CiberSegurança
Módulo 1 - 04 CIFRAS ASSIMÉTRICAS

MEET, MEIC, MEIM

2021–2022

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Em 1975, Whitfield Diffie, Martin Hellmann e Merkle idealizaram um novo sistema de encriptação chamado de **chave assimétrica** ou **chave pública**.

A ideia consistia num sistema de cifrado baseado em funções matemáticas que eles denominaram de *one way functions*, isto é, funções f tais que

- f é computável em tempo polinomial;
- f^{-1} não é computável em tempo polinomial (a menos que se tenha informação extra, uma *trap-door*).

Em 1977, Ron Rivest, Leonard Adleman e Adir Shamir publicaram pela primeira vez um candidato a *one-way funcion*, com *trap-door*: a cifra RSA.

Existem *one-way functions*?

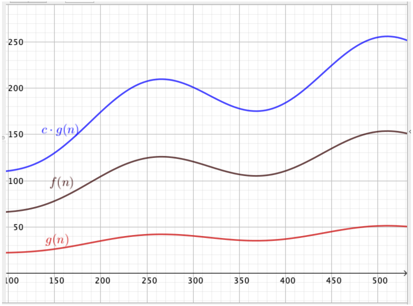
Todas as *one-way function* usadas em criptografia são, até a data, unicamente **candidatos** a *one-way function*, não foi provada formalmente a existência de tais funções.

$\rightsquigarrow$ Por exemplo a segurança da cifra RSA está baseada, dados p e q, números primos grandes e $n = p \cdot q$, na dificuldade computacional (**à data de hoje, com recursos de computação clássica**) em calcular p e q a partir de n.

- A teoria da complexidade tem como objetivo classificar algoritmos de acordo aos recursos necessários para os resolver;
- Frequentemente, a complexidade de um algoritmo é medida pelo tempo ou quantidade de espaço necessária para realizar a tarefa, que dependem, em geral, do tamanho do *input* do algoritmo;
- O objetivo principal das medidas de complexidade não é determinar exatamente o tempo ou memória necessários, mas sim conseguir comparar procedimentos.

Notação usada frequentemente para comparar o tempo de execução máxima de um algoritmo em função do tamanho do *input*.

Dadas duas funções $f, g : \mathbf{N} \to [0, +\infty[$, diz-se que $f \in O(g)$ si existe uma constante $c > 0$ tal que $f(n) \leq cg(n)$ para quase todo o $n \in \mathbf{N}$.



Exemplos: notação Big-O

1. $O(1)$ - constante: idependentemente do tamanho do *input*, o tempo de execução permanece constante;
2. $O(\log n)$ - logarítmico: o tempo de execução é proporcional ao logaritmo do tamanho do *input*
3. $O(n)$ - linear: o tempo de execução é proporcional ao tamanho do *input*;
4. $O(n^2)$ - quadrático: tempo de execução proporcional ao quadrado do tamanho do *input*;
5. $O(2^n)$ - exponencial: tempo de execução proporcional à exponencial do tamanho do *input*.

   $\rightsquigarrow$ **Se o tamanho do input passa de n a $n+1$, o tempo de execução duplica!**

Comportamento assintótico das operações aritméticas simples, assumindo que n é maior que o tamanho do input x, y:

$x + y$	$O(n)$
$x \cdot y$	$O(n^2)$
$x \operatorname{div} y$	$O(n^2)$
$x \pmod y$	$O(n^2)$

(Dados x, y inteiros positivos, $x \operatorname{div} y$ e $x \pmod y$ denotam, respetivamente, o quociente e o resto da divisão inteira de x por y.)

$\rightsquigarrow$ Os algoritmos de factorização conhecidos em computação clássica são, no *worst case scenario*, **quase-exponenciais**.

Parâmetros gerais

1. São escolhidos primos p e q que satisfaçam as condições de segurança e é determinado o módulo $n = p \times q$
2. São determinados ítems de informação, k e K, designados por **chave pública** e **chave privada**, de tal modo que:
   1. o conhecimento do par (p, q) é equivalente ao conhecimento da chave privada k;
   2. o conhecimento da chave pública K implica o conhecimento do módulo n.

