# Criptografia de Chave Pública

MOUTA Leonardo

PFC-F - ITA/ISAE-SUPAERO


May 13, 2022

# Objetivo

**Objetivo: Explicar o funcionamento dos criptossistemas RSA e ElGamal**

# Sumário

# Sumário

# Definição Formal de um Criptossistema

Um criptossistema consiste em uma tupla $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, onde valem as seguintes proposições:

1. $\mathcal{P}$ é o conjunto (finito) de todos os textos planos
2. $\mathcal{C}$ é o conjunto (finito) de todos os textos cifrados
3. $\mathcal{K}$ é o espaço de chaves, o conjunto finito de todas as chaves possíveis
4. Para cada $K \in \mathcal{K}$, existe uma função de criptografia $e_K \in \mathcal{E}$ e uma função de descriptografia $d_K \in \mathcal{D}$. Cada $e_K : \mathcal{P} \to \mathcal{C}$ e $d_K : \mathcal{C} \to \mathcal{P}$ são funções tais que $d_K(e_K(x)) = x$ para todo $x \in \mathcal{P}$

Observação: Se $e_K$ e $d_K$ são iguais (ou facilmente relacionáveis), o sistema é dito "simétrico". Caso contrário, o sistema é "assimétrico" ou "de chave pública".

# Alguns Exemplos de Sistemas Simétricos

Existem alguns criptossistemas simétricos célebres, por exemplo:

- A Cifra de Deslocamento (ou “de César”)
- A Cifra de Permutação
- O One-Time Pad (OTP)
- O DES (Data Encryption Standard)
- O AES (Advanced Encryption Standard)
- Etc.

# O Problema da Chave Simétrica

A utilização de sistemas baseados em chaves simétricas traz um grande problema: como transmitir essa chave entre as partes?

# O Problema da Chave Simétrica

A utilização de sistemas baseados em chaves simétricas traz um grande problema: como transmitir essa chave entre as partes?

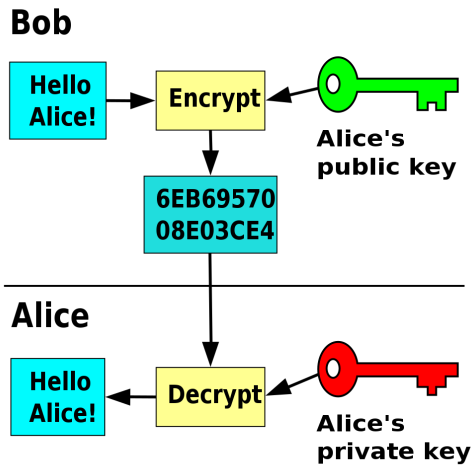
1. Você precisa de um canal previamente seguro

# O Problema da Chave Simétrica

A utilização de sistemas baseados em chaves simétricas traz um grande problema: como transmitir essa chave entre as partes?

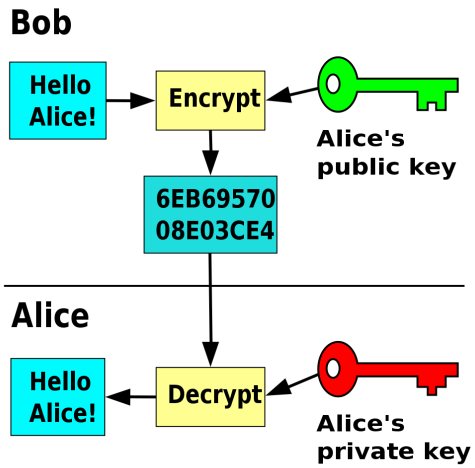
1. Você precisa de um canal previamente seguro
2. Em alguns sistemas (como o OTP), o tamanho da chave é igual ao tamanho da mensagem

# A Solução: Criptografia de Chave Pública

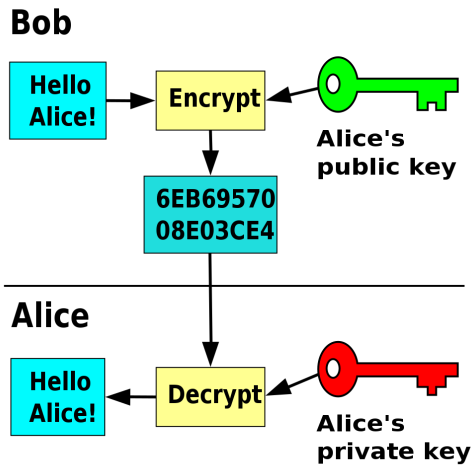


# A Solução: Criptografia de Chave Pública

1. Uma das partes gera um par de chaves: uma chave pública e uma chave privada

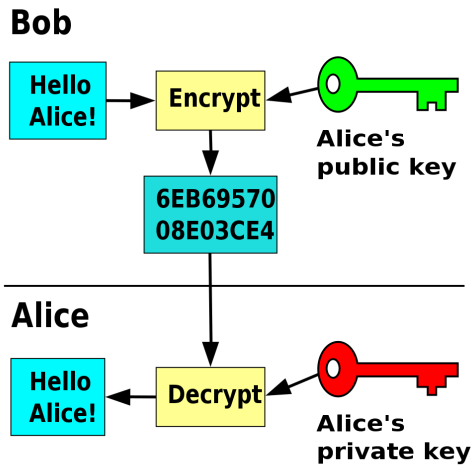


# A Solução: Criptografia de Chave Pública



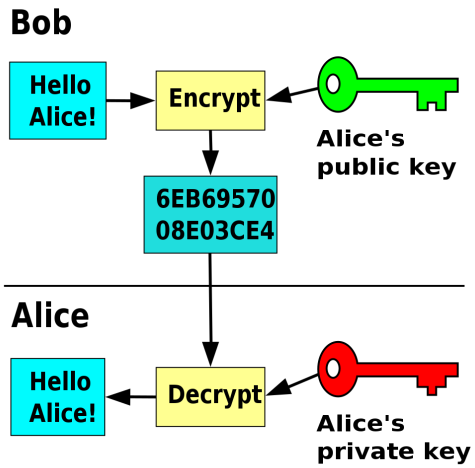

1. Uma das partes gera um par de chaves: uma chave pública e uma chave privada
2. A chave pública é usada para cifrar mensagens. A chave privada é usada para decifrar mensagens

