Segurança Computacional - Trabalho 03

Luis Fernando Lamellas

December 3, 2023

1 Introdução

Denominado a partir os sobrenomes de seus criadores - Rivest, Shamir e Adleman - o modelo de criptografia RSA se destaca como um algoritmo de criptografia assimétrica. A criptografia assimétrica difere da simétrica ao empregar um par de chaves único. A chave pública, divulgada abertamente, desempenha o papel de cifrar os dados, enquanto a chave privada, mantida em segredo, realiza a decifragem. Esta abordagem oferece uma camada adicional de segurança, fortalecendo a proteção de informações sensíveis durante transmissões online e armazenamento.

Neste projeto, será implementado um algoritmo de criptografia RSA, abordando a geração de suas chaves públicas e privadas, bem como assinatura e verificação de assinaturas. Ademais, foi utilizado o esquema OAEP para realizar o padding da mensagem.

2 Fundamentação Teórica

A criptografia RSA opera com base em um sistema de chaves assimétricas, empregando um par de chaves única para cifragem e decifragem. O processo inicia-se com a geração desse par de chaves, composto por uma chave pública, que pode ser compartilhada abertamente, e uma chave privada, mantida em segredo pelo destinatário.

Para cifrar uma mensagem utilizando RSA, o remetente utiliza a chave pública do destinatário. A mensagem é convertida em um valor numérico e elevada à potência da chave pública, seguida por um módulo da chave pública. O resultado é a mensagem cifrada, que pode ser transmitida com segurança.

A decifragem, por sua vez, é realizada pelo destinatário, que utiliza sua chave privada. A mensagem cifrada é elevada à potência da chave privada, também seguida por um módulo da chave pública. O resultado é a mensagem original, recuperando assim a informação original.

A segurança do RSA baseia-se na dificuldade de fatorar números grandes em seus primos constituintes, um problema considerado computacionalmente intratável para números suficientemente grandes. Isso significa que, mesmo conhecendo a chave pública, decifrar a mensagem sem a chave privada é extremamente desafiador.

Além do seu funcionamento básico, o RSA frequentemente incorpora esquemas de preenchimento (padding) para aprimorar a segurança e a eficiência. Um desses métodos é o Optimal Asymmetric Encryption Padding (OAEP), que protege contra certos ataques e assegura maior robustez ao algoritmo.

No contexto do RSA com OAEP, o processo de cifragem envolve a adição de uma sequência de bits aleatórios à mensagem antes da aplicação do algoritmo básico. Isso garante que mensagens idênticas não produzam resultados cifrados iguais, mitigando vulnerabilidades.

Durante a decifragem, o OAEP desempenha um papel crucial na remoção do preenchimento adicionado. Ele restaura a mensagem original, assegurando que o destinatário obtenha a informação sem ambiguidades. A inclusão do OAEP reforça a segurança do RSA, tornando-o mais resiliente a possíveis ataques criptográficos.

Resumidamente, implementaremos as seguintes funções:

1. Geração de Chaves
2. Cifragem e Decifragem
3. Assinatura e Verificação

2.1 Geração de Chaves

As chaves públicas e privadas são geradas conforme o algoritmo

Algorithm 1: Algoritmo para geração de chave pública e privada

1 Escolha dois números primos grandes p e q

2 Compute $n = p * q$

3 Compute $\phi(n)$, onde $\phi(n)$ representa a multiplicação de todos os divisores primos de n

4 Escolha um inteiro e que satisfaça $2 < e < \phi(n)$ e $\gcd(e, \phi(n)) = 1$

5 Determine d, onde $d \equiv e^{-1} \mod \phi(n)$

É importante ressaltar que os números primos escolhidos tiveram sua primalidade verificada através da utilização do algoritmo de **Miller-Rabin**. Desse modo, encontra-se todos os fatores necessários para cifração e decifração, são estes:

e: chave pública

d: chave privada

n: módulo

2.2 Cifragem de Decifragem

O primeiro passo, tanto no processo de cifragem quanto no de decifragem, é converter a mensagem m ou cifra c para um número inteiro.

2.2.1 Cifragem

No caso da cifragem, a cifra é gerada de acordo com a expressão

$$c \equiv m^e \pmod{n}$$

2.2.2 Decifragem

No caso da decifragem, a cifra é gerada de acordo com a expressão

$$c^d \equiv (m^e)^d \equiv m \pmod{n}$$

2.3 Assinatura e Verificação

A criação de uma assinatura digital requer uma função de hash que é aplicada para gerar um identificador de tamanho fixo para o conteúdo a ser assinado. A assinatura digital é então gerada usando a chave privada do usuário com base nesse identificador. O resultado é um pacote assinado que inclui o documento, a assinatura e o certificado digital do signatário.

Para verificar a validade do documento, o destinatário utiliza a chave pública contida no certificado digital para descriptografar a assinatura, obtendo assim o hash original. Em seguida, calcula novamente o hash do documento recebido e verifica se ambos coincidem, comprovando assim a validade da assinatura e a integridade do documento. É essencial também verificar a confiabilidade e validade do certificado utilizado na assinatura.

Para tal finalidade, foi utilizado o **Optimal Asymmetric Encryption Padding** (OAEP).

