

# Trabalho de Criptografia Assimétrica com RSA

**Aluno:** Mateus Almeida Rondon

**Disciplina:** Segurança em Sistemas de Computação

**Prof:** Rafael Simões

Cuiabá - MT

2025

## Trabalho de Criptografia Assimétrica com RSA

<b>1. Introdução</b>	<b>3</b>
<b>2. Fundamentação Teórica</b>	<b>3</b>
2.1 O Algoritmo RSA	3
2.2 Crivo de Eratóstenes	3
<b>3. Descrição Detalhada da Implementação</b>	<b>4</b>
3.1 Estrutura do Código	4
3.2 Lógica de Conversão e Cifragem	4
<b>4. Exemplos de Execução e Resultados</b>	<b>4</b>
<b>5. Conclusão</b>	<b>5</b>
<b>6. Referências Bibliográficas</b>	<b>6</b>

## 1. Introdução

A segurança da informação é um pilar fundamental nos sistemas computacionais modernos. A criptografia assimétrica, resolve problemas críticos de distribuição de chaves que a criptografia simétrica tradicional não consegue endereçar adequadamente.

O objetivo deste trabalho é implementar de forma funcional o algoritmo **RSA** (Rivest-Shamir-Adleman), um dos sistemas de criptografia assimétrica mais utilizados no mundo. O projeto abrange desde a base matemática, com a geração de números primos via **Crivo de Eratóstenes**, até a aplicação prática na cifragem e decifragem de mensagens textuais, convertendo-as para base hexadecimal conforme especificado os requisitos.

## 2. Fundamentação Teórica

### 2.1 O Algoritmo RSA

Desenvolvido em 1977 no MIT, o RSA baseia sua segurança na dificuldade computacional de fatorar números inteiros grandes. O sistema utiliza um par de chaves:

- **Chave Pública** ( $e, n$ ): Utilizada para cifrar a mensagem. Pode ser distribuída livremente.
- **Chave Privada** ( $d, n$ ): Utilizada para decifrar. Deve ser mantida em segredo absoluto.

A segurança reside no fato de que, embora  $n$  seja público (sendo  $n = p \times q$ ), descobrir os fatores primos  $p$  e  $q$  é um problema intratável para números suficientemente grandes.

### 2.2 Crivo de Eratóstenes

Para a geração de chaves RSA, são necessários números primos. O Crivo de Eratóstenes é um algoritmo eficiente e antigo (c. 240 a.C.) para encontrar todos os números primos até um limite  $L$ . O funcionamento consiste em listar inteiros de 2 até  $L$  e iterativamente marcar os múltiplos de cada primo encontrado como números compostos. Os números não marcados ao final do processo são primos. Sua complexidade é  $O(n \log \log n)$ , sendo muito eficiente para os limites propostos neste trabalho acadêmico.

## 3. Descrição Detalhada da Implementação

A solução foi desenvolvida na linguagem **Python** devido à sua clareza sintática e facilidade de manipulação de grandes inteiros. O código foi estruturado em funções modulares para garantir legibilidade e reaproveitamento.

### 3.1 Estrutura do Código

- `crivo_de_eratostenes(limite)`: Implementa a lógica de eliminação de múltiplos. Retorna uma lista de primos.
- `gerar_chaves()`:
  - Invoca o Crivo para obter candidatos a  $p$  e  $q$ .
  - Seleciona aleatoriamente dois primos distintos (garantindo que  $n > 255$  para suportar a tabela ASCII).
  - Calcula  $\phi(n) = (p-1)(q-1)$ .
  - Calcula  $e$  tal que  $\text{mdc}(e, \phi(n)) = 1$ .
  - Calcula  $d$  (inverso modular) usando o Algoritmo de Euclides Estendido.
- `cifrar(msg, pub_key)`: Converte cada caractere da mensagem em seu valor inteiro (ASCII) e aplica a fórmula  $C = M^e \pmod n$ .
- `decifrar(msg_cifrada, priv_key)`: Aplica a fórmula  $M = C^d \pmod n$  para recuperar o valor inteiro e o converte de volta para caractere.
- 

### 3.2 Lógica de Conversão e Cifragem

Seguindo as especificações, a mensagem inserida pelo usuário é primeiramente convertida para visualização em hexadecimal. Internamente, o algoritmo processa o valor numérico (ordinal) de cada caractere.

Exemplo lógico:

1. Letra 'A' -> Valor ASCII 65.
2. Cifragem:  $65^e \pmod n = X$ .
3. Decifragem:  $X^d \pmod n = 65$  -> Letra 'A'.

## 4. Exemplos de Execução e Resultados

Abaixo apresenta-se um log real de execução do programa desenvolvido:

### Cenário de Teste:

- Mensagem: Instituto Federal

### Saída Obtida:

Python

--- GERADOR DE CHAVES RSA (com Crivo de Eratóstenes) ---

[1] Primos gerados e selecionados:

p = 101

q = 113

[2] Cálculos Auxiliares:

n (Módulo) = 11413

phi(n) = 11200

[3] Chaves Geradas:

Chave Pública (e, n): (3, 11413)

Chave Privada (d, n): (7467, 11413)

[4] Representação Hexadecimal (Pré-codificação):

49 6e 73 74 69 74 75 74 6f 46 65 64 65 72 61 6c

[5] Mensagem Criptografada:

[8440, 2661, 6265, 8769, 5849, 8769, 8740, 8769, 10756, 1721, 6289, 7474, 6289, 3965, 7622, 10041]

[6] Mensagem Decifrada:

InstitutoFederal

SUCESSO: A mensagem decifrada corresponde à original.

O resultado demonstra a eficácia do algoritmo, onde o texto foi recuperado com total integridade após o ciclo de criptografia e descriptografia.

### 5. Conclusão

O trabalho permitiu a aplicação prática de conceitos teóricos complexos de Teoria dos Números. A implementação do Crivo de Eratóstenes mostrou-se eficaz para a geração dos primos necessários, e a aritmética modular do RSA garantiu a confidencialidade da mensagem.

Ficou evidente a importância da escolha de números primos grandes. Em nossa implementação didática, primos pequenos funcionam, mas em ambientes reais, primos com centenas de dígitos são necessários para evitar ataques de fatoração. O projeto cumpriu todos os objetivos propostos: geração de chaves, pré-codificação hexadecimal e o ciclo completo de cifragem/decifragem.

## 6. Referências Bibliográficas

7. ASSIS, T. S. *Criptografia RSA: Teoria e Prática com Python*. Rio Grande: FURG, 2024.
8. COUTINHO, S. C. *Números Inteiros e Criptografia RSA*. Rio de Janeiro: IMPA, 2014.
9. RIVEST, R.; SHAMIR, A.; ADLEMAN, L. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM, 1978.