# A Solução: Criptografia de Chave Pública



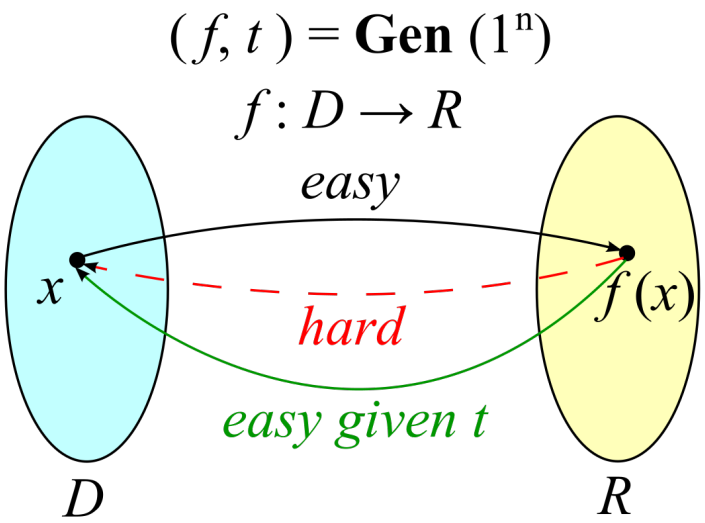

1. Uma das partes gera um par de chaves: uma chave pública e uma chave privada
2. A chave pública é usada para cifrar mensagens. A chave privada é usada para decifrar mensagens
3. Baseado na ideia de "funções alçapão" $\rightarrow$ Uma função que é fácil de calcular em um sentido, mas difícil em outro, a menos que você conheça a "manha"

# A Solução: Criptografia de Chave Pública




1. Uma das partes gera um par de chaves: uma chave pública e uma chave privada
2. A chave pública é usada para cifrar mensagens. A chave privada é usada para decifrar mensagens
3. Baseado na ideia de "funções alçapão" $\rightarrow$ Uma função que é fácil de calcular em um sentido, mas difícil em outro, a menos que você conheça a "manha"
4. Importante: nenhum desses sistemas pode fornecer segurança incondicional, apenas computacional ou provável

# As Funções Alçapão




# Alguns sistemas de chave pública

Alguns dos principais sistemas de chave pública são:

1. RSA
2. ElGamal
3. Certas aplicações de Criptografia de Curva Elíptica (ECC)
4. Cramer–Shoup
5. Merkle–Hellman
6. Etc.

# As Definições de Segurança

Segundo Shannon, existem três tipos de segurança:

# As Definições de Segurança

Segundo Shannon, existem três tipos de segurança:

- Segurança incondicional: o sistema não pode ser quebrado, mesmo com recursos computacionais infinitos

# As Definições de Segurança

Segundo Shannon, existem três tipos de segurança:

- Segurança incondicional: o sistema não pode ser quebrado, mesmo com recursos computacionais infinitos
- Segurança computacional: o melhor algoritmo para quebrar o sistema precisa de pelo menos N operações, onde N é um número grande

# As Definições de Segurança

Segundo Shannon, existem três tipos de segurança:

- Segurança incondicional: o sistema não pode ser quebrado, mesmo com recursos computacionais infinitos
- Segurança computacional: o melhor algoritmo para quebrar o sistema precisa de pelo menos N operações, onde N é um número grande
- Segurança provável: quebrar o sistema significa resolver um problema considerado difícil

# As Definições de Segurança

Segundo Shannon, existem três tipos de segurança:

- Segurança incondicional: o sistema não pode ser quebrado, mesmo com recursos computacionais infinitos
- Segurança computacional: o melhor algoritmo para quebrar o sistema precisa de pelo menos N operações, onde N é um número grande
- Segurança provável: quebrar o sistema significa resolver um problema considerado difícil

"Quebrar um sistema" tem múltiplos significados: pode ser encontrar a chave usada, um algoritmo equivalente para decifrar, alterar uma mensagem enviada, etc. A ideia é a mesma: furar a proposta de segurança.

# Tipos de Ataque

A segurança de um sistema é altamente dependente do tipo de ataque:

- Texto plano conhecido
- Texto plano escolhido
- Homem-no-meio
- Texto cifrado conhecido
- Adulteração de mensagem
- Etc.

# Tipos de Ataque

A segurança de um sistema é altamente dependente do tipo de ataque:

- Texto plano conhecido
- Texto plano escolhido
- Homem-no-meio
- Texto cifrado conhecido
- Adulteração de mensagem
- Etc.

Focaremos no ataque passivo de texto cifrado conhecido, no qual apenas a mensagem cifrada (e demais informações públicas) são conhecidas pelo atacante, que age passivamente, ou seja, não tenta alterar a mensagem em trânsito.

# Sumário

Seja $m \in \mathbb{N}^*$, definimos $\mathbb{Z}_m = \{0, 1, ..., m-1\}$ como um conjunto munido de duas operações, $+$ e $\times$, em cima do qual podemos construir a chamada “Aritmética Modular”.

Seja $m \in \mathbb{N}^*$, definimos $\mathbb{Z}_m = \{0, 1, ..., m-1\}$ como um conjunto munido de duas operações, $+$ e $\times$, em cima do qual podemos construir a chamada "Aritmética Modular".

Dizemos que dois números $a$ e $b$ são "congruentes módulo $m$" se $m$ divide $b - a$:

$$a \equiv b \pmod{m} \iff \exists k \in \mathbb{Z} | (b - a) = k \cdot m$$

**Importante**: não confundir com a notação sem parênteses, que se refere à operação de resto euclideano. Seja $r \in \mathbb{Z}_m$:

$$a \mod m = r \implies \exists k \in \mathbb{Z} | a = k \cdot m + r$$

Seja $m \in \mathbb{N}^*$, definimos $\mathbb{Z}_m = \{0, 1, ..., m-1\}$ como um conjunto munido de duas operações, $+$ e $\times$, em cima do qual podemos construir a chamada "Aritmética Modular".

Dizemos que dois números $a$ e $b$ são "congruentes módulo $m$" se $m$ divide $b - a$:

$$a \equiv b \pmod{m} \iff \exists k \in \mathbb{Z} | (b - a) = k \cdot m$$

**Importante**: não confundir com a notação sem parênteses, que se refere à operação de resto euclideano. Seja $r \in \mathbb{Z}_m$:

$$a \mod m = r \implies \exists k \in \mathbb{Z} | a = k \cdot m + r$$

Podemos ligar as duas operações: $a \mod m = r \implies a \equiv r \pmod{m}$

Se definirmos as operações $+$ e $\times$ como a soma e multiplicação usuais módulo $m$, temos as seguintes propriedades:

- Para $+$: Fechamento, comutatividade, associatividade, identidade aditiva $0 \in \mathbb{Z}_m$, inverso aditivo $-a \in \mathbb{Z}_m$ (o que permite a definição de uma operação diferença)
- Para $\times$: Fechamento, comutatividade, associatividade, identidade multiplicativa $1 \in \mathbb{Z}_m$
- Para $+$ e $\times$: Distributividade (esquerda e direita)

Se definirmos as operações $+$ e $\times$ como a soma e multiplicação usuais módulo $m$, temos as seguintes propriedades:

- Para $+$: Fechamento, comutatividade, associatividade, identidade aditiva $0 \in \mathbb{Z}_m$, inverso aditivo $-a \in \mathbb{Z}_m$ (o que permite a definição de uma operação diferença)
- Para $\times$: Fechamento, comutatividade, associatividade, identidade multiplicativa $1 \in \mathbb{Z}_m$
- Para $+$ e $\times$: Distributividade (esquerda e direita)

Essas propriedades fazem de $\mathbb{Z}_m$ um tipo especial de estrutura algébrica: $\mathbb{Z}_m$ é um **anel**.

