Trabalho 2 – Gerador e Verificador de Assinaturas RSA com OAEP

Autores:

Pedro Brum Tristão de Castro - 202067470

Mateus Oliveira Santos – 221029150

Link para o repositório:

https://github.com/mateusap1/rsa

1. Introdução

Este trabalho tem como objetivo implementar um sistema de assinatura digital baseado no algoritmo RSA, com uso de OAEP como esquema de preenchimento (padding), além de um algoritmo de assinatura digital. A assinatura digital garante a integridade e autenticidade de arquivos, sendo uma das aplicações mais importantes da criptografia assimétrica. Além disso, utiliza-se a função de hash SHA-3 para obter o resumo da mensagem a ser assinada.

2. Fundamentos Teóricos

2.1 Testes de Miller-Rabin

O algoritmo de Miller-Rabin é um teste de primalidade probabilístico eficiente, usado para verificar se um número é primo. Ele reescreve n−1 como 2s·d, escolhe bases aleatórias a, e realiza testes modulares. Se um teste falha, o número é composto; se passa em várias rodadas, é "provavelmente primo" com alta confiança. Miller-Rabin nunca erra ao identificar um número composto.

2.2 RSA e Assinaturas Digitais

O algoritmo RSA é um sistema de criptografia de chave pública que utiliza a dificuldade de fatorar números primos grandes para garantir a segurança. Ele emprega aritmética modular para gerar mensagens criptografadas e permite verificar a autoria da criptografia, sendo amplamente utilizado para assinaturas digitais. O RSA funciona através da geração de duas chaves a partir de dois números primos: a chave pública (n, e) e a chave privada (n, d).

A criptografia de uma mensagem (m) é feita com a chave pública, resultando em $c = m^e \ mod \ n$. A descriptografia da mensagem cifrada (c) é realizada com a chave privada, através de $m = c^d \ mod \ n$. Para assinar um arquivo, um resumo dele (hash) é criado e criptografado usando a chave privada.

2.3 OAEP (Optimal Asymmetric Encryption Padding)

O OAEP (Optimal Asymmetric Encryption Padding) é um esquema de preenchimento que aumenta a segurança do RSA, protegendo contra ataques que exploram a estrutura do texto original e assegurando que entradas similares gerem saídas distintas. Seu funcionamento envolve três passos principais:

1. **Preparação da Mensagem:** A mensagem original recebe um preenchimento e é combinada com uma máscara criada por uma função de geração de máscara (MGF) usando uma "semente" (seed) aleatória. Isso forma uma mensagem embaralhada (ME).
2. **Aplicação de Máscaras:** A semente é então mascarada utilizando a MGF e a própria mensagem embaralhada (ME) como semente. Isso garante que pequenas modificações na mensagem ou na semente produzam resultados completamente diferentes.
3. **Codificação Final:** O resultado é concatenado e formatado para ser cifrado pelo algoritmo RSA.

A principal vantagem do OAEP é transformar o RSA de um algoritmo determinístico em um probabilístico, dificultando ataques por análise de padrões. Além disso, ele incorpora verificações internas que detectam alterações ou corrupções nos dados antes da decifração.

3. Estrutura da Implementação

3.1 Geração de Chaves

- Escolha de dois primos grandes (p e q), com pelo menos 1024 bits.

- Teste de primalidade com Miller-Rabin.

- Cálculo de n = p * q, z = (p-1)(q-1).

- Escolha do expoente público **e** e cálculo do inverso modular **d**.

3.2 Assinatura

- Geração do hash SHA-3 da mensagem original.

- Cifração do hash com a chave privada usando RSA (**sign**).

- Concatenação **Hash_algorythm||n||e||sign**.

- Codificação do resultado em BASE64 para armazenamento e transmissão.

3.3 Verificação

- Parsing da mensagem assinada e extração da assinatura BASE64.

