

Aula de Criptografia RSA

Visão Geral


Este projeto é uma implementação educacional do algoritmo de criptografia RSA em Rust, desenvolvido para fins didáticos em aulas de segurança da informação e criptografia.

⚠️ AVISO IMPORTANTE: Esta implementação é apenas para fins educacionais. NÃO deve ser usada em sistemas de produção!

Características

- ☒ Implementação completa do algoritmo RSA do zero
- ☒ Geração de números primos usando Miller-Rabin
- ☒ Geração automática de pares de chaves públicas/privadas
- ☒ Criptografia e descriptografia de mensagens de texto
- ☒ Interface interativa com explicações passo-a-passo
- ☒ Código totalmente comentado e documentado
- ☒ Demonstração visual do processo completo

Tecnologias Utilizadas

- **Linguagem:** Rust 
- **Dependências:**
 - `num-bigint`: Manipulação de números grandes
 - `rand`: Geração de números aleatórios

Instalação e Execução

Pré-requisitos

- [Rust](#) instalado no sistema
- Cargo (incluído com Rust)

Passos para executar

1. Clone o repositório:

```
git clone <url-do-repositorio>
cd aula_cyber_algoritmo_criptografia
```

2. Compile o projeto:

```
cargo build
```

3. Execute o programa:

```
cargo run
```

4. Siga as instruções na tela:

- Digite uma palavra para ser criptografada
- Observe o processo completo de geração de chaves
- Veja a criptografia e descriptografia em ação

Como Usar

Exemplo de Execução

```
📖 DEMONSTRAÇÃO DE CRIPTOGRAFIA RSA
=====

Digite uma palavra para criptografar:
> HELLO

📝 Palavra a ser criptografada: "HELLO"

🔑 Gerando chaves RSA de 512 bits...
  Passo 1: Gerando números primos p e q...
    ✓ p gerado: 256 bits
    ✓ q gerado: 256 bits
    ...
```

Funcionalidades

- **Entrada interativa:** Digite qualquer palavra
- **Visualização do processo:** Veja cada etapa sendo executada
- **Verificação automática:** O programa confirma que a descriptografia funciona
- **Explicações didáticas:** Cada passo é explicado detalhadamente

Estrutura do Projeto

```
aula_cyber_algoritmo_criptografia/
├── src/
│   └── main.rs          # Código principal do RSA
├── Cargo.toml           # Configuração e dependências
├── README.md            # Esta documentação
└── criptografia_rsa.md  # Explicação detalhada do algoritmo
```

Principais Funções

Geração de Chaves

- `gerar_chaves_rsa(bits)` - Gera par de chaves RSA
- `gerar_primo(bits)` - Gera números primos seguros
- `eh_primo(n, k)` - Teste de primalidade Miller-Rabin

Criptografia

- `criptografar_rsa(mensagem, chave)` - Criptografa texto
- `descriptografar_rsa(cripto, chave)` - Descriptografa texto
- `exponenciacao_modular(base, exp, mod)` - Exponenciação eficiente

Utilitários

- `algoritmo_euclidiano_estendido()` - Para inverso modular
- `string_para_numeros()` - Conversão texto → números
- `numeros_para_string()` - Conversão números → texto

Objetivos Educacionais

Este projeto foi desenvolvido para ensinar:

1. Conceitos de Criptografia Assimétrica

- Diferença entre chaves públicas e privadas
- Conceito de par de chaves matematicamente relacionadas

2. Fundamentos Matemáticos

- Aritmética modular
- Números primos e sua importância
- Função totiente de Euler $\varphi(n)$
- Algoritmo euclidiano estendido

3. Segurança Computacional

- Problema da fatoração de números grandes
- Importância do tamanho das chaves
- Geração segura de números aleatórios

4. Implementação Prática

- Como algoritmos teóricos se tornam código
- Desafios de implementação (números grandes, eficiência)
- Boas práticas de programação segura

Aspectos de Segurança Abordados

☒ Implementados para fins educacionais:

- Geração de números primos verdadeiramente aleatórios
- Teste de primalidade confiável (Miller-Rabin)
- Exponenciação modular eficiente
- Cálculo correto do inverso modular

⚠ Limitações desta implementação:

- **Tamanho de chave pequeno** (512 bits - apenas para demonstração)
- **Sem padding** (vulnerable a ataques em produção)
- **Sem proteções contra side-channel attacks**
- **Números primos não validados para uso criptográfico real**

📖 Documentação Adicional

Para uma explicação detalhada do algoritmo RSA, incluindo a matemática por trás do método, consulte:

- [criptografia_rsa.md](#) - Explicação técnica completa

🎓 Para Educadores

Sugestões de Uso em Aula

1. Introdução Teórica (30 min)

- Apresentar conceitos de criptografia assimétrica
- Explicar o problema matemático por trás do RSA

2. Demonstração Prática (20 min)

- Executar o código com diferentes palavras
- Mostrar como o tamanho da chave afeta a segurança

3. Análise do Código (40 min)

- Revisar funções importantes
- Discutir escolhas de implementação

4. Discussão de Segurança (20 min)

- Limitações desta implementação
- Diferenças para uso em produção
- Ataques conhecidos e contramedidas

Exercícios Sugeridos

1. Modificar o tamanho das chaves e observar o impacto na performance
2. Tentar "quebrar" chaves pequenas (128 bits) por força bruta
3. Implementar verificação adicional de segurança dos primos
4. Comparar performance com implementações de bibliotecas profissionais

🤝 Contribuições

Este é um projeto educacional. Sugestões para melhorar o aspecto didático são bem-vindas:

- Melhorias na explicação dos conceitos
- Exemplos adicionais
- Exercícios práticos
- Correções de bugs ou otimizações

Licença

Este projeto é disponibilizado para fins educacionais. Use livremente em contextos acadêmicos e de ensino.

Recursos Adicionais

Leituras Recomendadas

- "Applied Cryptography" - Bruce Schneier
- "Introduction to Modern Cryptography" - Katz & Lindell
- RFC 3447 - PKCS #1: RSA Cryptography Specifications

Links Úteis

- [RSA Algorithm - Wikipedia](#)
- [Rust Cryptography Libraries](#)
- [NIST Guidelines for Key Management](#)

Desenvolvido com  e  para fins educacionais

"A melhor forma de entender criptografia é implementá-la do zero"