# A função $\phi$

Definição da função totiente de Euler $\phi$: seja $m \in \mathbb{N}^*$, definimos $\phi(m)$ como o número de elementos $a \in \mathbb{Z}_m$ tal que $mdc(m, a) = 1$ (ou seja, o número de co-primos de $m$ em $\mathbb{Z}_m$).

Se você conhecer a fatoração de $m$, a função $\phi$ é facilmente calculável:

# A função $\phi$

Definição da função totiente de Euler $\phi$: seja $m \in \mathbb{N}^*$, definimos $\phi(m)$ como o número de elementos $a \in \mathbb{Z}_m$ tal que $mdc(m, a) = 1$ (ou seja, o número de co-primos de $m$ em $\mathbb{Z}_m$).

Se você conhecer a fatoração de $m$, a função $\phi$ é facilmente calculável:

Teorema: Sendo:

$$m = \prod_{i=1}^{n} p_i^{e_i}$$

Onde $p_i$ são primos distintos e $e_i > 0$, temos que:

$$\phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})$$

# A Questão dos Inversos

Teorema: o elemento não-nulo $a \in \mathbb{Z}_m$ possui inverso $a^{-1} \in \mathbb{Z}_m$ tal que $a^{-1} \times a \equiv 1 \pmod{m}$ se e somente se $mdc(a, m) = 1$.

# A Questão dos Inversos

Teorema: o elemento não-nulo $a \in \mathbb{Z}_m$ possui inverso $a^{-1} \in \mathbb{Z}_m$ tal que $a^{-1} \times a \equiv 1 \pmod{m}$ se e somente se $mdc(a, m) = 1$.

Corolário: se $m$ for um número primo, todo elemento não nulo de $\mathbb{Z}_m$ possui inverso em $\mathbb{Z}_m$ (nesse caso, $\mathbb{Z}_m$ é promovido ao status de **corpo**)

Por outro lado, podemos construir o **grupo abeliano** $\mathbb{Z}_m^*$, constituído de todos os elementos $a < m, a \in \mathbb{N}$ tal que $mdc(a, m) = 1$.

# O Algoritmo de Euclides Estendido

Sejam dois números naturais não nulos $a$ e $b$. O algoritmo de Euclides estendido calcula, com complexidade $O(log(min(a, b))$, os números inteiros $r$, $s$ e $t$ tais que:

$$\begin{cases} mdc(a, b) = r \\ s \cdot a + t \cdot b = r \end{cases}$$

Por que esse algoritmo importa? Ele permite que calculemos inversos de forma rápida!

# O Algoritmo de Euclides Estendido

Considere o problema de achar $t \in \mathbb{Z}_m$ tal que $t \times b \equiv 1 \pmod{m}$.

# O Algoritmo de Euclides Estendido

Considere o problema de achar $t \in \mathbb{Z}_m$ tal que $t \times b \equiv 1 \pmod{m}$.

Como dito, esse problema só tem solução se $mdc(m, b) = 1$.
Aplicando o algoritmo em $(m, b)$:

$$1 = mdc(m, b) = s \cdot m + t \cdot b \implies 1 \equiv t \times b \pmod{m}$$

# O Algoritmo de Euclides Estendido

Considere o problema de achar $t \in \mathbb{Z}_m$ tal que $t \times b \equiv 1 \pmod{m}$.

Como dito, esse problema só tem solução se $mdc(m, b) = 1$.
Aplicando o algoritmo em $(m, b)$:

$$1 = mdc(m, b) = s \cdot m + t \cdot b \implies 1 \equiv t \times b \pmod{m}$$

Se quisermos achar $b^{-1} \in \mathbb{Z}_m$, basta tomar $t \mod m = b^{-1}$

Teorema de Euler: sejam $n, a \in \mathbb{N}^*$ tais que $mdc(a, n) = 1$, temos:

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

# Teorema de Euler e Teorema de Fermat

Teorema de Euler: sejam $n, a \in \mathbb{N}^*$ tais que $mdc(a, n) = 1$, temos:

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

Teorema de Fermat (Corolário do Teorema de Euler): sejam $p, a \in \mathbb{N}^*$ tais que $p$ é primo e $mdc(a, p) = 1$, temos:

$$a^{(p-1)} \equiv 1 \pmod{p}$$

# Sumário

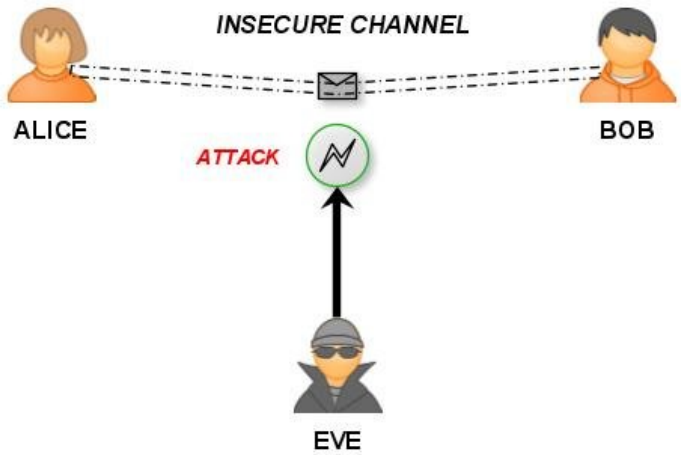
# O Sistema RSA

O criptossistema de chave pública mais conhecido é o sistema RSA (“Rivest-Shamir-Adleman”). Sua segurança é baseada no problema da fatoração de inteiros.

# O Sistema RSA

O criptossistema de chave pública mais conhecido é o sistema RSA (“Rivest-Shamir-Adleman”). Sua segurança é baseada no problema da fatoração de inteiros.

Sejam Alice e Bob duas partes que desejam estabelecer uma comunicação segura, de forma que Bob possa enviar mensagens a Alice, face a um adversário, Eva. O Sistema RSA consiste em duas partes: a geração das chaves e a comunicação em si.



# Geração de chaves em RSA

Alice deve, primeiramente, gerar um par de chaves

# Geração de chaves em RSA

Alice deve, primeiramente, gerar um par de chaves

1. Alice gera aleatoriamente dois números primos secretos $p$ e $q$. O produto $N = pq$ é calculado

# Geração de chaves em RSA

Alice deve, primeiramente, gerar um par de chaves

1. Alice gera aleatoriamente dois números primos secretos $p$ e $q$. O produto $N = pq$ é calculado
2. Alice calcula $\phi(n)$. Como ela sabe a decomposição de $N$, temos $\phi(N) = (p-1)(q-1)$

# Geração de chaves em RSA

Alice deve, primeiramente, gerar um par de chaves

1. Alice gera aleatoriamente dois números primos secretos $p$ e $q$. O produto $N = pq$ é calculado
2. Alice calcula $\phi(n)$. Como ela sabe a decomposição de $N$, temos $\phi(N) = (p-1)(q-1)$
3. Alice calcula um número (aleatório ou não) $a$ tal que $mdc(a, \phi(N)) = 1$

# Geração de chaves em RSA

Alice deve, primeiramente, gerar um par de chaves

1. Alice gera aleatoriamente dois números primos secretos $p$ e $q$. O produto $N = pq$ é calculado
2. Alice calcula $\phi(n)$. Como ela sabe a decomposição de $N$, temos $\phi(N) = (p-1)(q-1)$
3. Alice calcula um número (aleatório ou não) $a$ tal que $mdc(a, \phi(N)) = 1$
4. Alice calcula $b$ tal que $a \times b \equiv 1 \pmod{\phi(N)}$, usando o algoritmo de Euclides estendido (problema dos inversos)

# Geração de chaves em RSA

Alice deve, primeiramente, gerar um par de chaves

1. Alice gera aleatoriamente dois números primos secretos $p$ e $q$. O produto $N = pq$ é calculado
2. Alice calcula $\phi(n)$. Como ela sabe a decomposição de $N$, temos $\phi(N) = (p-1)(q-1)$
3. Alice calcula um número (aleatório ou não) $a$ tal que $mdc(a, \phi(N)) = 1$
4. Alice calcula $b$ tal que $a \times b \equiv 1 \pmod{\phi(N)}$, usando o algoritmo de Euclides estendido (problema dos inversos)
5. Alice divulga a chave pública $(N, a)$ e guarda a chave privada $(p, q, b)$

# Troca de Mensagens em RSA

A troca de mensagens funciona da seguinte forma:

# Troca de Mensagens em RSA

A troca de mensagens funciona da seguinte forma:

1. Bob, utilizando a chave pública $(N, a)$ e um texto plano $x \in \mathbb{Z}_N$, gera uma mensagem cifrada $y = x^a \mod N$

# Troca de Mensagens em RSA

A troca de mensagens funciona da seguinte forma:

1. Bob, utilizando a chave pública $(N, a)$ e um texto plano $x \in \mathbb{Z}_N$, gera uma mensagem cifrada $y = x^a \mod N$
2. Alice recebe a mensagem cifrada $y$, que é decodificada usando a chave privada $x' = y^b \mod N$

# Exemplo RSA

Geração de Chaves:

# Exemplo RSA

Geração de Chaves:

1. Alice escolhe aleatoriamente dois primos: $p = 7639$ e $q = 5981$. Alice calcula $N = pq = 45688859$

# Exemplo RSA

Geração de Chaves:

1. Alice escolhe aleatoriamente dois primos: $p = 7639$ e $q = 5981$. Alice calcula $N = pq = 45688859$
2. Alice calcula $\phi(N) = (p-1) * (q-1) = 45675240$

# Exemplo RSA

Geração de Chaves:

1. Alice escolhe aleatoriamente dois primos: $p = 7639$ e $q = 5981$. Alice calcula $N = pq = 45688859$
2. Alice calcula $\phi(N) = (p-1) * (q-1) = 45675240$
3. Alice escolhe o número $a = 17$. Podemos ver que $mdc(a, \phi(N)) = 1$

# Exemplo RSA

Geração de Chaves:

1. Alice escolhe aleatoriamente dois primos: $p = 7639$ e $q = 5981$. Alice calcula $N = pq = 45688859$
2. Alice calcula $\phi(N) = (p-1) * (q-1) = 45675240$
3. Alice escolhe o número $a = 17$. Podemos ver que $mdc(a, \phi(N)) = 1$
4. Alice, com o algoritmo de Euclides estendido, calcula $b = 16120673$. Podemos verificar que $a \times b \equiv 1 \pmod{\phi(N)}$

# Exemplo RSA

Geração de Chaves:

1. Alice escolhe aleatoriamente dois primos: $p = 7639$ e $q = 5981$. Alice calcula $N = pq = 45688859$
2. Alice calcula $\phi(N) = (p-1) * (q-1) = 45675240$
3. Alice escolhe o número $a = 17$. Podemos ver que $mdc(a, \phi(N)) = 1$
4. Alice, com o algoritmo de Euclides estendido, calcula $b = 16120673$. Podemos verificar que $a \times b \equiv 1 \pmod{\phi(N)}$
5. Alice publica $(N, a)$ e guarda $(p, q, b)$

# Exemplo RSA

Comunicação:

# Exemplo RSA

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$

# Exemplo RSA

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$
2. A mensagem que Bob envia é $y = x^a \mod N = 130102$

# Exemplo RSA

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$
2. A mensagem que Bob envia é $y = x^a \mod N = 130102$
3. Alice recebe $y$ e calcula $x = y^b \mod N = 4805697$. Fazendo o caminho inverso, temos
   $4805679 \rightarrow 0d4805697 \rightarrow 0x495441 \rightarrow 49\ 54\ 41 \xrightarrow{ASCII} ITA$

# Prova de Funcionamento

Para provarmos que o sistema funciona, devemos mostrar que a mensagem $x'$ encontrada por Alice é côngrua módulo $N$ à mensagem enviada por Bob. Por construção, sabemos que:

# Prova de Funcionamento

Para provarmos que o sistema funciona, devemos mostrar que a mensagem $x'$ encontrada por Alice é côngrua módulo $N$ à mensagem enviada por Bob. Por construção, sabemos que:

$$\begin{cases} y \equiv x^a \pmod{N} \\ x' \equiv y^b \pmod{N} \end{cases} \implies x' \equiv x^{a \cdot b} \pmod{N}$$

