Segurança Computacional - Trabalho 2

Alunos

- Artur Filgueiras Scheiba Zorron - 180013696
- Brenno Pereira Cordeiro - 190127465

Descrição

O segundo trabalho da disciplina de Segurança Computacional consiste em implementar:

```
1. Geração de chaves com teste de primalidade (Miller-Rabin)
2. Cifração e decifração RSA
3. OAEP
4. Formatação/parsing
5. AES modo CTR
```

Algoritmos

Miller-Rabin

O algoritmo de Miller-Rabin é um algoritmo de teste de primalidade. Ele é utilizado para testar se um número é primo ou não. Este algoritmo é chamado k vezes, onde quanto maior o k maior é a probabilidade de sucesso.

```
d = número primo o qual: d * 2^r = n - 1, tal que r >= 1
a = número aleatório entre 2 e n-1

x = a^d (mod n)

se x = 1 ou x = n-1, então o número é primo

se não:
    x = x^x (mod n)

    se x = 1, então o número não é primo
    se x = n-1, então o número é primo
```

RSA (Rivest, Shamir e Adleman)

RSA é um algoritmo de cifração assimétrica. A chave de cifração RSA é gerada aleatoriamente. A cifração RSA é implementada utilizando o algoritmo de exponenciação modular.

```
p = número primo aleatório
q = número primo aleatório
phi = função de Euler
e = número primo aleatório entre 1 e phi
m = mensagem

n = p * q
phi = (p-1) * (q-1)
d = e^-1 (mod phi)
c = m^e (mod n)
m' = c^d (mod n)
```

OAEP (Optimal Asymmetric Encryption Padding)

OAEP é um algoritmo de preenchimento comumente utilizado junto à cifração RSA. Este algoritmo é utilizado junto ao RSA pois com ele a propriedade deterministica é evitada.

```
cifrando:
    n = número de bits no módulo RSA
    k0 e k1 = inteiros fixados pelo protocolo
    r = string de k0 bits aleatória
    m = mensagem preenchida com k1 zeros para ter n - k0 bits
    G e H = funções de geração de máscaras baseadas em funções de hash

    G -> expande os k0 bits de r para n - k0 bits
    X = m XOR G(r)
    H -> reduz os n - k0 bits de X para k0 bits
    Y = H(X) XOR r

    saída -> X || Y

decifrando:
    recupera a string aleatória r = Y XOR H(X)
    recupera a mensagem m = X XOR G(r)
```

AES (Advanced Encryption Standard)

AES é um algoritmo de cifração simétrica utilizado para cifrar e decifrar mensagens. Este algoritmo é uma cifra de blocos com um comprimento de 128 bits. Utilizamos uma chave de 128 bits.

```
deriva o conjunto de chaves das rodadas da chave de cifra

inicializa o array de estado com os dados do bloco (texto simples)

adiciona a chave da rodada inicial à matriz de estado inicial
```

```
execute nove rodadas de manipulação do estado

cada rodada:
    substitua os bytes
    arraste os bytes
    misture as colunas
    execute XOR com a chave da rodada

realize a décima e última rodada de manipulação do estado
```

CTR (Counter Mode)

CTR é uma implementação utilizando operadores XOR bit a bit para cifrar e decifrar mensagens. Este algoritmo é uma cifra de blocos com um comprimento de 128 bits. Utilizamos uma chave de 128 bits.

```
cifrando:
    gera bits aleatórios com a chave de cifra e o vetor inicial
    execute XOR com a string recebida
    cifra o texto

decifrando:
    utiliza os mesmos bits aleatórios gerados na cifração
    execute XOR com a chave de cifra
    execute XOR com o vetor inicial
    decifra o texto
```

Implementação

`./main.py genkeys`

Gera as chaves pública e privada.

`./main.py cipher -f <file>`

Cifra arquivo utilizando a chave pública gerada anteriormente.

`./main.py decipher -f <ciphered_file> -o <target_file>`

Decifra arquivo <ciphered_file> utilizando a chave privada gerada anteriormente.

Referências

- https://datatracker.ietf.org/doc/html/rfc8017#section-7.1.1
- https://datatracker.ietf.org/doc/html/rfc8017#section-7.1.2
- http://www.moserware.com/assets/stick-figure-guide-to-advanced/A%20Stick%20Figure%20Guide%20to%20the%20Advanced%20Encryption%20Standard%20%28AES%29.pdf

- https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf
- https://en.wikipedia.org/wiki/Padding_(cryptography)#PKCS#5_and_PKCS#7
- https://www.geeksforgeeks.org/primality-test-set-3-miller-rabin/