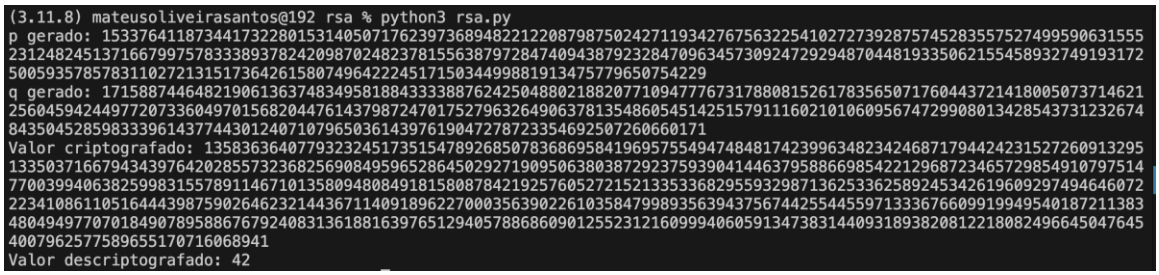
- Decodificação e decifração com a chave pública.

- Recomputação do hash da mensagem original e comparação com o hash decifrado.

4. Testes e Resultados

4.1 RSA e OAEP

A implementação do RSA foi feita, em primeiro lugar, com números inteiros. Como exemplo, utilizando o número 42, a forma mais básica do código RSA ficou:



Depois disso, foi implementado também o OAEP com funções para codificar e decodificar:

Finalmente, foi implementada uma versão do RSA utilizando o OAEP para codificar a mensagem antes de criptografá-la, gerando o seguinte resultado.

4.2 Assinatura e verificação

Foi usado um arquivo com a seguinte mensagem:

"*Este e um documento de teste para assinatura.*"

- Verificação de sucesso da assinatura.

Nessas condições a assinatura gerou a seguinte saída:

```
Gerando p (primo de 1024 bits)...
p gerado: 17314210943608008422726703586109243143081722137723685410554083426401009849255974551839671972272568017861909015398827815080155063428281102815614122183962943598879106128748030883542118951582845375492784690561472943510149883625815559948694416802254099098806830354114869689797749460237087544744454850461413767278 1
Gerando q (primo de 1024 bits)...
q gerado: 16268338389693566448021991428756373021301020426141394214394609714055595425199389237405266519037045926640243105077041763758550650024382909052202357657927379592690351531668449442110694337313769801488629165880985269329902820106744724748954622847428669012571228600572960673195074662640629621497311311491183595081

Hash do documento (32 bytes): 60a6c911dd7a301bb76295f3c29e9b7e2e9824f381a54fb9edf4157e6554b626
```

Enquanto na verificação obtivemos:

```
--- Iniciando Verificação ---
DEBUG: Decrypted hash (bytes): 60a6c911dd7a301bb76295f3c29e9b7e2e9824f381a54fb9edf4157e6554b626
DEBUG: Actual file hash (bytes): 60a6c911dd7a301bb76295f3c29e9b7e2e9824f381a54fb9edf4157e6554b626
Assinatura válida para o arquivo 'documento.txt'!
Verificação concluída com sucesso: Assinatura é autêntica.
```

Já para o caso após modificar o arquivo original, obteve-se a seguinte saída:

```
--- Tentando verificar após adulteração do arquivo original ---
DEBUG: Decrypted hash (bytes): 60a6c911dd7a301bb76295f3c29e9b7e2e9824f381a54fb9edf4157e6554b626
DEBUG: Actual file hash (bytes): 4008a5c5373c0380160b651c993adc87a5ec27d09543da7fa4c250eaa1533aad
Assinatura inválida para o arquivo 'documento.txt'. Os hashes não correspondem.
Adulteração detectada com sucesso (como esperado).
```

Com essas saídas foi observado que os algoritmos para a assinatura e para a verificação funcionam corretamente.

5. Considerações Finais

A implementação atendeu todos os requisitos propostos: geração de chaves RSA com teste de primalidade, assinatura de mensagens usando SHA-3 e OAEP, e verificação baseada em parsing e comparação de hashes.

Dificuldades incluíram o tratamento de arquivos grandes e otimização da geração de primos. Como melhoria futura, sugere-se implementar uma interface gráfica ou adaptar para múltiplos formatos de assinatura.

6. Referências

- KUROSE, Jim; ROSS, Keith. *Computer networking: a top-down approach*. 8. ed. Boston: Pearson, 2020.

-Wikipedia: https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding