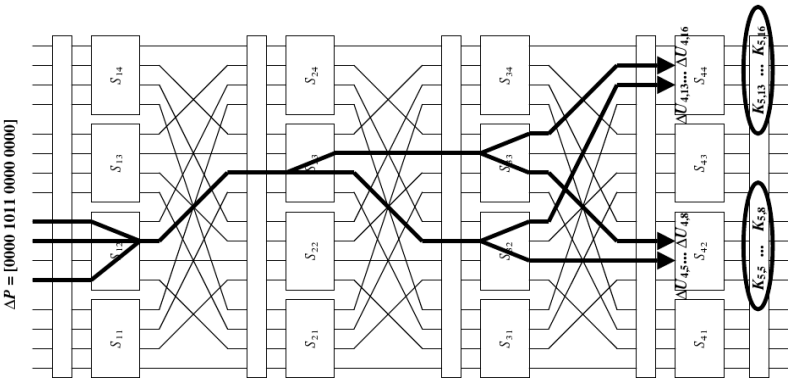
Criptoanálise Diferencial



Resistência à Criptoanálise Diferencial

cifra	complexidade na informação textos claros escolhidos	espaço complexidade	processamento complexidade
FEAL-8	2^7 pares	—	2min
FEAL-16	2^{29} pares	—	2^{30} operações
FEAL-24	2^{45} pares	—	2^{46} operações
FEAL-32	2^{66} pares	—	2^{67} operações

Função Unidireccional

Definição (Função Unidireccional)

Uma função f de um conjunto X para um conjunto Y é dita uma função unidireccional («one-way function») se $f(x)$ é «fácil de calcular» para todo o $x \in X$, mas «essencialmente para todos» os elementos $y \in \mathrm{Im}(f)$ é «computacionalmente difícil» achar um $x \in X$ tal que $f(x) = y$.

- Os termos «fácil de calcular» e «computacionalmente difícil» podem ser definidos de forma rigorosa.
- Com a utilização da frase «essencialmente para todos» pretende-se dizer que podem existir alguns elementos $y \in Y$ para os quais o cálculo de $x \in X$ tal que $y = f(x)$ é fácil, mas que no caso mais genérico tal não se verifica.

Função Unidireccional — Exemplo

Função Unidireccional

Seja $X = \{1, 2, 3, \ldots, 16\}$ e $f(x) = r_x$ para todo o $x \in X$ aonde r_x é o resto da divisão de 3^x por 17.

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$f(x)$	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1

dado um $0 \leq x \leq 16$ é fácil calcular $f(x)$, o contrário é muito mais difícil.

Função Unidireccional com Escapatória

Definição (Função Unidireccional com Escapatória)
Uma função unidireccional com escapatória («trapdoor one-way function») é uma função unidireccional $f : X \to Y$ com a propriedade de que dado algum tipo de informação adicional torna-se possível encontrar, para um dado $y \in \mathrm{Im}(f)$, um dado $x \in X$ tal que $f(x) = y$.

Função Unidireccional com Escapatória — Exemplo

Função Unidireccional
Sejam $p = 48611$ e $q = 53993$ dois números primos, $n = pq$, e $X = \{1, 2, \ldots, n-1\}$. Seja e $f(x) = r_x$ para todo o $x \in X$ aonde r_x é o resto da divisão de 3^x por n.

Calcular $f(x)$ é relativamente fácil.

Se os factores primos de n são desconhecidos e grandes o problema inverso é bastante difícil, no entanto se os factores primos de n, p e q forem conhecidos o cálculo de $y = f(x)$ pode ser feito de forma eficiente.

Cifras de Chave Pública

Seja $\{E_e : e \in \mathcal{K}\}$ um conjunto de funções de encriptação, e seja $\{D_d : d \in \mathcal{K}\}$ o correspondente conjunto de funções de desencriptação, aonde K é o espaço das chaves.

