Aula 11

GB-501: Programação em Criptografia
Message Authentication Code

Pedro Lara & Fábio Borges & Renato Portugal

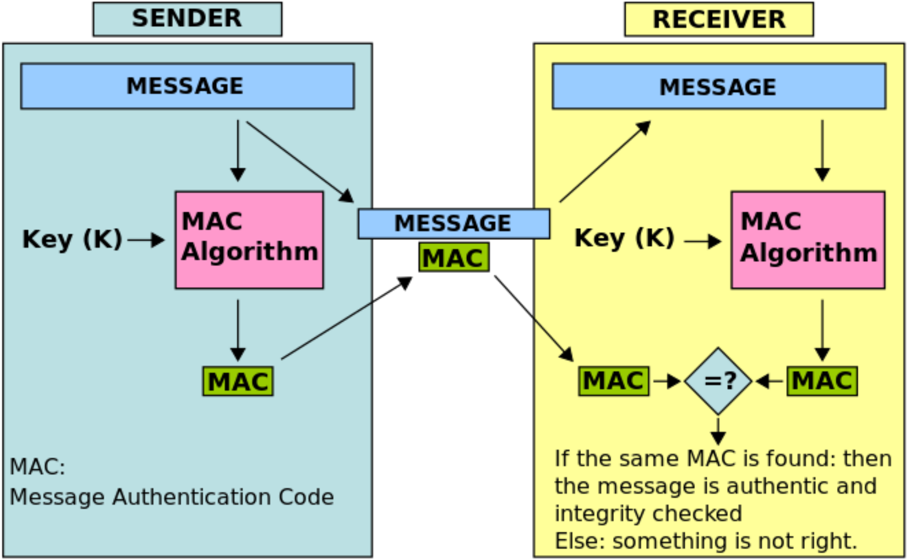
LNCC

June 3, 2020

MAC

- O MAC é utilizado para autenticar uma mensagem onde dois usuários compartilham uma chave.
- Baseados em Hash: HMAC.
- Baseados em cifra de bloco: CBC-MAC e PMAC, OMAC.
- Propósito específico: Poly1305.

MAC


SENDER
MESSAGE
Key (K)
MAC Algorithm
MAC
MAC:
Message Authentication Code
MESSAGE
MAC
RECEIVER
MESSAGE
Key (K)
MAC Algorithm
MAC
=?
MAC
If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.

HMAC

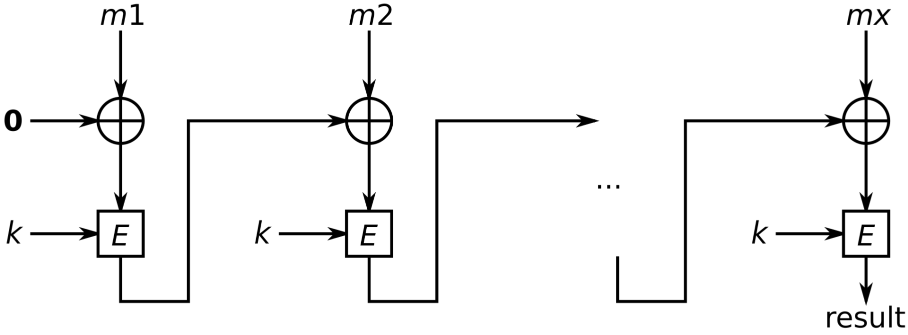
RFC 2104:

$$\mathrm{HMAC}(K, m) = \mathrm{H}\Big((K' \oplus opad) \parallel \mathrm{H}\big((K' \oplus ipad) \parallel m\big)\Big) \tag{1}$$

$$K' = \begin{cases} \mathrm{H}\left(K\right) & k \text{ é maior que o tamanho do bloco} \\ K & \text{caso contrário} \end{cases} \tag{2}$$

opad possui o tamanho do hash e consiste em repetidos 0x5c.
ipad possui o tamanho do hash e consiste em repetidos 0x36.

CBC-MAC



Poly1305

Input: Mensagem $m = m[0], m[1], \ldots, m[L-1]$, Chave $k = k[0], \ldots, k[31]$ (256 bits)

Output: MAC a com 128 bits.

$r \leftarrow \mathrm{import}(k[0], \ldots, k[15])$;

$r \leftarrow r \,\&\, \texttt{0x0ffffffc0ffffffc0ffffffc0fffffff}$;

$s \leftarrow \mathrm{import}(k[16], \ldots, k[31])$;

$a \leftarrow 0 \quad p \leftarrow 2^{130} - 5$;

for $i \leftarrow 0, \ldots \lceil L/16 \rceil$ **do**

  $n \leftarrow \mathrm{import}(m[16 \cdot i], \ldots, m[16 \cdot (i+1) - 1])$;

  $n \leftarrow n + 2^{128}$;

  $a \leftarrow a + n$;

  $a \leftarrow a \cdot r \mod p$;

end

$a \leftarrow a + s$;

return $a[0], \ldots, a[15]$

Aula 12

GB-501: Programação em Criptografia
Assinatura Digital

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Assinatura RSA

Assinatura RSA
Chave pública: (d, n).
Chave privada: (e, n).

Assinando

- Deseja-se assinar m.
- $h = H(m)$.
- $\alpha = h^e \mod n$.
- Enviar (m, α).

Verificando a assinatura

- $h^* = \alpha^d \mod n$.
- $h = H(m)$.
- Se $h = h^*$ então aceita a assinatura e rejeite, caso contrário.

Schnorr Signature

Geração de chaves

Escolha p e um g gerador de $\mathbb{Z}_p$.

- Escolha x, $1 < x < p$.
- Calcular $y = g^x \mod p$

Chave pública: y

Chave privada: x

Schnorr Signature

Assinatura

Escolha p e um g gerador de $\mathbb{Z}_p$.

- Escolha k, $1 < k < p$.
- Calcular $r = g^k \mod p$.
- Calcular $e = H(r \parallel M)$
- Calcular $s = k - xe \mod p$.

Assinatura é (s, e)

Schnorr Signature

Verificar Assinatura

- Calcular $r_v = g^s y^e \mod p$.
- Calcular $e_v = H(r_v \parallel M)$

Se $e_v = e$ então aceita a assinatura.