Exemplos: cifras RSA e Rabin

Seja n um inteiro positivo. A função de Euler-phi (ou Euler-totient), denotada por ϕ, associa a cada n o número de inteiros menores que n e primos com n, ou seja, o número de inteiros $1 \leq a < n$ tais que $\gcd(a, n) = 1$.

Exemplos:

1. $\phi(2) = 1$
2. $\phi(3) = 2$
3. $\phi(5) = 4$
4. $\phi(6) = 2$
5. $\phi(8) = 4$
6. $\phi(17) = 16.$

A função ϕ de Euler verifica:

1. Se p é um número primo, então $\phi(p) = p - 1$;
2. Se p é um número primo, então $\phi(p^k) = p^k - p^{k-1} = p^{k-1}(p-1)$;
3. Se n e m são dois inteiros positivos primos entre sim, então

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

Exemplos:

1. $\phi(6) = \phi(2)\phi(3) = 1 \cdot 2 = 2$;
2. $\phi(8) = \phi(2^3) = 2^3 - 2^2 = 8 - 4 = 4$;
3. $\phi(25) = 5^2 - 5 = 20$;
4. $\phi(12) = \phi(2^2)\phi(3) = 4$

Se n um número inteiro e p_1, p_2, ..., p_k os seus factores primos, então

$$\phi(n) = n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_k}\right)$$

Efectivamente, dada a factorização em primos do número inteiro n, se $n = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, verifica-se que

$$\begin{aligned}\phi(p) &= p_1^{k_1-1}(p_1-1)p_2^{k_2-1}(p_2-1)\cdots p_r^{k_r-1}(p_r-1)\\ &= p_1^{k_1-1}p_1\left(1-\tfrac{1}{p_1}\right)p_2^{k_2-1}p_2\left(1-\tfrac{1}{p_2}\right)\cdots p_r^{k_r-1}p_r\left(1-\tfrac{1}{p_r}\right)\end{aligned}$$

Por outras palavras,

$\leadsto$ **$\phi(n)$ é fácil de calcular dada a fatorização em primos de n, mas intratável caso contrário ...**

Por exemplo $\phi(45003677) = \phi(5683)\cdot\phi(7919) = 5682\cdot 7918 = 44990076$ porque $45003677 = 7919\cdot 5683$, é um produto de dois números primos.

O valor de $\phi(n)$ é precisamente o número de elementos invertíveis em $\mathbf{Z}_n$, ou seja, o número de elementos de $\mathbf{Z}_n^*$:

$$\phi(n) = |\mathbf{Z}_n^*|$$

Exemplos:

1. O número de elementos invertíveis em $\mathbf{Z}_{13}$ é $\phi(13) = 12$;
2. O número de elementos invertíveis em $\mathbf{Z}_{25}$ é $\phi(5^2) = 20$;
3. O número de elementos invertíveis em $\mathbf{Z}_{26}$ é $\phi(13 \cdot 2) = \phi(13)\phi(2) = 12$;
4. O número de elementos invertíveis em $\mathbf{Z}_{45003677}$ é 4990076

$$\phi(45003677) = \phi(5683) \cdot \phi(7919) = 5682 \cdot 7918 = 44990076$$

porque $45003677 = 7919 \cdot 5683$, é um produto de dois números primos.

Seja $n \geq 2$ um inteiro. Se $a \in \mathbf{Z}_n$ é invertível, então

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Em particular $a^{-1} \equiv a^{\phi(n)-1} \pmod{n}$

Exemplos

1. se $n = 3$, tem-se que $a^2 \equiv 1 \pmod{3}$ e que $a^{-1} = a \pmod{3}$;
2. se $n = 7$, tem-se que $a^6 \equiv 1 \pmod{7}$ e que $a^{-1} = a^5 \pmod{7}$;
3. se $n = 8$, tem-se que $a^4 \equiv 1 \pmod{8}$ e que $a^{-1} = a^3 \pmod{8}$

$\rightsquigarrow$ Este resultado permite calcular inversos módulo n, **desde que seja conhecida** $\phi(n)$.