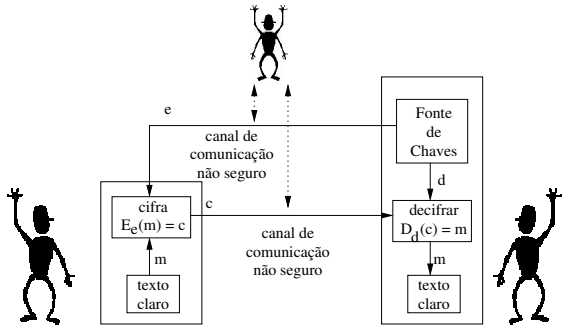
Considere-se um qualquer par de funções de encriptação/desencriptação (E_e, D_d) e suponha-se que cada um desses pares tem a propriedade de que sabendo E_e é computacionalmente intratável, dado um texto cifrado aleatório $c \in \mathcal{C}$, determinar a mensagem $m \in \mathcal{M}$ tal que $E_e(m) = c$.

Esta propriedade implica que, dado a chave de encriptação, e, é impraticável determinar a correspondente chave de desencriptação d.

E_e é então, essencialmente, uma função unidireccional com escapatória, com d a escapatória necessária ao cálculo da função inversa e como tal permitir a desencriptação.

Cifras de Chaves Públicas

Num sistema de cifra de chave pública é necessário saber a chave pública do destinatário de forma a encriptar uma mensagem a ele destinada. Só o destinatário é capaz de decifrar a mensagem.
A chave pode ser enviada por um canal não seguro.



Cifra de Chave Pública

Definição (Cifra de Chave Pública)

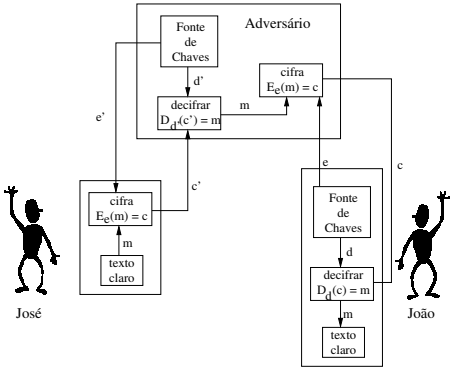
Considere-se um esquema de encriptação que consiste nos conjuntos de encriptação e desencriptação $\{E_e : e \in \mathcal{K}\}$ *e* $\{D_d : d \in \mathcal{K}\}$ *respectivamente. O esquema de encriptação é dito um esquema de encriptação de chave pública se para cada par de chaves* (e, d)*, a chave* e *(a chave pública) é* **disponibilizada publicamente**, *enquanto a outra chave,* d*, (a chave privada) é* **mantida secreta**. *Para que este esquema possa ser considerado seguro é necessário que seja* **computacionalmente intratável** *calcular* d *a partir de* e.

Nota (chave privada vs chave secreta)

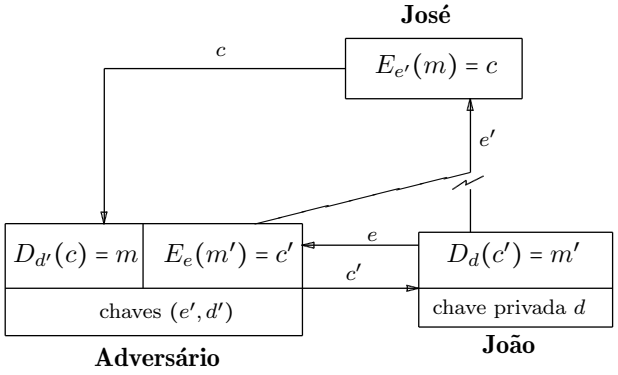
Para evitar ambiguidades convenciona-se que num esquema de cifra de chave pública se designa a chave a manter secreta por chave privada, *reservando-se o termo* chave secreta *para os sistemas de cifra de chaves simétricas.*

Autenticação de Chaves Públicas

Aparentemente os sistemas de chave pública resolvem completamente o problema da troca de chaves entre entidades. Infelizmente isso não é totalmente correcto dado que estes sistemas permitem um tipo de ataque designado por **personificação**, o qual permite quebrar o protocolo.



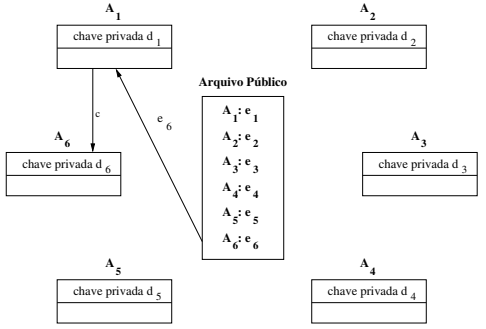
Manutenção de Chaves Públicas



O adversário substitue a chave e por uma nova chave pública e'. A partir desta modificação só o adversário pode desencriptar as mensagens para o João. Se o José tentar enviar uma mensagem para o João esta pode ser lida/alterada pelo adversário.