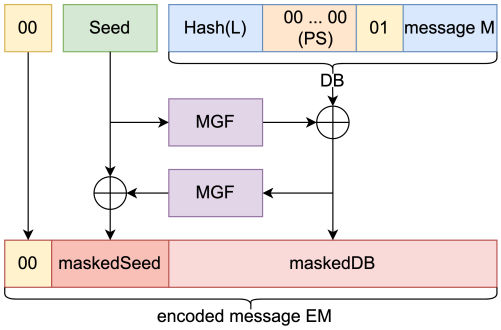

Figure 1: Optimal Asymmetric Encryption Padding

- MGF é a função geradora de máscara, geralmente MGF1,

- Hash é a função de hash escolhida **SHA-3**,
- hLen é o comprimento da saída da função de hash em bytes,
- k é o comprimento do módulo RSA n em bytes,
- M é a mensagem a ser preenchida (no máximo $k - 2 \cdot \text{hLen} - 2$ bytes),
- L é um rótulo opcional a ser associado à mensagem (o rótulo é a string vazia por padrão e pode ser usada para autenticar dados sem exigir criptografia),
- PS é uma sequência de bytes de $k - \text{mLen} - 2 \cdot \text{hLen} - 2$ bytes nulos.
- $\oplus$ é uma operação XOR.

2.3.1 Codificação OAEP

1. Hash da etiqueta L usando a função de hash escolhida:

$$\text{lHash} = \text{Hash}(L)$$

2. Gere uma sequência de preenchimento PS consistindo em $k - \text{mLen} - 2 \cdot \text{hLen} - 2$ bytes com o valor 0x00.

3. Concatene lHash, PS, o byte único 0x01 e a mensagem M para formar um bloco de dados DB:

$$\text{DB} = \text{lHash}||\text{PS}||\text{0x01}||M$$

   Este bloco de dados tem comprimento $k - \text{hLen} - 1$ bytes.

4. Gere uma semente aleatória com comprimento hLen.

5. Use a função geradora de máscara para gerar uma máscara do comprimento apropriado para o bloco de dados:

$$\text{dbMask} = \text{MGF}(\text{seed}, k - \text{hLen} - 1)$$

6. Mascare o bloco de dados com a máscara gerada:

$$\text{maskedDB} = \text{DB} \oplus \text{dbMask}$$

7. Use a função geradora de máscara para gerar uma máscara de comprimento hLen para a semente:

$$\text{seedMask} = \text{MGF}(\text{maskedDB}, \text{hLen})$$

8. Mascare a semente com a máscara gerada:

$$\text{maskedSeed} = \text{seed} \oplus \text{seedMask}$$

9. A mensagem codificada (preenchida) é o byte 0x00 concatenado com amaskedSeed e maskedDB:

$$\text{EM} = \text{0x00}||\text{maskedSeed}||\text{maskedDB}$$

2.3.2 Decodificação OAEP

1. Hash da etiqueta L usando a função de hash escolhida:

$$\text{lHash} = \text{Hash}(L)$$

2. Para reverter o passo 9, divida a mensagem codificada EM nos bytes 0x00, a maskedSeed (com comprimento hLen) e a maskedDB:

$$\text{EM} = \text{0x00}||\text{maskedSeed}||\text{maskedDB}$$

3. Gere a seedMask que foi usada para mascarar a semente:

$$\text{seedMask} = \text{MGF}(\text{maskedDB}, \text{hLen})$$

4. Para reverter o passo 8, recupere a semente com a seedMask:

$$\text{seed} = \text{maskedSeed} \oplus \text{seedMask}$$

5. Gere a dbMask que foi usada para mascarar o bloco de dados:

$$\text{dbMask} = \text{MGF}(\text{seed}, k - \text{hLen} - 1)$$

6. Para reverter o passo 6, recupere o bloco de dados DB:

$$\text{DB} = \text{maskedDB} \oplus \text{dbMask}$$

7. Para reverter o passo 3, divida o bloco de dados em suas partes:

$$\text{DB} = \text{lHash'}||\text{PS}||\text{0x01}||\text{M}$$

3 Implementação

Para implementação da criptografia RSA-OAEP foi utilizada a linguagem de progrmação **python**. O código encontra-se disponível em `https://github.com/lflamellas/RSA-Encryption`. No arquivo `rsa.py` foram implementadas todas as funções necessárias para execução da criptografia de um arquivo texto.

Para executar os testes use o comando `python rsa.py`

4 Conclusão

O trabalho enriqueceu o conhecimento em segurança computacional, destacando a relevância da criptografia RSA. Explorar os princípios da criptografia assimétrica, incluindo a geração de chaves e técnicas como a máscara geradora, aprofundou a compreensão sobre a proteção da comunicação digital.

O processo de cifragem e decifragem no contexto do RSA revelou-se crucial para preservar a privacidade e a integridade dos dados. Essa experiência não apenas solidificou meu entendimento teórico, mas também ressaltou a importância prática desses conceitos na cibersegurança.

Ademais, consolidou habilidades de programação e resolução de problemas, contribuindo para o aprendizado na área.

References

[1] Nist special publication 800-56br2. https://doi.org/10.6028/NIST.SP.800-56Br2, 2023. [Acessado 3-December-2023].

[2] Wikipedia contributors. Optimal asymmetric encryption padding — Wikipedia, the free encyclopedia. https://w.wiki/8NG4, 2023. [Acessado 3-December-2023].

[3] Wikipedia contributors. Rsa (cryptosystem) — Wikipedia, the free encyclopedia. https://w.wiki/8NF$, 2023. [Acessado 3-December-2023].