Se p, q são co-primos, $\phi(p \cdot q) = (p-1)(q-1)$

Seja $n = 26 = 2 \times 13$, tem-se que $\phi(n) = 12$. Se a é invertível $\bmod 26$:

$$a^{12} \equiv 1 \pmod{26} \quad \text{e} \quad a^{-1} \equiv a^{11} \pmod{26}$$

Por exemplo, considerando $a = 5$, que é invertível porque 5 e 26 são primos entre si, tem -se que

$$5^{-1} \equiv 5^{11} \pmod{26}.$$

As potências modulares são mais rápidas de calcular do que parece, aplicando o chamado **repeated squaring method**:

$$\begin{array}{lcl} 5^2 & = & 25 \pmod{26} = (-1) \pmod{26} \\ 5^4 & = & (5^2)^2 = (-1)^2 \pmod{26} = 1 \pmod{26} \\ 5^8 & = & (5^4)^2 = 1 \pmod{26} \end{array}$$

por tanto

$$5^{11} = (5^8)(5^2)5 = 1(-1)5 \pmod{26} = (-5) \pmod{26} = 21 \pmod{26}$$

Assim, o inverso módulo 26 de 5 é 21.

Redução de expoentes $\bmod \phi(n)$

Seja $n \geq 2$ produto de primos distintos, e r e s inteiros positivos.

$$\text{Se } r \equiv s (\bmod \phi(n)) \quad \text{então} \quad a^r \equiv a^s (\bmod n)$$

Por outras palavras, se n é produto de primos distintos, ao trabalhar módulo n os expoentes podem reduzir-se módulo $\phi(n)$.

Atenção ...

- Nos expoentes das potências **não** se trabalha $\bmod n$;

  Por exemplo, $3^2 \equiv 4 \bmod 5$ mas $3^{2+5} = 3^7 \equiv 2 \bmod 5$

- Se $n = p_1 \cdot p_2 \cdots p_k$. com p_i primos distintos, podemos trabalhar nas potências módulo $\phi(n)$;

  Por exemplo, $2^2 \equiv 4 \bmod 15$ e $2^{2+8} = 2^{10} \equiv 4 \bmod 15$, $\quad (\phi(15) = 8)$

- Se n contém factores primos repetidos, essa redução não funciona em geral.

  Por exemplo, $2^2 \equiv 4 \bmod 8$ mas $2^{2+4} = 2^6 \equiv 0 \bmod 8$, $\quad (\phi(8) = 4)$

1. Considere-se $n = 6$, produto dos primos $p = 3$ e $q = 2$.

   Tem-se que $\phi(6) = 2$, pelo que, aplicando a proposição anterior, se $r \equiv s \pmod 2$, com r e s positivos, então $x^r \equiv x^s \pmod 6$ e então:

   $$x^2 \equiv x^4 \pmod 6, \ \forall x \in \mathbf{Z}_6$$

   $$x \equiv x^3 \pmod 6, \ \forall x \in \mathbf{Z}_6$$

2. Considere-se $n = 26$, producto dos primos $p = 13$ e $q = 2$.

   Tem-se que $\phi(26) = 12$, pelo que, aplicando a proposição anterior, se $r \equiv s \pmod{12}$, com r e s positivos, então $x^r \equiv x^s \pmod{26}$ e então, por exemplo

   $$x^2 \equiv x^{14} \pmod{26}, \ \forall x \in \mathbf{Z}_{26}$$

   $$x^{10} \equiv x^{22} \pmod{26}, \ \forall x \in \mathbf{Z}_{26}$$

O caso $n = p$

No caso particular em que $n = p$ é um número primo, obtemos uma versão simplificada dos resultados anteriores.