Manutenção de Chaves Públicas

Numa rede de chaves públicas cada entidade tem um par (chave pública,chave privada). Para assegurar um mecanismo de manutenção de chaves basta criar um repositório de chaves, usualmente designado por *Arquivo Público*.



Na presença de um adversário activo (que possa alterar o arquivo público) o mecanismo pode ser comprometido.

Chaves Simétricas e Chaves Públicas

Os actuais sistemas de criptografia podem combinar o melhor de cada um dos sistemas, por exemplo: os sistemas de chaves simétricas são muito eficientes, mas têm um manuseamento de chaves um pouco delicado, então:

- usa-se a componente de chave pública para a troca de chaves simétricas;
- usa-se a componente de chaves simétricas para efectuar a troca de informação.

O ataque mais eficiente às cifras de chaves simétricas é (para as actuais cifras) o método exaustivo, como tal, num sistema deste tipo basta a chave ser da ordem dos 64, ou 128bits.
Por outro lado as cifras de chave públicas actuais estão sempre sujeitas a forma mais eficientes de ataque, por exemplo a factorização de números primos no caso da cifra RSA, isso obriga a que a grandeza da chave tenha que ser da ordem dos 1024 bits.

Modos de Operação

A cifras de chaves públicas (por blocos) têm um modo de funcionamento semelhante às cifras de chaves secretas (por blocos), é assim necessário fazer a divisão da mensagem em blocos com o eventual preenchimento do último bloco de forma a obter-se um múltiplo do comprimento do bloco.

Os modos **CFB** e **OFB** não podem ser usados dado que usam a função de encriptação tanto para o processo de cifragem como para a decifragem. Os modos **ECB** e **CBC** podem ser usados

Menor Múltiplo Comum

Definição (Menor Múltiplo Comum)

Dados $a, b \in \mathbb{Z}\backslash\{0\}$, um inteiro não nulo c é um múltiplo comum de a e b se $a|c$ e $b|c$.

Sejam $a, b \in \mathbb{Z}\backslash\{0\}$ e considere-se $\mathcal{M} = \{c \in \mathbb{N} : a|c \wedge b|c\}$.

$\mathcal{M} \neq \varnothing$ porque $|ab| \in \mathcal{M}$. Além disso, $\mathcal{M} \subseteq \mathbb{N}$. Então $\mathcal{M}$ tem um mínimo ao qual se chama menor múltiplo comum de a e b. Esse mínimo é representado por $\mathrm{mmc}(a, b)$.

Exemplo:
Múltiplos positivos de 18: 18, 36, 54, 72, 90, 108, 126, 144, ...
Múltiplos positivos de 21: 21, 42, 63, 84, 105, 126, ...
Então $\mathrm{mmc}(18, 21) = 126$.

MMC, Propriedades

Para quaisquer $a, b \in \mathbb{Z}\backslash\{0\}$, tem-se:

1. $\mathrm{mmc}(a, b)$ é o único múltiplo comum, positivo, de a e b tal que:
$$\begin{cases} x \in \mathbb{Z} \\ a|x \\ b|x \end{cases} \Rightarrow \mathrm{mmc}(a, b)|x;$$
2. Se $n \in \mathbb{N}$ é um divisor comum de a e b então
$$\mathrm{mmc}\left(\frac{a}{n}, \frac{b}{n}\right) = \frac{\mathrm{mmc}(a, b)}{n};$$
3. $\mathrm{mmc}(a, b) \times \mathrm{mdc}(a, b) = |ab|$.

O menor múltiplo comum de $a, b \in \mathbb{Z}\backslash 0$ pode ser calculado usando o algoritmo de Euclides e a propriedade 3.

MDC de mais do que dois inteiros

$n \in \mathbb{N}, n \geq 2, a_1, a_2, \ldots, a_n \in \mathbb{Z}$ não todos nulos. O máximo divisor comum de $a_1, a_2, \ldots, a_n$ é o maior dos divisores comuns positivos de $a_1, a_2, \ldots, a_n$. Representa-se por $\mathrm{mdc}(a_1, a_2, \ldots, a_n)$

Propriedades:

1. $\mathrm{mdc}(a_1, a_2, \ldots, a_n)$ é o menor inteiro positivo da forma $a_1x_1 + a_2x_2 + \cdots + a_nx_n$, com $x_1, x_2, \ldots, x_n \in \mathbb{Z}$;
2. $\mathrm{mdc}(a_1, a_2, \ldots, a_n)$ é o único divisor comum, positivo, de $a_1, a_2, \ldots, a_n$ que é múltiplo de qualquer divisor comum de $a_1, a_2, \ldots, a_n$;
3. $\mathrm{mdc}(a_1, a_2, \ldots, a_n) = \mathrm{mdc}(\mathrm{mdc}(a_1, a_2, \ldots, a_{n-1}), a_n)$.

Inteiros Primos entre si e Inteiros Primos 2 a 2

Os inteiros $a_1, a_2, \ldots, a_n$ são primos entre si se $\mathrm{mdc}(a_1, a_2, \ldots, a_n) = 1$.

Os inteiros $a_1, a_2, \ldots, a_n$ são primos dois a dois se $\mathrm{mdc}(a_i, a_j) = 1$, para $i, j = 1, 2, \ldots, n$, com $i \neq j$.

$a_1, a_2, \ldots, a_n$ são primos dois a dois
$\Downarrow$
$a_1, a_2, \ldots, a_n$ são primos entre si.

A implicação recíproca é falsa: 2, 3 e 4 são primos entre si mas não são primos dois a dois.

MMC de mais do que dois inteiros

$n \in \mathbb{N}, n \geq 2, a_1, a_2, \ldots, a_n \in \mathbb{Z}$ não todos nulos. O menor múltiplo comum de $a_1, a_2, \ldots, a_n$ é o menor dos múltiplos comuns positivos de $a_1, a_2, \ldots, a_n$. Representa-se por $\mathrm{mmc}(a_1, a_2, \ldots, a_n)$.

Propriedades:

1. $\mathrm{mmc}(a_1, a_2, \ldots, a_n)$ é o único múltiplo comum, positivo, de $a_1, a_2, \ldots, a_n$ que divide qualquer múltiplo comum de $a_1, a_2, \ldots, a_n$;
2. $\mathrm{mmc}(a_1, a_2, \ldots, a_n) = \mathrm{mmc}(\mathrm{mmc}(a_1, a_2, \ldots, a_{n-1}), a_n)$.

Para $n \geq 3$, em geral,
$\mathrm{mdc}(a_1, a_2, \ldots, a_n) \times \mathrm{mmc}(a_1, a_2, \ldots, a_n) \neq |a_1, a_2, \ldots, a_n|$.

Definição & Teorema Fundamental da Aritmética

Um inteiro $p > 1$ diz-se um número primo se os únicos divisores positivos de p são 1 e p.

Um inteiro $a > 1$ diz-se um número composto se não é primo.

Teorema
p número primo; $a_1, a_2, \ldots, a_n \in \mathbb{Z}$.

$$p|a_1a_2\cdots a_n \Rightarrow p|a_1 \vee p|a_2 \vee \cdots \vee p|a_n.$$

Teorema (Teorema Fundamental da Aritmética)

Todo o inteiro maior que 1 pode ser escrito, de modo único (a menos da ordem dos factores), como produto de números primos.

Factorização em Primos

Se $a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ e $b = p_1^{\beta_1} p_2^{\beta_2} \dots p_k^{\beta_k}$,
onde $p_1, \dots, p_k$ são números primos distintos dois a dois e $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in \mathbb{N}_0$, então

$$a|b \Leftrightarrow (\alpha_i \leq \beta_i, i = 1, \dots, k)$$

$$\mathrm{mdc}(a,b) = p_1^{\min\{\alpha_1,\beta_1\}} p_2^{\min\{\alpha_2,\beta_2\}} \dots p_k^{\min\{\alpha_k,\beta_k\}}$$

e

$$\mathrm{mmc}(a,b) = p_1^{\max\{\alpha_1,\beta_1\}} p_2^{\max\{\alpha_2,\beta_2\}} \dots p_k^{\max\{\alpha_k,\beta_k\}}$$

A Função de Euler

A função de Euler é a função $\phi : \mathbb{N} \to \mathbb{N}$ definida por:

$$\phi(n) = |\{a \in \mathbb{N} : a \leq n \text{ e } \mathrm{mdc}(a,n) = 1\}|, \quad n \in \mathbb{N}$$

.