ECDSA

Geração de Chaves

- Escolha uma curva $E(\mathbb{Z}_p)$,
- Escolha d, $1 < d < n$.
- Escolha $G \in E(\mathbb{Z}_p)$ de ordem n.
- Calcular $Q = dG$.

Chave pública: (E, n, G, Q)
Chave privada: d

ECDSA

Assinatura

- Escolha k, $1 < k < n$.
- Calcule $W = kG = (x_1, y_1)$
- Calcule $r = x_1 \mod n$.
- Calcule $s = k^{-1}(H(m) + dr) \mod n$

(r, s) é a assinatura de m.

ECDSA

Verificar Assinatura

- Calcule $w = s^{-1} \mod n$.
- Calcule $u_1 = H(m) \cdot w \mod n$.
- Calcule $u_2 = rw \mod n$.
- Calcule $u_1 G + u_2 Q = (x_2, y_2)$.
- Calcule $v = x_2 \mod n$.

Se $v = r$ então aceita a assinatura.

Aula 13

GB-501: Programação em Criptografia

Criptografia Pós-Quântica: Multivariáveis Quadráticas

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Multivariáveis Quadráticas

Def. Seja $\mathbb{F}$ um corpo finito e $i(x) \in \mathbb{F}[x]$ um polinômio irredutível de grau n. Seja $\mathbb{E} = \mathbb{F}[x]/i(x)$ a classe de equivalência módulo o polinômio $i(x)$ então chamaremos $(E, +, \cdot)$ de "Big Field" que é uma extensão de grau n do "Single Field" $\mathbb{F}$.

Multivariáveis Quadráticas

Def. [Bijeção entre $\mathbb{E}$ e $\mathbb{F}^n$] Seja $\mathbb{E}$ uma extensão de $\mathbb{F}$ de grau n e seja $a(x) \in \mathbb{E}$ e $b \in \mathbb{F}^n$ de forma que

$$b = (b_0, \ldots, b_{n-1}), b_i \in \mathbb{F}$$

$$a(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}, a_i \in \mathbb{F}$$

Chamaremos $\phi : \mathbb{E} \rightarrow \mathbb{F}^n$ de bijeção canonica entre $\mathbb{E}$ e $\mathbb{F}^n$ definida por

$$\phi(a(x)) = b \text{ se } b_i = a_i \text{ para } i \in \{0, \ldots, n\}$$

Também temos que $\phi(\phi^{-1}(b)) = b, \forall b \in \mathbb{F}^n$ e $\phi^{-1}(\phi(a)) = a, \forall a \in \mathbb{E}$

Transformações afim

Def. Seja $M_S \in \mathbb{F}^{n\times n}$ e $v_S \in \mathbb{F}^n$ e seja $S(x) = M_S x + v_S$. Chamaremos isso de representação matricial para a transformação afim $S(x)$.

Def. Seja $s_1, \ldots, s_n$ n polinômios de grau 1 (no máximo) sobre $\mathbb{F}$, isto é, $s_i(x_1, \ldots, x_n) = \beta_{i,1}x_1 + \ldots + \beta_{i,n}x_n + \alpha_i$ com $\beta_{i,j}, \alpha_i \in \mathbb{F}$. Seja $S(x) = (s_1(x), \ldots, s_n(x))$ para um vetor $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$. Chamaremos isso de representação multivariável para a transformação afim $S(x)$.

Background da área

Definição informal

- **Função One-Way** é *fácil* dada uma entrada computar uma saída. No entanto é difícil dada um valor na imagem (saída) obter uma pré-imagem.
- **Função One-Way Trapdoor** é um tipo especial de Função one-way. Se é conhecido um SEGREDO é fácil obter uma pré-imagem.

Problema de Polinômios $\mathcal{M}$ultivariáveis $\mathcal{Q}$uadráticas

Um problema $\mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m)$ consiste em encontrar um vetor $(x_1, \dots, x_n) \in \mathbb{F}^n$ tais que o sistema de equação abaixo se satisfaça para um vetor $(y_1, \dots, y_m) \in \mathbb{F}^m$ dado.

$$\begin{aligned}
p_1(x_1, \dots, x_n) &= \sum_{1 \leq j \leq k \leq n} \gamma_{1,j,k} x_j x_k + \sum_j \beta_{1,j} x_j + \alpha_1 &= \\
p_2(x_1, \dots, x_n) &= \sum_{1 \leq j \leq k \leq n} \gamma_{1,j,k} x_j x_k + \sum_j \beta_{2,j} x_j + \alpha_2 &= \\
\vdots & \quad \vdots \\
p_m(x_1, \dots, x_n) &= \sum_{1 \leq j \leq k \leq n} \gamma_{m,j,k} x_j x_k + \sum_j \beta_{m,j} x_j + \alpha_2 &=
\end{aligned}$$

$\mathcal{MQ}$-trapdoor

- O problema $\mathcal{MQ}$ sozinho não dá pra utilizar em criptografia.
- Precisamos definir um trapdoor.
- Neste caso embutimos o trapdoor $(S, \mathcal{P}', T)$ no sistema criptográfico

$$\mathcal{P} = S \circ \mathcal{P}' \circ T$$

onde S, T são transformações afim e $\mathcal{P}'$ é um sistema $\mathcal{MQ}$

$\mathcal{MQ}$-trapdoor

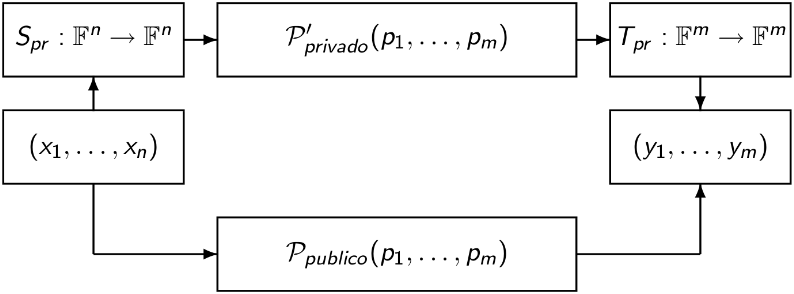


Figura: $\mathcal{MQ}$ Trapdoor

Gerar uma assinatura

- Dadas uma mensagem $y \in \mathbb{F}^m$ é necessário obter uma pré-imagem $x \in \mathbb{F}^n$ através das inversões dos componentes da chave privada.

$$x = S^{-1}(\mathcal{P}'^{-1}(T^{-1}(y)))$$

- Em geral, $\mathcal{P}'$ não é sobrejetora. Ao obter a pré-imagem x pode-se fazer uso do artifícil se se escolher aleatoriamente alguns termos. Isso depende do trapdoor utilizado.

Verificação Assinatura

Dadas uma assinatura $x \in \mathbb{F}^n$, uma mensagem $y \in \mathbb{F}^m$ e uma chave pública dada por $\mathcal{P}$ a assinatura consiste em verificar se a o vetor $x \in \mathbb{F}^n$ avaliado pelo polinômio $\mathcal{P}$ tem como resultado o vetor y.

$$
\begin{aligned}
y_1 &\stackrel{?}{=} p_1(x_1, \dots, x_n) \\
y_2 &\stackrel{?}{=} p_2(x_1, \dots, x_n) \\
&\vdots \\
y_m &\stackrel{?}{=} p_m(x_1, \dots, x_n)
\end{aligned}
$$

Complexidade: $O(n^2 m)$ operações no corpo $\mathbb{F}$.

Aula 14

GB-501: Programação em Criptografia

Criptografia Pós-Quântica: NTRU

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Anel Truncado $\mathbb{Z}_q[x]/(x^N - 1)$

NTRU é a abreviação de Nth degree-truncated polynomial ring.

- Denotamos $\mathbb{Z}_q[x]/(x^N - 1)$ de polinômios cujo os coeficientes pertencem a $\mathbb{Z}_q$.
- Os polinômios possuem grau menor que N.
- A soma e subtração são usuais.
- A multiplicação $\star$ é convolucional.
- O algoritmo de Euclides poderá ser utilizado para a inversão.

Anel Truncado $\mathbb{Z}_q[x]/(x^N - 1)$

Exemplo:

$$\mathbb{Z}_3[x]/(x^3 - 1)$$

0	x^2	$2x^2$
1	x^2+1	$2x^2+1$
2	x^2+2	$2x^2+2$
x	x^2+x	$2x^2+x$
$x+1$	x^2+x+1	$2x^2+x+1$
$x+2$	x^2+x+2	$2x^2+x+2$
$2x$	x^2+2x	$2x^2+2x$
$2x+1$	x^2+2x+1	$2x^2+2x+1$
$2x+2$	x^2+2x+2	$2x^2+2x+2$

Anel Truncado $\mathbb{Z}_q[x]/(x^N - 1)$

Multiplicação: Se $s(x) = h(x) \star f(x)$, então

$$s_k = \sum_{i+j \equiv k \mod N} h_i \cdot f_j \mod q.$$

Em

$$\mathbb{Z}_3[x]/(x^3 - 1)$$

$$s_0 = h_0 \cdot f_0 + h_2 \cdot f_1 + h_1 \cdot f_2 \mod 3$$
$$s_1 = h_1 \cdot f_0 + h_0 \cdot f_1 + h_2 \cdot f_2 \mod 3$$
$$s_2 = h_2 \cdot f_0 + h_1 \cdot f_1 + h_0 \cdot f_2 \mod 3$$

Anel Truncado $\mathbb{Z}_q[x]/(x^N - 1)$

Multiplicação no formato de matriz circulante

$$\begin{aligned} s_0 &= h_0 \cdot f_0 + h_2 \cdot f_1 + h_1 \cdot f_2 \mod 3 \\ s_1 &= h_1 \cdot f_0 + h_0 \cdot f_1 + h_2 \cdot f_2 \mod 3 \\ s_2 &= h_2 \cdot f_0 + h_1 \cdot f_1 + h_0 \cdot f_2 \mod 3 \end{aligned}$$

$$h(x) \star f(x) = \begin{bmatrix} h_0 & h_2 & h_1 \\ h_1 & h_0 & h_2 \\ h_2 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix}$$

Algoritmo 1: Geração de chaves para o NTRU

input : p, q, d_F, d_g

output : Chave privada $f(x), g(x)$ e a chave pública $h(x)$

do

$F(x) \leftarrow \mathrm{s}(\mathcal{B}(d_F))$

$f(x) \leftarrow 1 + pF(x)$

while $f(x)$ *não é inversível módulo* q;

$f_q(x) \leftarrow f^{-1}(x) \mod q$

$g(x) \leftarrow \mathrm{s}(\mathcal{B}(d_g))$

$h(x) \leftarrow f_q(x) \star pg(x) \mod q$

NTRU

Para criptografar:
$r(x) \leftarrow \mathrm{s}(\mathcal{B}(d_r))$
$e(x) \leftarrow m(x) + r(x) \star h(x) \mod q$

Para decifrar:
$a(x) \leftarrow e(x) \star f(x) \mod q$
$m(x) \leftarrow f_p(x) \star a(x) \mod p$

NTRU: versões

NTRU	q	p	$\mathcal{L}_f$	$\mathcal{L}_g$	$\mathcal{L}_m$	$\mathcal{L}_r$	F
1998	2^k	3	$L(d_f, d_f - 1)$	$L(d_g, d_g)$	L_m	$L(d_r, d_r)$	-
2001	2^k	$2 + x$	$1 + p \star F$	$\mathcal{B}(d_g)$	$\mathcal{B}$	$\mathcal{B}(d_g)$	$\mathcal{B}(d_F)$
2005	prime	2	$1 + p \star F$	$\mathcal{B}(d_g)$	$\mathcal{B}$	$\mathcal{B}(d_g)$	$\mathcal{B}(d_F)$

$$L(d_1, d_2) = \{F \in \mathbb{Z}_q[x]/(x^N - 1) : F \text{ have } d_1 \text{ coefficients equal to } 1 \text{ and } d_2 \text{ equal to } -1, \text{ remain is } 0\}.$$

$$\mathcal{B}(d) = L(d, 0)$$

Set L_m is defined as

$$L_m = \left\{ m \in \mathbb{Z}_q[x]/(x^N - 1) : m_i \in \left[-\frac{p-1}{2}, \frac{p-1}{2} \right] \right\}.$$

Aula 15

GB-501: Programação em Criptografia
Criptografia Parcialmente Homomórfica

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Criptografia Homomórfica

Objetivo: processar o dado criptografado.

$$E^{-1}(E(m_1) \odot E(m_2)) = m_1 \cdot m_2$$
$$E^{-1}(E(m_1) \oplus E(m_2)) = m_1 + m_2$$

RSA

Criptografia

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

Decifragem

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

Operação homomórfica: multiplicação.

$$c = (m_1^e) \cdot (m_2^e) \mod n = (m_1 \cdot m_2)^e \mod n$$

$$c^d \mod n = m_1 \cdot m_2 \mod n$$

Pailler

Geração de Chaves

- Escolher dois primos p e q.
- Calcular $n = pq$.
- $\lambda = \varphi(n) = (p-1)(q-1)$.
- $\mu = \lambda^{-1} \mod n$.
- $g = n + 1$.

Chave pública: (n, g).
Chave privada: (λ, μ)

Pailler

Criptografar

- A mensagem $m \in \{0, \ldots, n-1\}$.
- Escolher $r \in \{0, \ldots, n-1\}$ e $gcd(r, n) = 1$.
- $c = g^m r^n \mod n^2$.

Pailler

Decifrar

- Calcular $L = \frac{(c^\lambda \mod n^2)-1}{n}$.
- $m = L \cdot \mu \mod n$.

Pailler

Operação Homomórfica

$$(g^{m_1} r_1^n \mod n^2) \cdot (g^{m_2} r_2^n \mod n^2) =$$

$$g^{(m_1+m_2)} r_1^n r_2^n \mod n^2$$

Ao decifrar teríamos: $m_1 + m_2 \mod n$.

Pailler

Multiplicação por Constante

$$E^{-1}(E(m_1)^k \mod n^2) = k \cdot m_1 \mod n$$

.

Pailler

Integral usando 1/3 de Simpson

$$\int_{x_0}^{x_2} f(x)dx \approx \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2))$$

Aula 16

GB-501: Programação em Criptografia

Paralelização em Criptografia Simétrica

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Um pouco de OpenMP

Diretivas #pragma

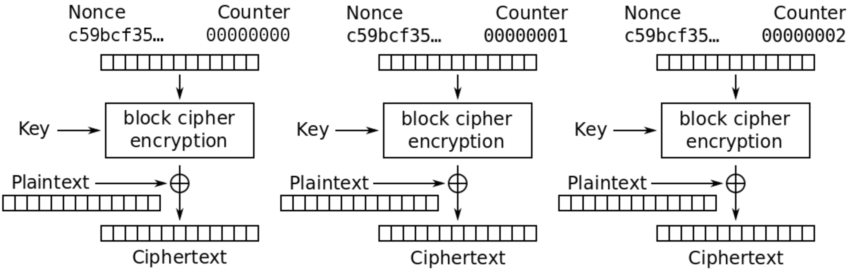
- `#pragma omp parallel`
- `#pragma omp sections`
- `#pragma omp task`
- `#pragma omp parallel for`
- `#pragma omp critical`
- `#pragma omp atomic`
- `#pragma omp barrier`
- `#pragma omp single`
- `#pragma omp master`

Um pouco de OpenMP

Algumas funções

- `omp_get_thread_num();`
- `omp_get_num_threads();`
- `omp_set_num_threads(4);`
- `omp_set_lock(&lock);`
- `omp_unset_lock(&lock);`

CTR (Counter Mode)



Counter (CTR) mode encryption

Aula 17

GB-501: Programação em Criptografia

Paralelização em Criptografia Assimétrica

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Algoritmos que Usam Exponenciação Modular

- RSA
- ElGamal
- DSA
- DH
- Damgård-Jurik
- Pailler
- MVQ
- Schnorr
- Okamoto-Uchiyama
- Blum-Goldwasser

Exponenciação Modular

Queremos calcular $a^e \mod n$

Seja

$$e = \sum_{i=0}^{w-1} b_i 2^i.$$

Desta forma, se temos que

$$a^{\sum_{i=0}^{w-1} b_i 2^i} \mod n = \prod_{i=0}^{w-1} \left(a^{b_i}\right)^{2^i} \mod n =$$

$$\prod_{i=0}^{w-1} \left(a^{2^i}\right)^{b_i} \mod n$$

Exponenciação Modular

Algoritmo 2: Algoritmo Binário de Exponenciação Modular

input : $e = (b_0 b_1 \ldots b_{w-1})_2 \in \mathbb{Z}^+$, $a \in \mathbb{Z}$ e $n \in \mathbb{Z}^+$

output : $r = a^e \mod n$

$r \leftarrow 1$
for $i \leftarrow 0 \ldots w-1$ **do**
  if $b_i = 1$ **then**
    $r \leftarrow r \cdot a \mod n$
  $a \leftarrow a^2 \mod n$
end
return r

Exponenciação Modular

Vamos supor que queremos computar $a^e \mod n$ e teremos disponíveis p processadores independentes. Então, considere a seguinte representação para o expoente

$$e = \sum_{i=0}^{p-1} 2^{i \cdot w} e_i$$

onde $k = \frac{\lfloor \log_2 e \rfloor}{p}$ (por simplicidade consideramos que $\lfloor \log_2 e \rfloor$ é múltiplo de p) e seja

$$e_i = (b_{ik} b_{ik+1} b_{ik+2} \ldots b_{(i+1)k-1})_2.$$

Exponenciação Modular

Então cada processador poderá computar, independentemente,

$$a_i = a^{2^{ik} e_i} \mod n = a^{2^{ik}} a^{e_i} \mod n$$

E ao final, bastaria calcular o produtório

$$r = a^e \mod n = \prod_{i=0}^{p-1} a_i \mod n.$$

Custo computacional

$$T(e)_{\mathrm{par}} = (p \cdot k)\, S + \left(\frac{k}{2} + \log_2 p + 1 \right) M.$$

Multiplicação por Escalar

Algoritmo 3: Multiplicação por escalar paralela.

input : $k = (k_{L-1}, k_{L-2}, \ldots, k_1, k_0)$, $w \in \mathbb{Z}^+$ e um ponto $P \in E$
output: $Q = kP$

```
do in parallel
    Q_0 ← k_0 P.
    Q_1 ← 2^w k_1 P.
    ...
    Q_{p-1} ← 2^{(p-1)w} k_{p-1} P.
end
Sincronizar.
do in parallel
    Q ← Σ_{i=0}^{p-1} Q_i
end
return Q
```