Seja p um número primo,

1. **(Teorema de Fermat)** Se $\gcd(a, p) = 1$, então $a^{p-1} \equiv 1 (\bmod p)$ e $a^p \equiv a (\bmod p)$.
2. Se $r \equiv s (\bmod p - 1)$ então

$$a^r \equiv a^s (\bmod p)$$

Atenção que a^0 ou a^{-1} 'so estão definidos se a é invertível!!!!

Se $n \geq 2$ é produto de primos distintos e $t \equiv r^{-1} \bmod \phi(n)$, ou seja, se

$$r \cdot t \equiv 1 \bmod \phi(n)$$

então

$$(x^r)^t \equiv x \bmod n$$

$\rightsquigarrow$ Dado um expoente r para encontrar o expoente *inverso* t é preciso calcular o seu inverso $\bmod \phi(n)$.

$\rightsquigarrow$ Se é conhecida a factorização $n = p \cdot q$, então é computacionalmente tratável calcular $r^{-1} \bmod \phi(n)$, porque

$$\phi(n) = (p-1)(q-1).$$

Exemplo: $\bmod 299$

Considere-se $n = 299$, produto dos primos $p = 23$ e $q = 13$. Tem-se que

$$\phi(299) = 22 \cdot 12 = 264.$$

Considerando, por exemplo, $r = 115$ e $s = 163$, que são inversos módulo 264,

$$115 \cdot 163 \equiv 1 \pmod{264}$$

tem-se que:

$$\left(x^{115}\right)^{163} \equiv x \pmod{299}$$

Parâmetros específicos RSA

Denotamos por p,q os primos escolhidos, por $n = p \cdot q$ o módulo e por ϕ a função de Euler do módulo.

1. É calculado $\phi = \phi(n) = (p-1)(q-1)$ e escolhido $e \in \mathbf{Z}_{\phi}^{*}$;
2. É calculado $d = e^{-1} (\bmod \phi)$ e são destruídos os valores p, q, ϕ;
3. A chave pública é $k =< n, e >$, a chave privada é $K =< n, d >$

Os inteiros e e d são chamados, respetivamente, **expoente de encriptação** (*encryption exponent*) e **expoente de decifrado** (*decryption exponent*).

Consideramos os primos $p = 23$ e $q = 13$, então

$$n = p \cdot q = 299 \quad \text{e} \quad \phi(299) = 22 \cdot 12 = 264.$$

Precisamos de escolher $e \in \mathbf{Z}_{264}$, invertível, por exemplo, $e = 115$ e calcular o seu inverso em $\mathbf{Z}_{264}$, que é $d = 163$.

Chave pública $k = (299, 115)$, chave privada $K = (299, 163)$.

Observe-se que, se $k = (n, e)$ e $K = (n, d)$ são as chaves pública e privada obtidas do modo anterior, então, para todo o $x \in \mathbf{Z}_n$, tem-se que

$$(x^e)^d \equiv x \pmod n.$$

Métodos de cifragem e decifragem - RSA

Sejam $k = (n, e)$ e $K = (n, d)$ as chave pública e privada, respetivamente.

- Cifragem (usa a chave pública): para cifrar $x \in \mathbf{Z}_n$,

$$\mathrm{rsa}(x) = x^e \bmod (n)$$

- Descifragem (usa a chave privada): para decifrar $y \in \mathbf{Z}_n$,

$$\mathrm{rsa}^{-1}(y) = y^d \bmod (n)$$

$\rightsquigarrow$ A cifra RSA encripta um elemento x em $\mathbf{Z}_n$ em outro elemento y de $\mathbf{Z}_n$. Assim, para cifrar um texto claro usando RSA é preciso representar de algum modo o texto claro (ou blocos do texto claro) como um inteiro em $[0, n-1]$.

Exemplo 1: cifra RSA - cifragem e decifragem

Os parâmetros deste exemplo não podem ser usados na prática.

Considerando a chave pública $k = (299, 115)$ e chave privada $K = (299, 163)$.

$$\mathrm{rsa}(x) = x^{115} \bmod (299) \qquad \mathrm{rsa}^{-1}(y) = y^{163} \bmod (299)$$

Por exemplo,

$$x = 100,\ \ \mathrm{rsa}(100) = 100^{115} \bmod (299) = 269$$

e efectivamente

$$\mathrm{rsa}^{-1}(269) = 269^{163} \bmod (299) = 100$$

Os parâmetros deste exemplo não podem ser usados na prática.

Considerando a chave pública $k = (988027, 490063)$ e a chave privada $K = (988027, 282607)$.

$$\mathrm{rsa}(x) = x^{490063} \bmod (988027) \qquad \mathrm{rsa}^{-1}(y) = y^{282607} \bmod (988027)$$

Por exemplo,

$$x = 100,\ \mathrm{rsa}(100) = 100^{490063} \bmod (988027) = 827479$$

e efectivamente

$$\mathrm{rsa}^{-1}(827479) = 827479^{282607} \bmod (988027) = 100$$

Sobre a implementação da RSA

Na definição dos parâmetros da cifra RSA, é necessário escolher *aleatoriamente*, dois primos distintos, p e q para definir $n = p \cdot q$ e um inteiro e invertível em $\mathbf{Z}_{\phi(n)}$. Depois, é necessário ainda inverter e em $\mathbf{Z}_{\phi(n)}$, para conseguir a chave privada d. Note-se que:

- Uma escolha *aleatória* significa, na prática, a escolha de um número com a ajuda de um gerador pseudo-aleatório de números. Uma escolha inadequada do gerador compromete a cifra.
- A escolha de um número primo passa por escolher, primeiro, aleatoriamente um inteiro ímpar qualquer m (se for par, alterar para $m + 1$) e aplicar um *teste de primalidade* para verificar se m efetivamente é ou não primo. Se não for, tentar $m + 2$, $m + 4$ ... até conseguir um primo. O famoso **Teorema do Número Primo** permite esperar que se encontre o primeiro primo $p \geq m$, após testar $O(\log m)$ números.
- Para conseguir um RSA resistente aos ataques mais frequentes é necessário (mas **não suficiente** ;)) que os primos p e q sejam grandes, aproximadamente do mesmo tamanho e tais que $p - q$ não é demasiado pequeno.

No caso dos números reais, usando ferramentas computacionais, não há diferença significativa entre o cálculo de

$$y = \alpha^x$$

ou da sua função inversa,

$$x = \log_\alpha(y)$$

com a mesma precisão.

No caso dos conjuntos $\mathbf{Z}_n$, a potencia α^x pode ser calculada muito rapidamente usando o *repeated-squaring method*.

No entanto, para o processo inverso, ou seja, dado y determinar um x tal que $\alpha^x = y$ em $\mathbf{Z}_n$, não se conhecem em computação clássica algoritmos eficientes.

Calculam-se as potências sucessivas α^2, α^4, α^8, α^{16} ... módulo n.

O cálculo de α^x realiza-se considerando a descrição de x em base 2.

Exemplo:

Considerem-se $p = 23$ e $\alpha = 7$. Para calcular, por exemplo, $7^{21} \bmod 23$, calculamos

$$\begin{array}{rcccl} 7^2 & = & 49 & \equiv & 3 \pmod{23} \\ 7^4 & = & 7^2 \cdot 7^2 & \equiv & 3 \cdot 3 \pmod{23} \equiv 9 \pmod{23} \\ 7^8 & = & 7^4 \cdot 7^4 & \equiv & 9 \cdot 9 \pmod{23} \equiv 12 \pmod{23} \\ 7^{16} & = & 7^8 \cdot 7^8 & \equiv & 12 \cdot 12 \pmod{23} \equiv 6 \pmod{23} \end{array}$$

e então

$$7^{21} = 7^{16} \cdot 7^4 \cdot 7 \equiv 6 \cdot 9 \cdot 7 \pmod{23} = 10 \pmod{23}$$

Observe-se que, dado $y = 10$, para tentar calcular x tal que $\alpha^x = 10$, com $\alpha = 7$ seria preciso testar todas as potências até $x = 21$:

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$7^x \bmod 23$	7	3	21	9	17	4	5	12	15	13	22	16	20	2	14	6	19	18	11	8	10	1

O problema do logaritmo discreto com base α e módulo n, consiste em, dados $\alpha, y \in \mathbf{Z}_n^*$, determinar $k \in \mathbf{Z}$ tal que

$$\alpha^k = y \bmod n$$

Se o problema tiver solução para todo o y, α diz-se um **gerador multiplicativo** de $\mathbf{Z}_n^*$, ou **gerador multiplicativo módulo** n.

Teorema (Gauss): Existem geradores multiplicativos módulo n se e só se $n = 2, 4, p^k$ ou $2p^k$, com p um primo ímpar e $k \in \mathbf{N}$.

Por exemplo:

- Existem geradores multiplicativos módulo 5, módulo 23 ...
- Não existem geradores multiplicativos módulo 8.
  (tem-se que $Z_8^* = \{1, 3, 5, 7\}$, com $3^2 = 5^2 = 7^2 = 1 \bmod 8$)

Exemplo: geradores multiplicativos $\bmod 5$

Se $p = 5$, tem-se que $\mathbf{Z}_5^* = \{1, 2, 3, 4\}$ e verifica-se que

$$\begin{array}{lll} 2^2 \equiv 4 \pmod 5 & 2^3 \equiv 3 \pmod 5 & 2^4 \equiv 1 \pmod 5 \\ 3^2 \equiv 4 \pmod 5 & 3^3 \equiv 2 \pmod 5 & 3^4 \equiv 1 \pmod 5 \\ 4^2 \equiv 1 \pmod 5 & 4^3 \equiv 4 \pmod 5 & 4^4 \equiv 1 \pmod 5 \end{array}$$

Então, 2 e 3 são geradores de $\mathbf{Z}_5^*$, enquanto que 4 não é gerador de $\mathbf{Z}_5^*$

$\rightsquigarrow$ Como $\mathbf{Z}_p^*$ possui exatamente $\phi(p) = p - 1$ elementos, α será um gerador multiplicativo módulo p se e só se o menor k tal que $\alpha^k = 1 \bmod p$ é precisamente $k = p - 1$.

A lista completa de geradores multiplicativos módulo 23 é

$$(5, 7, 10, 11, 14, 15, 17, 19, 20, 21)$$

Podemos verificar, por exemplo, que 2 não é um gerador multiplicativo módulo 23:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$2^k \bmod 23$	2	4	8	16	9	18	13	3	6	12	1	2	4	8	16	9	18	13	3	6	12	1

Outro elemento que não é gerador de $\mathbf{Z}_{23}^*$ é 22, visto que

$$22^2 = 1 (\bmod 23) \quad 22^3 \equiv 22 (\bmod 23) \quad 22^4 = 1 (\bmod 23) \cdot$$

- Seja $p = 991$. O grupo multiplicativo $\mathbf{Z}^*_{991}$ possui 240 geradores, entre os quais 6, 7, 11 e 989. Observe-se que 990 não pode ser gerador visto que
$$990 \equiv (-1) (\bmod 991)$$
pelo que as potências pares de 990 são iguais a 1 e as potências impares são iguais a -1 (ou seja, $990 (\bmod 991)$)

- Se p é um primo, 1 e $p - 1$ nunca são geradores de $\mathbf{Z}^*_p$. Em geral, $n - 1$ nunca é gerador do grupo multiplicativo $\mathbf{Z}^*_n$

O Protocolo Diffie-Hellman de intercâmbio de chaves

- Primeira solução prática encontrada ao problema da distribuição das chaves (1976).
- Este protocolo permite a duas entidades, sem partilha de informação anterior ou canal seguro de comunicação, estabelecer uma chave secreta através de um canal aberto.
- O protocolo está baseado no problema do logaritmo discreto, mais precisamente, no seguinte pressuposto (problema de Diffie-Hellman, DHP):

Se p é um número primo suficientemente grande, o cálculo de α^{xy} a partir de α, α^x e α^y em $\mathbf{Z}_p$ é computacionalmente intratável com os recursos atuais de computação clássica.