Teorema

A função ϕ é multiplicativa, isto é, se $m, n \in \mathbb{N}$ são tais que $\mathrm{mdc}(m,n) = 1$, então

$$\phi(mn) = \phi(m)\phi(n)$$

.

Teorema

Sejam, $p_1, \dots, p_k$, números primos distintos dois a dois e $\alpha_1, \dots, \alpha_k \in \mathbb{N}$. Então:

$$\phi(p_1^{\alpha_1} \cdots p_k^{\alpha_k}) = \left(p_1^{\alpha_1} - p_1^{\alpha_1 - 1}\right) \cdots \left(p_k^{\alpha_k} - p_k^{\alpha_k - 1}\right).$$

Observação: $n \in \mathbb{N}$ é primo se e só se $\phi(n) = n - 1$.

Pequeno Teorema de Fermat

Teorema (Pequeno Teorema de Fermat)

Se n é um número primo, então $a^{n-1} \equiv 1 (\mathrm{mod}\ n)$, para todo o $a \in \mathbb{Z}$ tal que $\mathrm{mdc}an = 1$

Teorema Chinês dos Restos

Teorema (Teorema Chinês dos Restos)

Sejam $m_1, m_2, \dots, m_k \in \mathbb{N}$ primos dois a dois e $a_1, a_2, \dots, a_k \in \mathbb{Z}$.
O sistema

$$\begin{cases} x \equiv a_1 (\mathrm{mod}\ m_1) \\ \quad \vdots \qquad\qquad (1) \\ x \equiv a_k (\mathrm{mod}\ m_k) \end{cases}$$

tem solução.
Seja $m = m_1 m_2 \dots m_k$. Para $i = 1, 2, \dots, k$ seja $b_i \in \mathbb{Z}$ tal que $\frac{m}{m_i} b_i \equiv 1 (\mathrm{mod}\ m_i)$ e considere-se

$$x_0 = \sum_{i=1}^{k} \frac{m}{m_i} a_i b_i.$$

O conjunto das soluções do sistema é $[x_0]_m$.

RSA: Bibliografia

- R. Rivest, A. Shamir, e L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems*. Communications of the ACM, 21(2):120–126, 1978.
- D. Atkins, M. Graff, A. Lenstra, and P. Leyland. *The magic words are squeamish ossifrage*. In ASIACRYPT 1994, pages 263–277, 1994.
- Pedro Quaresma e Elsa Lopes, «Criptografia», Gazeta de Matemática, 154, 7–11, Março de 2008, SPM, Lisboa.

RSA

Com $n = pq$, p e q primos.

Chave Pública, $\mathcal{C}_p = (e, n)$
$1 < e < \phi(n)$ e $\text{mdc}(e, \phi(n)) = \text{mdc}(e, (p-1)(q-1)) = 1$.

Chave Privada, $\mathcal{C}_s = (d, n)$
d é o inverso multiplicativo de e, módulo $\phi(n)$.

O *algoritmo de encriptação*, $\mathcal{A}_{C_p} : M \to \mathcal{A}_{C_p}(M) = C$, é:

$$C = M^e (\text{mod } n)$$

O *algoritmo de desencriptação*, $\mathcal{A}_{C_s} : C \to \mathcal{A}_{C_p}(C) = M$, é:

$$M = C^d (\text{mod } n)$$

RSA — Geração de Chaves

1. Geração de um par de números Primos
   - →
   - ← um par de primos de **grande dimensão**
     - ≥ 704bits, 212 dígitos. ▸ Desafio RSA
2. Geração das chaves
   - → p, q (dois números primos);
   - ← (e, n) e (d, n), as chaves: pública e privada.

e Escolhe-se um e tal que $1 < e < \phi(n)$ e primo relativo com $\phi(n)$.

Com $\phi(n)$, a função de Euler, que nos dá o número de primos relativos com n.