Multiplicação por Escalar

$$Q = \sum_{i=0}^{p-1} 2^{i \cdot w} k_i P.$$

Custo

$$T(k)_{\text{par}} = (p \cdot w)\, D + \left(\frac{w}{2} + \log_2 p + 1 \right) A. \tag{4}$$

Aula 18

GB-501: Programação em Criptografia

Criptoanálise em Algoritmos Simétricos

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Índice

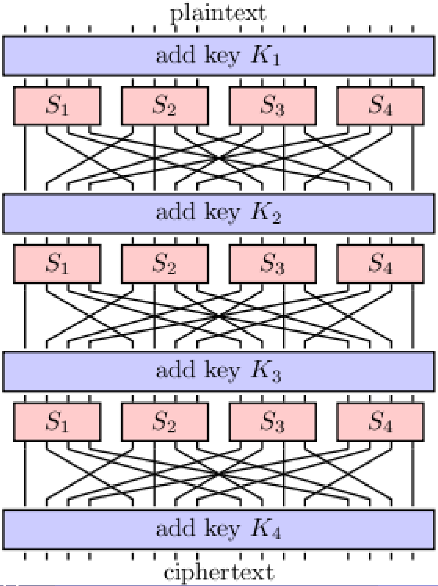
Criptoanálise: técnicas/métodos para quebrar criptografia.

- Criptoanálise Linear.
- Criptoanálise Diferencial.

Criptoanálise Linear

- Ataque do tipo texto plano conhecido.
- O objetivo é criar uma aproximação linear para a criptografia.
- Faz uso de teoria da probabilidade e estatística.

Criptoanálise Linear (SPN)



X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Y	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

SBOX hipotética.

Aproximação Linear para uma Sbox:

$$X_{i_1} \oplus X_{i_2} \oplus \ldots \oplus X_{i_n} \oplus Y_{j_1} \oplus Y_{j_2} \oplus \ldots \oplus Y_{j_m} \tag{5}$$

Queremos encontrar expressões deste tipo que tenha uma probabilidade longe de $\frac{1}{2}$ para ser 0.

Criptoanálise Linear

Def. Seja $L = X_1 \oplus \ldots \oplus X_n$ uma variável aleatória com probabilidade P_L para $L = 0$. Ou seja

$$\Pr[L = 0] = P_L.$$

Definimos o bias de L como

$$\epsilon_L = P_L - \frac{1}{2}$$

Se X_i é uniformemente distribuída em $\{0, 1\}$ então seu bias será $\epsilon_L = 0$.

Criptoanálise Linear

Piling-Up Lemma O bias de $X_1 \oplus \ldots \oplus X_n$ quando X_i são independentes é dado por

$$\epsilon = 2^{n-1} \prod_{i=1}^{n} \epsilon_i,$$

onde ϵ_i é o bias de X_i.

Criptoanálise Linear

Tabela de Bias para SBOX (fator de 1/16)

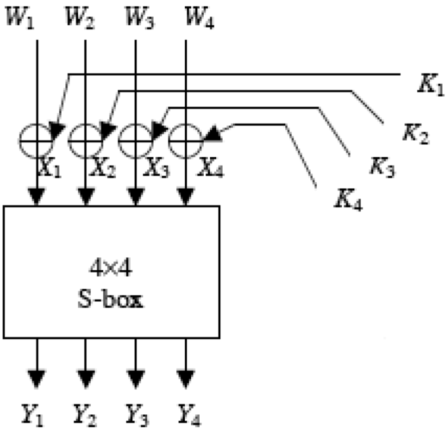
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
1	0	-2	-2	0	0	-2	6	2	2	0	0	2	2	0	0
2	0	-2	-2	0	0	-2	-2	0	0	2	2	0	0	-6	2
3	0	0	0	0	0	0	0	2	-6	-2	-2	2	2	-2	-2
4	2	0	-2	-2	-4	-2	0	0	-2	0	2	2	-4	2	0
5	-2	-2	0	-2	0	4	2	-2	0	-4	2	0	-2	-2	0
6	2	-2	4	2	0	0	2	0	-2	2	4	-2	0	0	-2
7	-2	0	2	2	-4	2	0	-2	0	2	0	4	2	0	2
8	0	0	0	0	0	0	0	-2	2	2	-2	2	-2	-2	-6
9	0	-2	-2	0	0	-2	-2	-4	0	-2	2	0	4	2	-2
A	4	-2	2	-4	0	2	-2	2	2	0	0	2	2	0	0
B	4	0	-4	4	0	4	0	0	0	0	0	0	0	0	0
C	-2	4	-2	-2	0	2	0	2	0	2	4	0	2	0	-2
D	2	2	0	-2	4	0	2	-4	-2	2	0	2	0	0	2
E	2	2	0	-2	-4	0	2	-2	0	0	-2	-4	2	-2	0
F	-2	-4	-2	-2	0	2	0	0	-2	4	-2	-2	0	2	0

Criptoanálise Linear

Por exemplo $X = 3$ e $Y = 9$ possui bias alto (-6 na tabela).
$X = 3 = 0011 = X_3 \oplus X_4$
$Y = 9 = 1001 = Y_1 \oplus Y_4$
Assim, a expressão

$$\Pr[X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 = 1] = \frac{14}{16} = 0.875$$

Criptoanálise Linear



Trecho do Algoritmo Simétrico

Criptoanálise Linear

Aproximação linear do algoritmo:

$$X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 =$$

$$W_3 \oplus K_3 \oplus W_4 \oplus K_4 \oplus Y_1 \oplus Y_4 = 1$$

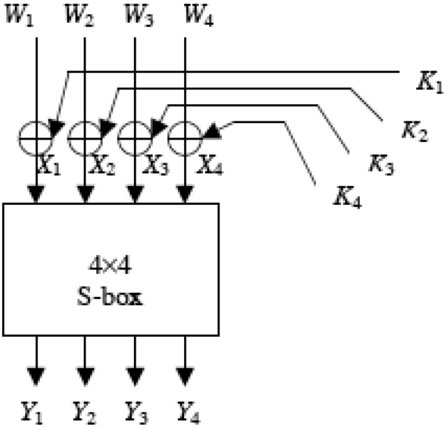
$$K_3 \oplus K_4 = W_3 \oplus W_4 \oplus Y_1 \oplus Y_4 \oplus 1$$

com probabilidade 0.875.

Criptoanálise Diferencial

- Ataque usando texto plano escolhido.
- Baseado na estatística de $(\Delta X, \Delta Y)$. Onde ΔX é uma diferença (escolhida) da entrada e ΔY é a diferença obtida na saída.
- A ideia é recuperar, inicialmente, alguns bits da última chave de rodada.

Criptoanálise Diferencial



$$\Delta X = X^1 \oplus X^2 =$$
$$= (W^1 \oplus K) \oplus (W^2 \oplus K)$$
$$= W^1 \oplus W^2 = \Delta W$$

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Y	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

SBOX hipotética.

Podemos calcular a característica diferencial para esta SBOX.

Criptoanálise Diferencial

X	Y	ΔY, $\Delta X = 1011$	ΔY, $\Delta X = 1000$	ΔY , $\Delta X = 0100$
0	E	2	D	C
1	4	2	E	B
2	D	7	B	6
3	1	2	D	9
4	2	5	7	C
5	F	F	6	B
6	B	2	B	6
7	8	D	F	9
8	3	2	D	6
9	A	7	E	3
A	6	2	B	6
B	C	2	D	B
C	5	D	7	6
D	9	2	6	3
E	0	F	B	6
F	7	5	F	B

Criptoanálise Diferencial

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Criptoanálise Diferencial

for $i \leftarrow 0, \ldots, 3$ **do**

$\quad x_1 \leftarrow x_1 \oplus k_1;\ x_2 \leftarrow x_2 \oplus k_2;\ x_3 \leftarrow x_3 \oplus k_3;\ x_4 \leftarrow x_4 \oplus k_4$

$\quad x_1 \leftarrow sbox[x_1];\ x_2 \leftarrow sbox[x_2];\ x_3 \leftarrow sbox[x_3];\ x_4 \leftarrow sbox[x_4]$

end

$x_1 \leftarrow x_1 \oplus k_1;\ x_2 \leftarrow x_2 \oplus k_2;\ x_3 \leftarrow x_3 \oplus k_3;\ x_4 \leftarrow x_4 \oplus k_4$

return x_1, x_2, x_3, x_4

Criptoanálise Diferencial

$$\Delta X = B \rightarrow \Delta Y = 2 \quad \text{Prob. } \frac{8}{16}$$

$$\Delta X = 2 \rightarrow \Delta Y = 5 \quad \text{Prob. } \frac{6}{16}$$

$$\Delta X = 5 \rightarrow \Delta Y = A \quad \text{Prob. } \frac{4}{16}$$

$$\Delta X = A \rightarrow \Delta Y = 8 \quad \text{Prob. } \frac{6}{16}$$

$$P = \frac{8}{16} \cdot \frac{6}{16} \cdot \frac{4}{16} \cdot \frac{6}{16} = 0.017578125$$

Aula 19

GB-501: Programação em Criptografia
Criptoanálise em Algoritmos Assimétricos

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Pollard ρ

- Algoritmo Pollard ρ.
- Algoritmo $p-1$.
- Algoritmo de Lenstra (ECM).
- Algoritmo BSGS.

Pollard ρ

Algoritmo 4: Fatoração usando Pollard ρ.

input : Inteiro n

output: Um fator d de n

$x \leftarrow \mathrm{Random}(0, \ldots, n-1)$

$y \leftarrow x \qquad k \leftarrow 2 \qquad i \leftarrow 0$

while True **do**

  $x \leftarrow x^2 + 1 \mod n$

  $d \leftarrow \gcd(y - x, n)$

  if $d \neq 1$ *and* $d \neq n$ **then**

    return d

  if $i = k$ **then**

    $k \leftarrow 2k$

    $y \leftarrow x$

  $i \leftarrow i + 1$

end

Algoritmo $p - 1$

Algoritmo 5: Fatoração usando $p - 1$.

input : Inteiro n e um limitante B
output: Um fator d de n

$a \leftarrow 2$
for $j = 2, \ldots, B$ **do**
    $a \leftarrow a^j \mod n$
end
$d \leftarrow gcd(a - 1, n)$
if $d \neq 1$ **then**
    return d
else
    return *Falhou*
end

Método de Lenstra

Algoritmo 6: Fatoração usando Algoritmo de Lenstra.

input : Inteiro n e um fator K

output: Um fator d de n

```
do
    Escolher a, x_0, y_0 ∈ {0, ..., n − 1}
    b = y_0^2 − x_0^3 − a x_0  mod n
    G ← (x_0, y_0)
    for i ← 0, ..., k − 1 do
        Tente calcular G = p_i G. Observe se é possível calcular a
          inversa necessária na adição e dobra. Caso não seja
          possível, verifique se gcd(d, n) ≠ 1.
    end
while True;
```

Algoritmo Baby-step Giant-step

Algoritmo 7: Logaritmo Discreto usando BSGS

input : α, β e n
output: Um valor x tal que $\alpha^x = \beta \mod n$

```
m ← ⌈√n⌉
for j = 0, ..., m − 1 do
    Crie uma tabela com (j, α^j mod n)
end
γ ← β
for i = 0, ..., m − 1 do
    if γ ∈ (j, α^j mod n) then
        return i · m + j mod n
    γ ← γ · α^(−m) mod n
end
```

Aula 20

GB-501: Programação em Criptografia

Comunicação Segura

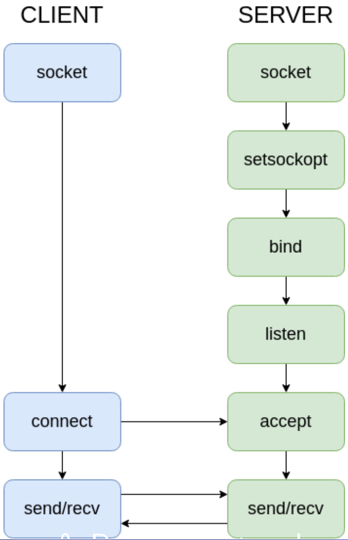
Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Sockets

Passos para trocar informação usando sockets em `C`.



Sockets

Cria um socket

`int sockfd = socket(domain, type, protocol)`

- `sockfd`: socket descriptor, assim como um file descriptor.
- `domain`: int, domínio da comunicação e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol).
- `type`: Tipo de comunicação, SOCK_STREAM: TCP SOCK_DGRAM: UDP.
- `protocol`: usar 0 para IP.

Sockets

Essa função irá permitir reutilizar um IP

```
int setsockopt(int sockfd, int level,
               int optname,
               const void *optval,
               socklen_t optlen);
```

Exemplo:

```
setsockopt(sockfd, SOL_SOCKET,
           SO_REUSEADDR | SO_REUSEPORT,
           &opt, sizeof(opt)))
```

Sockets

Associa um endereço a uma porta e a um socket

```
int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);
```

Exemplo:

```
struct sockaddr_in address;
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( 8880 );
bind(server_fd, (struct sockaddr *)&address,
                        sizeof(address));
```

Sockets

Coloca o socket em modo passivo. Ou seja vai aguardar um `connect`.

```
int listen(int sockfd, int backlog);
```

Exemplo:

```
listen(sockfd, 10);
```

Sockets

Aceita uma conexão de um cliente. Se bloqueia até receber um `connect`.

```
int new_socket= accept( int sockfd,
                        struct sockaddr *addr,
                        socklen_t *addrlen);
```

Exemplo:

```
new_socket = accept(server_fd,
                   (struct sockaddr *)&address,
                   (socklen_t*)&addrlen));
```

Sockets

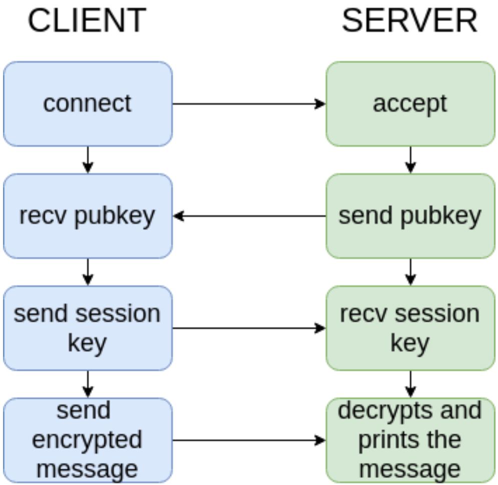
Se conecta a um servidor. O servidor deverá estar aguardando em um accept.

```
int connect(int sockfd,
            const struct sockaddr *addr,
            socklen_t addrlen);
```

Exemplo:

```
connect (sock,
        (struct sockaddr *)&serv_addr,
        sizeof(serv_addr) );
```

Protocolo



CLIENT
SERVER
connect
accept
recv pubkey
send pubkey
send session key
recv session key
send encrypted message
decrypts and prints the message

Aula 21

GB-501: Programação em Criptografia
Ofuscação de Códigos

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Ofuscação de Códigos

- Conjunto de técnicas que dificultam a legibilidade ou análise do código-fonte ou da execução de um programa.
- Assegurar que o código pareça ser diferente do que de fato é para que seja difícil entendê-lo.

Por que usar ofuscação?

- Proteger um algoritmo.
- Proteger um dado secreto dentro do programa (White Box Cryptography).

Ofuscação de Códigos

Seja P_1 um programa. Um ofuscador $O()$ leva o programa P_1 para o programa P_2 de forma que

$$P_1 \equiv P_2,$$

ou seja $\{P_1, x\} \to y$ então $\{P_1, x\} \to y$

Ofuscação de Códigos

Técnicas apresentadas em *Implementation of an Obfuscation Tool for C/C++ Source Code Protection on the XScale Architecture*

- Add Redundant Operand.
- Splits variables.
- Array folding.
- Extend Loop Conditions.

Ofuscação de Códigos

Ofuscação usando ponteiros de função.

```
int foo (void) {
    return 949;
}
int bar (void) {
    int (*fooPtr)(void);
    fooPtr = foo;
    return fooPtr();
}
```

Ofuscação de Códigos

Ofuscação usando ponteiros de função em C++.

```
int MyClass::foo(void) {
    return 310;
}
int bar (void) {
    MyClass baz;
    int (MyClass::*fooPtr)(void);
    fooPtr = &MyClass::foo;
    return (MyClass.*baz)fooPtr();
}
```

Ofuscação de Códigos

```
// C
int x;
x = 7;
x <<= 2;
x *= 2;
x -= 12;
x += (x*x)<<2;
printf("%d\n", x);
```

```
// Assembly
PUSH 1E6C
PUSH "%d\n"
CALL $PRINTF
```

Ofuscação de Códigos

```
// C
volatile int x;
x = 7;
x <<= 2;
x *= 2;
x -= 12;
x += (x*x)<<2;
printf("%d\n", x);
```

```
// Assembly
MOV [ESP],7
SHL [ESP],2
MOV EAX,[ESP]
ADD EAX,EAX
MOV [ESP],EAX
ADD [ESP],-0C
MOV ECX,[ESP]
MOV EDX,[ESP]
MOV EAX,[ESP]
IMUL ECX,EDX
...
```

Ofuscação de Códigos

```
doThis();
doThat();
doMore();
```

```
int x=2;
sw: switch(x) {
    case 0: doThat();
    x = 1;
    goto sw;
    case 1: doMore();
    break;
    case 2: doThis();
    x = 0;
    goto sw;
}
```

Ofuscação de Códigos

```
try {
    volatile int trigger=20;
    doThis();
    doThat();
    /* trigger divide-by-zero exception */
    trigger=trigger/(trigger-trigger);
    neverExecutes();
} catch (...) {
    doMore();
    doTonsMore();
}
```

Ofuscação de Códigos

Strip: remove dados não essenciais em um binário.

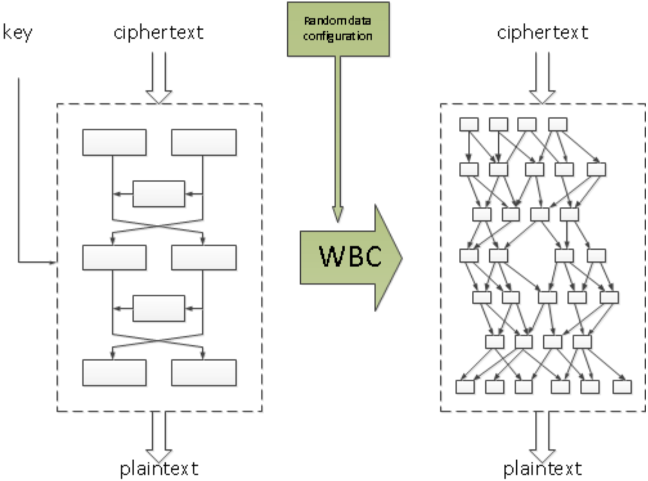
```
STRIP(1)                    GNU Development Tools                    STRIP(1)

NAME
       strip - Discard symbols from object files.

SYNOPSIS
       strip [-F bfdname |--target=bfdname]
             [-I bfdname |--input-target=bfdname]
             [-O bfdname |--output-target=bfdname]
             [-s|--strip-all]
             [-S|-g|-d|--strip-debug]
             [--strip-dwo]
             [-K symbolname|--keep-symbol=symbolname]
             [-M|--merge-notes][--no-merge-notes]
             [-N symbolname |--strip-symbol=symbolname]
             [-w|--wildcard]
             [-x|--discard-all] [-X |--discard-locals]
             [-R sectionname |--remove-section=sectionname]
             [--remove-relocations=sectionpattern]
             [-o file] [-p|--preserve-dates]
             [-D|--enable-deterministic-archives]
             [-U|--disable-deterministic-archives]
             [--keep-file-symbols]
             [--only-keep-debug]
             [-v |--verbose] [-V|--version]
             [--help] [--info]
             objfile...
```

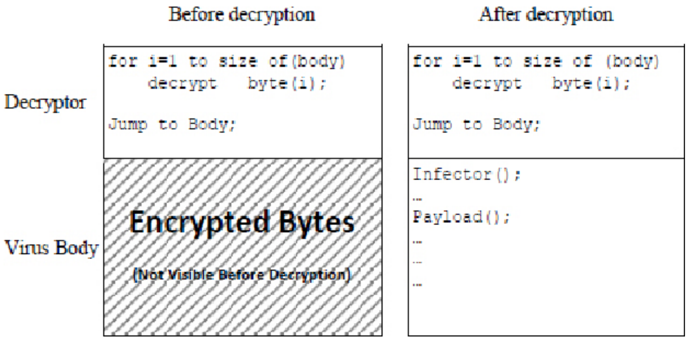
White-box cryptography

Ideia Geral: esconder uma chave privada através de implementação de criptografia ofuscada.



Codificação de Vírus

Vírus com corpo criptografado.



Aula 22

GB-501: Programação em Criptografia
Criptografia em Fortran

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Tipos de dados em Fortran 90

Os 5 tipos intrínsecos de dados são:

- Tipo Inteiro integer
- Tipo Real real
- Tipo Complexo complex
- Tipo Lógico logical
- Tipo Literal (Caractere) character

Tipos de dados (inteiros)

```
program testingInt
implicit none
   !two byte integer
   integer(kind=2) :: shortval
   !four byte integer
   integer(kind=4) :: longval
   !eight byte integer
   integer(kind=8) :: verylongval
   !sixteen byte integer
   integer(kind=16) :: veryverylongval
   !default integer
   integer :: defval
end program testingInt
```

Tipos de dados (inteiros)

Tipo literal (caractere) Por exemplo,

```
character (len=40) :: name
name = "Zara Ali"
```

Tipos de dados (ponto flutuante)

Tipo de dado real

```
! 32 bits float
real (kind=4) :: x
! 64 bits double
real (kind=8) :: y
x = 1.1234
y = 6.4332
```

Dificuldade com Operadores Lógicos em Fortran

Tem que compilar com `-fdec`.

```
INTEGER :: i, j
i = z'33'
j = z'cc'
print *, i .AND. j
```

Dificuldade para Criptografia usando Fortran

- Carência de bibliotecas para criptografia simétrica e assimétrica.
- Dificuldade para atuar com inteiros de precisão múltipla.
- Operadores lógicos e tipos sem sinal.
- Algumas formas obsoletas em LP: `ISHIFT(I, SHIFT)`.

Interface C/Fortran

Uma alternativa seria criar interfaces entre C e Fortran.

```
use, intrinsic :: iso_c_binding
implicit none

interface
double precision FUNCTION soma(a, b) bind(C)
    double precision , VALUE, INTENT(IN) :: a
    double precision , VALUE, INTENT(IN) :: b
END FUNCTION soma
end interface
```

Interface C/Fortran

Subrotinas

```
use, intrinsic :: iso_c_binding
implicit none

interface
subroutine encrypt(a, b) bind(C)
    integer, value, intent(in) :: a
    integer, value, intent(in) :: b
end subroutine encrypt
end interface
```

Interface C/Fortran

Subrotina para criptografar arquivos.

```
interface
subroutine RSA_EncryptFile (filenamein, filenamepubkey, filenameout) bind(C)
    character(len=1), dimension(*), intent(in) :: filenamein
    character(len=1), dimension(*), intent(in) :: filenamepubkey
    character(len=1), dimension(*), intent(in) :: filenameout
end subroutine RSA_EncryptFile
end interface
```

Aula 23

GB-501: Programação em Criptografia
Criptografia em Python

Pedro Lara & Fábio Borges & Renato Portugal

LNCC

June 3, 2020

Criptografia em Python

Implementa primitivas criptográficas de baixo e alto nível.

- Biblioteca `pyca/cryptography`.
- Instalação `$ pip install cryptography`
- Página `https://cryptography.io/`

Criptografia em Python: Alg. Assimétricos

Algoritmos Assimétricos

- Ed25519 signing
- X25519 key exchange
- Ed448 signing
- X448 key exchange
- Elliptic curve cryptography
- RSA
- Diffie-Hellman key exchange
- DSA

Algoritmos Simétricos

- AES
- Camellia
- ChaCha20
- TripleDES
- CAST5
- SEED
- Blowfish
- IDEA

Criptografia em Python: Modos de Criptografia

Modos de Criptografia

- ECB
- CBC
- OFB
- CTR
- GCM
- XTS

Criptografia em Python: Hash

Hash

- SHA-1
- SHA-2
- SHA-3
- BLAKE
- MD5

Criptografia em Python: MAC

MAC

- Cipher-based message authentication code (CMAC)
- Hash-based message authentication codes (HMAC)
- Poly1305