O protocolo de Diffie-Hellman é também chamado de **intercâmbio exponencial da chave** (*exponential key exchange*)

Protocolo básico de Diffie-Hellman de intercâmbio de chaves

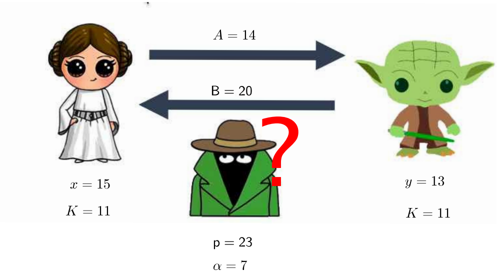
Dados públicos: p primo e um inteiro α, tal que $2 \leq \alpha \leq p-2$ seja um **gerador multiplicativo** de $\mathbf{Z}_p^*$.

- Alice escolhe aleatoriamente x, $1 \leq x \leq p-2$, calcula $A = \alpha^x \pmod p$ e envia a Bob o valor A obtido. Bob escolhe aleatoriamente y, com $1 \leq y \leq p-2$, calcula $B = \alpha^y \pmod p$ e envia a Alice o valor B obtido.
- Alice calcula $K = B^x = (\alpha^y)^x$, Bob calcula $K = A^y = (\alpha^x)^y$.

A **chave partilhada** é K

$\rightsquigarrow$ A escolha aleatória de x, y precisa de um **adequado** gerador pseudo-aleatório de números...

Considerando $p = 23$ e $\alpha = 7$ (dados públicos)



- Alice escolhe aleatoriamente $x = 15$ e calcula $7^{15} \equiv 14 \pmod{23}$, donde $A = 14$;
- Bob escolhe aleatoriamente $y = 13$ e calcula $7^{13} \equiv 20 \pmod{23}$, donde $B = 20$.
- Alice calcula a chave secreta partilhada como $B^x \pmod{23} = 20^{15} \pmod{23} = 11 \pmod{23}$, Bob calcula a mesma chave secreta usando $A^y \pmod{23} = 14^{13} \pmod{23}$, obtendo os dois o valor $K = 11$.

A cifra ElGama está baseada também no problema do logaritmo discreto em $\mathbf{Z}_p$, mais precisamente, em que

Se p é um número primo suficientemente grande, e α um gerador do grupo multiplicativo $\mathbf{Z}_p^*$, o cálculo de a a partir de p, α, α^a em $\mathbf{Z}_p$ é computacionalmente intratável com os recursos atuais de computação clássica.

Parâmetros específicos ElGamal

Seja p um primo.

1. É calculado α um gerador do grupo multiplicativo $\mathbf{Z}_p^*$
2. É escolhido aleatoriamente um inteiro a, $2 \leq a \leq p - 2$, e calculado $\alpha^a \bmod p$.
3. A chave pública é $k =< p, \alpha, \alpha^a >$, a chave privada é $K =< p, a >$

Seja $p = 13$ e consideramos $\alpha = 2$. Observe-se que 2 é efetivamente um gerador de $\mathbf{Z}_{13}^*$:

a	1	2	3	4	5	6	7	8	9	10	11	12
$2^a \bmod 13$	2	4	8	3	6	12	11	9	5	10	7	1

Escolhemos aleatoriamente $a = 6$ e então $2^6 = 12$. A chave pública k e a chave privada K de ElGamal são, respetivamente,

$$k =< p, \alpha, \alpha^a >=< 13, 2, 12 >$$

e

$$K =< p, a >=< 13, 6 >$$

Exemplo 2: Parâmetros específicos da cifra ElGamal

Seja $p = 23$ e consideramos $\alpha = 5$. Observe-se que 5 é efetivamente um gerador de $\mathbf{Z}_{23}^*$:

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$5^a \bmod 23$	5	2	10	4	20	8	17	16	11	9	22	18	21	13	19	3	15	6	7	12	14	1

Escolhemos aleatoriamente $a = 19$ e então $5^{19} = 7$. A chave pública k e a chave privada K de ElGamal são, respetivamente,

$$k =< p, \alpha, \alpha^a >=< 23, 5, 7 >$$

e

$$K =< p, a >=< 23, 19 >$$

Sejam $k =< p, \alpha, \alpha^a >$ e $K =< p, a >$ a chave pública e a chave privada, respetivamente.

- Cifragem (com a chave pública):

  Para cifrar $x \in \mathbf{Z}_p^*$, é escolhido aleatoriamente r (**chave efímera**) tal que $2 \leq r \leq p - 2$, e calculados

  $$y = \alpha^r \pmod p \quad \text{e} \quad z = x \cdot (\alpha^a)^r \pmod p$$

  A encriptação de x é definida então por

  $$\text{elgam}(x) = (y, z)$$

- Descifragem (com a chave privada) de um par (y, z), com $y, z \in \mathbf{Z}_p^*$:

  $$\text{elgam}^{-1}(y, z) = \left(y^{-a}\right) z \pmod p$$

Cifra ElGamal (verificação)

Efectivamente, se $(y, z) = (\alpha^r \pmod p), x \cdot (\alpha^a)^r \pmod p)$ então

$$
\begin{aligned}
\mathrm{elgam}^{-1}(y, z) &= (y^{-a})\, z \pmod p \\
&= (\alpha^r)^{-a} \cdot x \cdot (\alpha^a)^r \pmod p \\
&= x \pmod p
\end{aligned}
$$

$\rightsquigarrow$ Recorde-se que

$$y^{-a} = y^{p-1-a} \bmod p,$$

pelo que não é preciso calcular y^{-1} para realizar a decifragem, basta usar *squaring method*.

Considere-se a cifra de ElGamal tal que a chave pública k e a chave privada K são, respetivamente,

$$k =< 13, 2, 12 > \quad \text{e} \quad K =< 13, 6 >$$

Dado um texto claro $x = 10$, para proceder à cifra realizamos o seguinte processo (com a chave pública):

- Escolhemos aleatoriamente r, tal que $1 \leq r \leq 12$, por exemplo, $r = 5$
- Calculamos $y = \alpha^r = 2^5 \pmod{13} = 6 \pmod{13}$
- Calculamos $z = x \cdot (\alpha^a)^r = 10 \cdot 12^5 \pmod{13} = 3 \pmod{13}$

A cifra ElGamal de $x = 10$ é então $(y, z) = (6, 3)$.

Exemplo: decifragem ElGamal

Considere-se a cifra de ElGamal tal que a chave pública k e a chave privada K são, respetivamente,

$$k =< 13, 2, 12 > \quad \text{e} \quad K =< 13, 6 >$$

Dado texto cifrado $(y, z) = (6, 3)$, para proceder à decifragem usando a chave privada $K =< 13, 6 >$, procedemos do modo seguinte:

$$\begin{aligned} x &= \left(y^{p-1-a}\right) z \pmod p \\ &= 6^{13-1-6} \cdot 3 \pmod{13} \\ &= 6^6 \cdot 3 \pmod{13} \\ &= 10 \pmod{13} \end{aligned}$$

Implementação da cifra de ElGamal

1. Por motivos de segurança, é necessário (mas **não suficiente**), que o primo p seja tal que $p - 1$ tenha um divisor primo q muito grande, ou seja, a escolha do primo base para ElGamal deve ser muito cuidadosa (consultar algoritmo 4.84 de [HAC])
2. A escolha aleatória do expoente a na construção da chave e do r, a chave efímera de cada cifrado, precisa, mais uma vez, de um **adequado** gerador pseudo-aleatório de números;
3. A segurança do cifrado fica comprometida se o mesmo expoente efímero r for usado para cifrar dois mensagens;
4. A cifra de ElGamal implica uma *expansão da mensagem*, visto que o texto cifrado é duas vezes mais longo que o texto claro.