$\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$, isto é, dada a escolha feita acima, p e q, $\phi(n)$ é fácil de calcular.

d d é o inverso multiplicativo de e, módulo $\varphi(n)$, isto é $de = 1(\text{mod } \phi(n))$

Dado que $a \equiv b(\text{mod } n)$ então $a = b + kn$ para um dado $k \in \mathbb{Z}$ e que $de = 1(\text{mod } \phi(n))$. Podemos calcular o d achando o k tal que $d = (1 + k\phi(n))/e$, seja uma divisão inteira exacta.

RSA — Geração de Chaves — Algoritmo

```
/*
 * Determinação das Chaves Pública e Privada:
 * ⟶ p,q dois números primos.
 * ⟵ (e,n) e (d,n), as chaves públicas e privadas.
 */
void chaves(long int p, long int q, long int *e, long int *d) {
  long int n,k,fi;
  bool achou;
  double auxd;

  fi = (p-1)*(q-1);  // função de Euler
  // cálculo de 'e'
  *e = 2;              // escolha inicial de 'e'
  do {                 // incrementa o 'e' até ser primo relativo com fi
    (*e)++;
  } while (mdc(fi,*e) != 1); //  mdc - algoritmo de Euclides
  // cálculo de 'd'
  achou = false;
  k = 0;
  while (!achou){ // incrementa o 'k' até
    *d = (1 + (k * fi))/(*e); // divisão inteira
    if ( ((1 + (k * fi)) % (*e)) == 0) { // a divisão é exacta (resto zero)
      achou = true;
    }
    else {
      k = k+1;
    }
  }
}
```

RSA — Encriptação

1. Encriptação
   - → mensagem a cifrar (em binário);
   - preenchimento (Modo 1 ou 2) e divisão em blocos (ECB ou CBC);
   - encriptar bloco a bloco através da função de encriptação;
   - ← texto cifrado.

```
/*
 * Encriptar RSA
 * ---> (e,n): chave pública
 *      m: mensagem a cifrar (vector de inteiros)
 *      comp: comprimento do bloco
 * <--- x, mensagem cifrada (vector de inteiros)
 */
void cifrar(long int e, long int n, long int m[], long int x[], int comp) {
  int i;
  for (i=1;i<comp;i++) {
    x[i] = mod((potencia(m[i]),e), n); // mod e potencia, funções auxiliares
  }
}
```

Para aumentar a eficiência da encriptação é desejável escolher um expoente de encriptação e pequeno tal como $e = 3$.

RSA — Desencriptação

1. Desencriptação
   - → mensagem cifrada;
   - divisão em blocos (ECB ou CBC);
   - desencriptar bloco a bloco através da função de desencriptação;
   - ← texto original.

```
/*
 * Desencriptação RSA
 * ---> (d,n): chave privada
 *      c: mensagem a decifrar (vector de inteiros)
 *      comp: comprimento do bloco
 * <--- m: mensagem decifrada (vector de inteiros)
 */
void decifrar(long int d, long int n, long int c[], long int m[], int comp){
  int i,j;

  for (i=1;i<comp;i++){
    m[i] = 1;
    j=1;
    while (j <= d) {
      m[i] = mod((c[i])*m[i], n);
      j++;
    }
  }
}
```


Teorema — Cifra RSA

Para que o algoritmo RSA possa ser considerado uma cifra o procedimento tem de ser invertível.

$$A_{C_s}(A_{C_p}(M)) = A_{C_p}(A_{C_s}(M)) = M^{ed} (\mathrm{mod}\ n) = M$$

Para provar este resultado são necessários dois resultados auxiliares da teoria dos números: a definição de Congruência módulo n e a consequência que daí se tira que se $a \equiv b(\mod n)$ então $a = b + kn$, para um dado $k \in \mathbb{Z}$.

Para o desenvolvimento da demonstração são necessários alguns resultados auxiliares: a definição de *Congruência módulo* n, o *Pequeno Teorema de Fermat* e o *Teorema Chinês dos Restos*.

Teorema (Sistema de Criptografia RSA)

Sendo (e, n) *e* (d, n) *as chaves pública e privada respectivamente do Sistema de Criptografia RSA verifica-se então que:*

$$(m^e)^d (\mathrm{mod}\ n) = m$$

para qualquer inteiro m*, com* $0 \leq m < n$.

