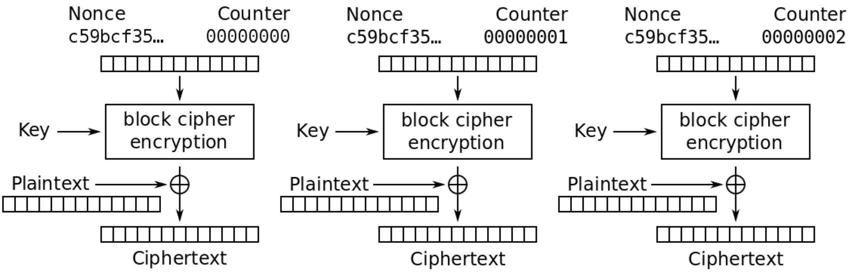
[1] Miklos Santha, Umesh V. Vazirani. "Generating quasi-random sequences from slightly-random sources" (PDF). Proceedings of the 25th IEEE Symposium on Foundations of Computer Science.

CSPRNG: Semente (seed)

IDQ Quantis



CSPRNG: Baseado em Primitivas Criptográficas



Counter (CTR) mode encryption

CSPRNG: Problemas Matemáticos

Método Blum Blum Shub

$$x_{n+1} = x_n^2 \bmod M$$
$$M = p \cdot q$$

Onde p e q são primos.
A cada iteração pode-se coletar o LSB ou paridade de x_{n+1}.
Baseado no problema do resíduo quadrático:
Definição: Dados a e $M = p \cdot q$ descobrir se existe um b tal que

$$a \equiv b^2 \mod M$$

CSPRNG: Special Designs

Microsoft CryptGenRandom

- O ID do processo atual (GetCurrentProcessID).
- O ID da Thread atual (GetCurrentThreadID).
- A contagem de ticks desde o momento da inicialização (GetTickCount).
- A hora atual (GetLocalTime).
- Vários contadores de desempenho de alta precisão (QueryPerformanceCounter).
- Um hash do bloco de ambiente do usuário, que inclui nome de usuário, nome do computador e caminho de pesquisa.
- Contadores de CPU internos de alta precisão, como RDTSC, RDMSR, RDPMC.