# Prova de Funcionamento

Para provarmos que o sistema funciona, devemos mostrar que a mensagem $x'$ encontrada por Alice é côngrua módulo $N$ à mensagem enviada por Bob. Por construção, sabemos que:

$$\begin{cases} y \equiv x^a \pmod{N} \\ x' \equiv y^b \pmod{N} \end{cases} \implies x' \equiv x^{a \cdot b} \pmod{N}$$

Também sabemos que:

$$a \times b \equiv 1 \pmod{\phi(N)} \implies a \cdot b = 1 + k\phi(N)$$

# Prova de Funcionamento

Para provarmos que o sistema funciona, devemos mostrar que a mensagem $x'$ encontrada por Alice é côngrua módulo $N$ à mensagem enviada por Bob. Por construção, sabemos que:

$$\begin{cases} y \equiv x^a \pmod{N} \\ x' \equiv y^b \pmod{N} \end{cases} \implies x' \equiv x^{a \cdot b} \pmod{N}$$

Também sabemos que:

$$a \times b \equiv 1 \pmod{\phi(N)} \implies a \cdot b = 1 + k\phi(N)$$

Logo:

$$x' \equiv (x^{\phi(N)})^k x \pmod{N}$$

# Prova de Funcionamento

Se $mdc(x, N) = 1$, podemos aplicar o teorema de Euler:

$$x' \equiv (x^{\phi(N)})^k x \pmod{N} \implies x' \equiv x \pmod{N}$$

# Prova de Funcionamento

Já se $mdc(x, N) \neq 1 \implies x \equiv 0 \pmod{p} \vee x \equiv 0 \pmod{q}$

# Prova de Funcionamento

Já se $mdc(x, N) \neq 1 \implies x \equiv 0 \pmod{p} \vee x \equiv 0 \pmod{q}$

No caso onde a congruência é nula, temos que $x^{a \cdot b} \equiv x \pmod{p, q}$

# Prova de Funcionamento

Já se $mdc(x, N) \neq 1 \implies x \equiv 0 \pmod{p} \vee x \equiv 0 \pmod{q}$

No caso onde a congruência é nula, temos que $x^{a \cdot b} \equiv x \pmod{p, q}$

Se a congruência não é nula, podemos aplicar o teorema de Fermat, já que $\phi(N) = (p-1)(q-1)$: $x^{k(p-1)(q-1)} x \equiv 1 \pmod{p, q}$

# Prova de Funcionamento

Já se $mdc(x, N) \neq 1 \implies x \equiv 0 \pmod{p} \vee x \equiv 0 \pmod{q}$

No caso onde a congruência é nula, temos que $x^{a \cdot b} \equiv x \pmod{p, q}$

Se a congruência não é nula, podemos aplicar o teorema de Fermat, já que $\phi(N) = (p-1)(q-1)$: $x^{k(p-1)(q-1)} x \equiv 1 \pmod{p, q}$

Pelo teorema chinês do resto: $x' \equiv x \pmod{pq}$

# Prova de Funcionamento

Já se $mdc(x, N) \neq 1 \implies x \equiv 0 \pmod{p} \vee x \equiv 0 \pmod{q}$

No caso onde a congruência é nula, temos que $x^{a \cdot b} \equiv x \pmod{p, q}$

Se a congruência não é nula, podemos aplicar o teorema de Fermat, já que $\phi(N) = (p-1)(q-1)$: $x^{k(p-1)(q-1)} x \equiv 1 \pmod{p, q}$

Pelo teorema chinês do resto: $x' \equiv x \pmod{pq}$

Ou seja, não importando o caso: $x' \equiv x \pmod{N}$

Além disso, como há uma operação resto antes de $x'$ e $x \in \mathbb{Z}_N$, temos mais que uma congruência: $\boxed{x' = x}$.

# Prova de Segurança

Imagine que Eva quer realizar um ataque de tipo texto cifrado conhecido, captando a mensagem de Bob para Alice. O problema de Eva é achar a chave privada $b$ tal que $a \times b \equiv 1 \pmod{\phi(N)}$. O que Eva não conhece é o valor de $\phi(N)$, que pode ser obtido de duas formas:

# Prova de Segurança

Imagine que Eva quer realizar um ataque de tipo texto cifrado conhecido, captando a mensagem de Bob para Alice. O problema de Eva é achar a chave privada $b$ tal que $a \times b \equiv 1 \pmod{\phi(N)}$. O que Eva não conhece é o valor de $\phi(N)$, que pode ser obtido de duas formas:

- Por enumeração simples, contando o número de elementos em $\mathbb{Z}_N$ que são primos relativos a N (segurança computacional)

# Prova de Segurança

Imagine que Eva quer realizar um ataque de tipo texto cifrado conhecido, captando a mensagem de Bob para Alice. O problema de Eva é achar a chave privada $b$ tal que $a \times b \equiv 1 \pmod{\phi(N)}$. O que Eva não conhece é o valor de $\phi(N)$, que pode ser obtido de duas formas:

- Por enumeração simples, contando o número de elementos em $\mathbb{Z}_N$ que são primos relativos a N (segurança computacional)
- **Fatorando N em seus componentes primos e aplicando o teorema do cálculo de $\phi(N)$**

# Prova de Segurança

Imagine que Eva quer realizar um ataque de tipo texto cifrado conhecido, captando a mensagem de Bob para Alice. O problema de Eva é achar a chave privada $b$ tal que $a \times b \equiv 1 \pmod{\phi(N)}$. O que Eva não conhece é o valor de $\phi(N)$, que pode ser obtido de duas formas:

- Por enumeração simples, contando o número de elementos em $\mathbb{Z}_N$ que são primos relativos a N (segurança computacional)
- **Fatorando N em seus componentes primos e aplicando o teorema do cálculo de $\phi(N)$**

Podemos ver que quebrar o sistema RSA equivale a resolver o problema da fatoração de um inteiro, para o qual não conhecemos nenhum algoritmo clássico eficiente! (Atenção: não sabemos se não existe um!)
Curiosidade: o maior inteiro já fatorado continha 240 dígitos

# Comentários Sobre a Segurança

O Sistema RSA é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

# Comentários Sobre a Segurança

O Sistema RSA é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se a mensagem $x$ e o número $a$ forem pequenos, o problema se converte em uma simples extração da raiz a-ésima

# Comentários Sobre a Segurança

O Sistema RSA é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se a mensagem $x$ e o número $a$ forem pequenos, o problema se converte em uma simples extração da raiz a-ésima
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada

# Comentários Sobre a Segurança

O Sistema RSA é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se a mensagem $x$ e o número $a$ forem pequenos, o problema se converte em uma simples extração da raiz a-ésima
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Gerar os primos $p$ e $q$ aleatoriamente de forma segura e sem que eles contenham propriedades que facilitem a decomposição de $N$ é um problema bastante complicado

# Comentários Sobre a Segurança

O Sistema RSA é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se a mensagem $x$ e o número $a$ forem pequenos, o problema se converte em uma simples extração da raiz a-ésima
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Gerar os primos $p$ e $q$ aleatoriamente de forma segura e sem que eles contenham propriedades que facilitem a decomposição de $N$ é um problema bastante complicado
- Em raros casos, não há uma injeção na função de criptografia

# Comentários Sobre a Segurança

O Sistema RSA é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se a mensagem $x$ e o número $a$ forem pequenos, o problema se converte em uma simples extração da raiz a-ésima
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Gerar os primos $p$ e $q$ aleatoriamente de forma segura e sem que eles contenham propriedades que facilitem a decomposição de $N$ é um problema bastante complicado
- Em raros casos, não há uma injeção na função de criptografia
- Um atacante pode testar a criptografia de diversas mensagens planas até chegar na que “bate”

# Comentários Sobre a Segurança

O Sistema RSA é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se a mensagem $x$ e o número $a$ forem pequenos, o problema se converte em uma simples extração da raiz a-ésima
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Gerar os primos $p$ e $q$ aleatoriamente de forma segura e sem que eles contenham propriedades que facilitem a decomposição de $N$ é um problema bastante complicado
- Em raros casos, não há uma injeção na função de criptografia
- Um atacante pode testar a criptografia de diversas mensagens planas até chegar na que “bate”
- Etc.

# Formalização

Definição formal do sistema RSA:
Seja $N = pq$, onde $p$ e $q$ são primos. Considere $\mathcal{P} = \mathcal{C} = \mathbb{Z}_N$. Defina o espaço de chaves $\mathcal{K}$ como:

$$\mathcal{K} = \{(N, p, q, a, b) | a \times b \equiv 1 \pmod{\phi(N)}\}$$

Para uma chave $K \in \mathcal{K}$, temos:

$$e_K(x) = x^a \mod N$$

$$d_K(y) = y^b \mod N$$

Onde $x \in \mathcal{P}, y \in \mathcal{C}$ são, respectivamente, a mensagem plana e a mensagem cifrada. Os valores $(N, a)$ constituem a chave pública e os valores $(p, q, b)$ são a chave privada.

# Sumário

# Introdução

Enquanto o sistema RSA se baseia na fatoração de inteiros, o sistema ElGamal é pautado em outro problema: o cálculo de logaritmos discretos.

# O Problema dos Logaritmos Discretos

Lembrete: Seja $p$ um número primo, pode-se denotar $\mathbb{Z}_p^* = \{1, 2, 3, ..., p-1\}$ como o conjunto de todos os naturais menores que $p$ e co-primos de $p$.

# O Problema dos Logaritmos Discretos

Lembrete: Seja $p$ um número primo, pode-se denotar $\mathbb{Z}_p^* = \{1, 2, 3, ..., p-1\}$ como o conjunto de todos os naturais menores que $p$ e co-primos de $p$.

O grupo $\mathbb{Z}_p^*$ é um **grupo multiplicativo**. Além disso, também é um **grupo cíclico** pois existe pelo menos um elemento $g$ do grupo tal que:

$$\{g^i \mod p, 1 \leq i \leq p-1\} = \mathbb{Z}_p^*$$

# O Problema dos Logaritmos Discretos

Lembrete: Seja $p$ um número primo, pode-se denotar $\mathbb{Z}_p^* = \{1, 2, 3, ..., p-1\}$ como o conjunto de todos os naturais menores que $p$ e co-primos de $p$.

O grupo $\mathbb{Z}_p^*$ é um **grupo multiplicativo**. Além disso, também é um **grupo cíclico** pois existe pelo menos um elemento $g$ do grupo tal que:

$$\{g^i \mod p, 1 \leq i \leq p-1\} = \mathbb{Z}_p^*$$

O elemento $g$ é chamado um **gerador** de $\mathbb{Z}_p^*$

# O Problema dos Logaritmos Discretos

Lembrete: Seja $p$ um número primo, pode-se denotar $\mathbb{Z}_p^* = \{1, 2, 3, ..., p-1\}$ como o conjunto de todos os naturais menores que $p$ e co-primos de $p$.

O grupo $\mathbb{Z}_p^*$ é um **grupo multiplicativo**. Além disso, também é um **grupo cíclico** pois existe pelo menos um elemento $g$ do grupo tal que:

$$\{g^i \mod p, 1 \leq i \leq p-1\} = \mathbb{Z}_p^*$$

O elemento $g$ é chamado um **gerador** de $\mathbb{Z}_p^*$

**Importante:** o cálculo de $g$ é um problema à parte. Apesar de não haver uma fórmula geral, ele é facilmente encontrável.

Segue-se o seguinte teorema:

Seja $p$ um número primo tal que $\alpha$ é um gerador do conjunto $\mathbb{Z}_p^*$. Para qualquer $\beta \in \mathbb{Z}_p^*$, existe $x \in \mathbb{Z}_p^*$ tal que $\beta \equiv \alpha^x \pmod{p}$

# O Problema dos Logaritmos Discretos

Com base no teorema anterior, podemos definir o problema do logaritmo discreto:

# O Problema dos Logaritmos Discretos

Com base no teorema anterior, podemos definir o problema do logaritmo discreto:

Definimos $x = \log_\alpha \beta$ como o valor em $\mathbb{Z}_p^*$ tal que $\alpha^x \equiv \beta \pmod{p}$

# O Problema dos Logaritmos Discretos

Com base no teorema anterior, podemos definir o problema do logaritmo discreto:

Definimos $x = \log_\alpha \beta$ como o valor em $\mathbb{Z}_p^*$ tal que $\alpha^x \equiv \beta \pmod{p}$

Importante: não há nenhum algoritmo clássico conhecido que resolva o problema dos logaritmos discretos em tempo polinomial. Se $p$ for grande o suficiente, o problema dos logaritmos discretos se torna intratável.

# Sumário

# O Sistema ElGamal

Definamos o criptossistema ElGamal. Sejam Alice e Bob duas partes que desejam estabelecer uma comunicação segura, de forma que Bob possa enviar mensagens a Alice, face a um adversário, Eva. O Sistema ElGamal consiste em duas partes: a geração das chaves e a comunicação em si.

# Geração de Chaves em ElGamal

Alice deve, primeiramente, gerar um par de chaves:

# Geração de Chaves em ElGamal

Alice deve, primeiramente, gerar um par de chaves:

1. Alice gera, aleatoriamente, um número primo $p$ suficientemente grande. Com esse número, Alice tem implicitamente o grupo $\mathbb{Z}_p^*$, do qual é escolhido um elemento gerador $\alpha$

# Geração de Chaves em ElGamal

Alice deve, primeiramente, gerar um par de chaves:

1. Alice gera, aleatoriamente, um número primo $p$ suficientemente grande. Com esse número, Alice tem implicitamente o grupo $\mathbb{Z}_p^*$, do qual é escolhido um elemento gerador $\alpha$
2. Alice escolhe aleatoriamente um elemento $a$ do conjunto $\mathbb{Z}_p^*$ e calcula $\beta = \alpha^a \mod p$

# Geração de Chaves em ElGamal

Alice deve, primeiramente, gerar um par de chaves:

1. Alice gera, aleatoriamente, um número primo $p$ suficientemente grande. Com esse número, Alice tem implicitamente o grupo $\mathbb{Z}_p^*$, do qual é escolhido um elemento gerador $\alpha$
2. Alice escolhe aleatoriamente um elemento $a$ do conjunto $\mathbb{Z}_p^*$ e calcula $\beta = \alpha^a \mod p$
3. Alice publica a chave $(p, \alpha, \beta)$ e guarda para si a chave privada $a$

# Troca de Mensagens em ElGamal

A troca de mensagens funciona da seguinte forma:

# Troca de Mensagens em ElGamal

A troca de mensagens funciona da seguinte forma:

1. Bob, que agora conhece a chave pública $(p, \alpha, \beta)$, gera um texto plano $x \in \mathbb{Z}_p^*$

# Troca de Mensagens em ElGamal

A troca de mensagens funciona da seguinte forma:

1. Bob, que agora conhece a chave pública $(p, \alpha, \beta)$, gera um texto plano $x \in \mathbb{Z}_p^*$
2. Bob escolhe um elemento aleatório $k \in \mathbb{Z}_p^*$

# Troca de Mensagens em ElGamal

A troca de mensagens funciona da seguinte forma:

1. Bob, que agora conhece a chave pública $(p, \alpha, \beta)$, gera um texto plano $x \in \mathbb{Z}_p^*$
2. Bob escolhe um elemento aleatório $k \in \mathbb{Z}_p^*$
3. Bob gera a primeira parte da mensagem cifrada $y_1 = \alpha^k \mod p$

# Troca de Mensagens em ElGamal

A troca de mensagens funciona da seguinte forma:

1. Bob, que agora conhece a chave pública $(p, \alpha, \beta)$, gera um texto plano $x \in \mathbb{Z}_p^*$
2. Bob escolhe um elemento aleatório $k \in \mathbb{Z}_p^*$
3. Bob gera a primeira parte da mensagem cifrada $y_1 = \alpha^k \mod p$
4. Bob gera a segunda parte da mensagem cifrada $y_2 = x \cdot \beta^k \mod p$

# Troca de Mensagens em ElGamal

A troca de mensagens funciona da seguinte forma:

1. Bob, que agora conhece a chave pública $(p, \alpha, \beta)$, gera um texto plano $x \in \mathbb{Z}_p^*$
2. Bob escolhe um elemento aleatório $k \in \mathbb{Z}_p^*$
3. Bob gera a primeira parte da mensagem cifrada $y_1 = \alpha^k \mod p$
4. Bob gera a segunda parte da mensagem cifrada $y_2 = x \cdot \beta^k \mod p$
5. Bob envia a mensagem $(y_1, y_2)$

# Troca de Mensagens em ElGamal

A troca de mensagens funciona da seguinte forma:

1. Bob, que agora conhece a chave pública $(p, \alpha, \beta)$, gera um texto plano $x \in \mathbb{Z}_p^*$
2. Bob escolhe um elemento aleatório $k \in \mathbb{Z}_p^*$
3. Bob gera a primeira parte da mensagem cifrada $y_1 = \alpha^k \mod p$
4. Bob gera a segunda parte da mensagem cifrada $y_2 = x \cdot \beta^k \mod p$
5. Bob envia a mensagem $(y_1, y_2)$
6. Alice calcula $x' = y_2(y_1^a)^{-1} \mod p$

# Exemplo ElGamal

Geração de Chaves:

# Exemplo ElGamal

Geração de Chaves:

1. Alice escolhe aleatoriamente um primo $p = 345553971$. O valor $\alpha = 2121$ é um gerador desse grupo

# Exemplo ElGamal

Geração de Chaves:

1. Alice escolhe aleatoriamente um primo $p = 345553971$. O valor $\alpha = 2121$ é um gerador desse grupo
2. Alice escolhe $a = 31221$. Temos $\beta = \alpha^a \mod p = 27866523$

# Exemplo ElGamal

Geração de Chaves:

1. Alice escolhe aleatoriamente um primo $p = 345553971$. O valor $\alpha = 2121$ é um gerador desse grupo
2. Alice escolhe $a = 31221$. Temos $\beta = \alpha^a \mod p = 27866523$
3. Alice publica $(p, \alpha, \beta)$ e guarda $a$

# Exemplo ElGamal

Comunicação:

# Exemplo ElGamal

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$

# Exemplo ElGamal

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$
2. Bob escolhe aleatoriamente $k = 21$

# Exemplo ElGamal

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$
2. Bob escolhe aleatoriamente $k = 21$
3. Bob gera $y_1 = \alpha^k \mod p = 24780864$, a primeira parte da mensagem

# Exemplo ElGamal

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$
2. Bob escolhe aleatoriamente $k = 21$
3. Bob gera $y_1 = \alpha^k \mod p = 24780864$, a primeira parte da mensagem
4. Bob calcula $y_2 = x \cdot \beta^k \mod p = 18310053$, a segunda parte da mensagem cifrada

# Exemplo ElGamal

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$
2. Bob escolhe aleatoriamente $k = 21$
3. Bob gera $y_1 = \alpha^k \mod p = 24780864$, a primeira parte da mensagem
4. Bob calcula $y_2 = x \cdot \beta^k \mod p = 18310053$, a segunda parte da mensagem cifrada
5. Bob envia a mensagem $(y_1, y_2)$

# Exemplo ElGamal

Comunicação:

1. Bob deseja transmitir a mensagem “ITA”. Temos $ITA \xrightarrow{ASCII,hex} 49\ 54\ 41 \rightarrow 0x495441 \rightarrow 0d4805697$. Logo, $x = 4805697$
2. Bob escolhe aleatoriamente $k = 21$
3. Bob gera $y_1 = \alpha^k \mod p = 24780864$, a primeira parte da mensagem
4. Bob calcula $y_2 = x \cdot \beta^k \mod p = 18310053$, a segunda parte da mensagem cifrada
5. Bob envia a mensagem $(y_1, y_2)$
6. Alice recebe $(y_1, y_2)$ e calcula $x' = y_2(y_1^a)^{-1} \mod p = 4805697$. Fazendo o caminho inverso, temos $4805679 \rightarrow 0d4805697 \rightarrow 0x495441 \rightarrow 49\ 54\ 41 \xrightarrow{ASCII} ITA$

# Prova de Funcionamento

Devemos provar que $x' \equiv x \pmod{p}$ para mostrar que o sistema funciona. Da mensagem recebida, temos:

$$x' \equiv y_2(y_1^a)^{-1} \pmod{p} \implies x' \equiv x \cdot \beta^k(\alpha^{a \cdot k})^{-1} \pmod{p}$$

# Prova de Funcionamento

Devemos provar que $x' \equiv x \pmod{p}$ para mostrar que o sistema funciona. Da mensagem recebida, temos:

$$x' \equiv y_2(y_1^a)^{-1} \pmod{p} \implies x' \equiv x \cdot \beta^k(\alpha^{a \cdot k})^{-1} \pmod{p}$$

Porém, por construção: $\alpha^a \equiv \beta \pmod{p}$. Logo:

# Prova de Funcionamento

Devemos provar que $x' \equiv x \pmod{p}$ para mostrar que o sistema funciona. Da mensagem recebida, temos:

$$x' \equiv y_2(y_1^a)^{-1} \pmod{p} \implies x' \equiv x \cdot \beta^k(\alpha^{a \cdot k})^{-1} \pmod{p}$$

Porém, por construção: $\alpha^a \equiv \beta \pmod{p}$. Logo:

$$x' \equiv x \cdot \beta^k(\beta^k)^{-1} \pmod{p} \implies x' \equiv x \pmod{p}$$

# Prova de Funcionamento

Devemos provar que $x' \equiv x \pmod{p}$ para mostrar que o sistema funciona. Da mensagem recebida, temos:

$$x' \equiv y_2(y_1^a)^{-1} \pmod{p} \implies x' \equiv x \cdot \beta^k(\alpha^{a \cdot k})^{-1} \pmod{p}$$

Porém, por construção: $\alpha^a \equiv \beta \pmod{p}$. Logo:

$$x' \equiv x \cdot \beta^k(\beta^k)^{-1} \pmod{p} \implies x' \equiv x \pmod{p}$$

Além disso, como há uma operação resto antes de $x'$ e $x \in \mathbb{Z}_p^*$, temos mais que uma congruência: $\boxed{x' = x}$.

# Prova de Segurança

Imagine que Eva quer realizar um ataque de tipo texto cifrado conhecido, captando a mensagem de Bob para Alice. O problema de Eva é achar a chave privada $a$ tal que $\alpha^a \equiv \beta \pmod{p}$.

# Prova de Segurança

Imagine que Eva quer realizar um ataque de tipo texto cifrado conhecido, captando a mensagem de Bob para Alice. O problema de Eva é achar a chave privada $a$ tal que $\alpha^a \equiv \beta \pmod{p}$.

Podemos ver que quebrar o sistema ElGamal corresponde a resolver o problema do logaritmo discreto, considerado intratável. O sistema é seguro!

# Comentários Sobre a Segurança

O Sistema ElGamal também é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

# Comentários Sobre a Segurança

O Sistema ElGamal também é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se o gerador $\alpha$ for pequeno e o primo *p* for grande, o logaritmo discreto pode se converter em um logaritmo normal

# Comentários Sobre a Segurança

O Sistema ElGamal também é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se o gerador $\alpha$ for pequeno e o primo $p$ for grande, o logaritmo discreto pode se converter em um logaritmo normal
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada

# Comentários Sobre a Segurança

O Sistema ElGamal também é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se o gerador $\alpha$ for pequeno e o primo *p* for grande, o logaritmo discreto pode se converter em um logaritmo normal
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Se a chave *a* for pequena, ela pode ser descoberta por tentativa e erro

# Comentários Sobre a Segurança

O Sistema ElGamal também é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se o gerador $\alpha$ for pequeno e o primo *p* for grande, o logaritmo discreto pode se converter em um logaritmo normal
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Se a chave *a* for pequena, ela pode ser descoberta por tentativa e erro
- Em raros casos, não há uma injeção na função de criptografia

# Comentários Sobre a Segurança

O Sistema ElGamal também é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se o gerador $\alpha$ for pequeno e o primo *p* for grande, o logaritmo discreto pode se converter em um logaritmo normal
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Se a chave *a* for pequena, ela pode ser descoberta por tentativa e erro
- Em raros casos, não há uma injeção na função de criptografia
- Um atacante pode testar a criptografia de diversas mensagens planas até chegar na que “bate”

# Comentários Sobre a Segurança

O Sistema ElGamal também é notoriamente complicado de se implementar e algumas falhas, que foram ignoradas, reduzem a sua segurança:

- Se o gerador $\alpha$ for pequeno e o primo *p* for grande, o logaritmo discreto pode se converter em um logaritmo normal
- Um adversário ativo pode alterar as mensagens enviadas por Bob sem que Alice perceba, sem que para isso ele precise descobrir a chave provada
- Se a chave *a* for pequena, ela pode ser descoberta por tentativa e erro
- Em raros casos, não há uma injeção na função de criptografia
- Um atacante pode testar a criptografia de diversas mensagens planas até chegar na que “bate”
- Etc.

# Formalização

Definição formal do sistema ElGamal:
Seja $p$ um primo grande o suficiente para que o problema do logaritmo discreto em $\mathbb{Z}_p^*$ seja intratável. Seja $\alpha$ um elemento gerador em $\mathbb{Z}_p^*$. Considere $\mathcal{P} = \mathbb{Z}_p^*$ e $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$. Defina o espaço de chaves $\mathcal{K}$ como:

$$\mathcal{K} = \{(p, \alpha, a, \beta) | \beta \equiv \alpha^a \pmod{p}\}$$

Para uma chave $K \in \mathcal{K}$ e um número secreto $k \in Z_p^*$, temos:

$$e_K(x, k) = (y_1, y_2) | y_1 = \alpha^k \mod p,\ y_2 = x\beta^k \mod p$$

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \mod p$$

Onde $x \in \mathcal{P}, (y_1, y_2) \in \mathcal{C}$ são, respectivamente, a mensagem plana e a mensagem cifrada. Os valores $(p, \alpha, \beta)$ constituem a chave pública e o valor $a$ é a chave privada.