Demonstração — Cifra RSA

Da definição de e e d tira-se que $ed \equiv 1(\mod \phi(n))$ existe então um $k \in \mathbb{Z}$ tal que $ed = 1 + k\phi(n)$, ou seja:

$$ed = 1 + k(p-1)(q-1),\ k \in \mathbb{Z}$$

donde

$$(m^e)^d = m^{ed} = m^{1+k(p-1)(q-1)} = m(m^{(p-1)(q-1)})^k$$

segue-se que

$$(m^e)^d \equiv m(m^{(p-1)})^{(q-1)k} \equiv m(\mathrm{mod}\ p)$$

Se p não é um divisor de m esta congruência é uma consequência do *Pequeno Teorema de Fermat*. Caso contrário a asserção é trivial dado que ambos os membros da equação são congruentes com $0 \mod p$.
De forma análoga ter-se-ia que:

$$(m^e)^d \equiv m(\mathrm{mod}\ q)$$

Dado que p e q são números primos distintos pode-se aplicar o *Teorema Chinês dos Restos* e dado que se assume que $0 \leq m < n$, obtém-se

$$(m^e)^d \equiv m(\mathrm{mod}\ pq) \equiv (\mathrm{mod}\ n) = m$$

□

Expoente de encriptação e pequeno

De forma a melhorar a eficiência da encriptação é desejável seleccionar um expoente de encriptação e pequeno.

$$c = m^e \pmod{n}$$

No entanto um expoente de encriptação pequeno, tal como $e = 3$ não deve ser usada se se vai enviar a mesma mensagem para vários destinos, ou a mesma mensagem com pequenas variantes.
Ao enviar a mesma mensagem m para três entidades cujos módulos públicos são n_1, n_2, e n_3 e cujo expoente de encriptação é, $e = 3$, vai se enviar $c_i = m^3 \bmod n_i$ para $1 \leq i \leq 3$.
Dado que é provável que os módulos sejam primos relativos dois a dois, pode-se usar os textos encriptados para achar a solução x, $0 \leq x < n_1 n_2 n_3$, das três congruências

$$\begin{cases} x \equiv c_1 \pmod{n_1} \\ x \equiv c_2 \pmod{n_2} \\ x \equiv c_3 \pmod{n_3} \end{cases}$$

Dado que $m^3 < n_1 n_2 n_3$, pelo Teorema Chinês dos Restos, tem-se que $x = m^3$. Consequentemente, calculando a raiz cúbica inteira de x, o adversário pode recuperar o texto claro m.

Expoente de encriptação e pequeno

Para prevenir o tipo de ataque descrito atrás, deve-se «**salgar**» a mensagem:

Definição (Salgar a mensagem)

Designa-se por salgar uma mensagem o acto de apensar à mensagem uma sequência de bits pseudo-aleatória de um comprimento apropriado antes da encriptação.
A sequência pseudo-aleatória deve ser gerada de forma independente para cada encriptação.

Expoentes de encriptação pequenos são também um problema quando conjugados com mensagens m pequenas, isto dado que se $m < \sqrt[e]{n}$, então m pode ser recuperado do texto cifrado $c = m^e \bmod n$ simplesmente calculando a raiz inteira de ordem e de c.

O salgar da mensagem de texto claro também resolve este problema.

Ataque por Procura Exaustiva

Se o espaço das mensagens é pequeno ou previsível, um adversário pode decifrar o texto cifrado c simplesmente encriptando todas as mensagens em texto claro possíveis até que se obtenha c.

O salgar das mensagens é uma forma simples de prevenir um tal ataque.

Expoente de desencriptação d pequeno

Assim como no caso do expoente de encriptação e, pode ser desejável seleccionar um expoente de desencriptação pequeno d de forma a melhorar a eficiência da desencriptação.

$$m = c^d \pmod{n}$$

No entanto se $\mathrm{mdc}(p-1, q-1)$ é pequeno, como é usual, e se d tem, quanto muito um quarto dos bits de n, então existe um algoritmo eficiente para calcular d a partir da chave pública.

Este algoritmo não é extensível para o caso em que d tem aproximadamente o mesmo tamanho que n.

Consequentemente, para evitar este tipo de ataque, o expoente de desencriptação d deve ter, aproximadamente, o mesmo comprimento que n.