Trab3Seg

Davi Mansur Costa

December 2023

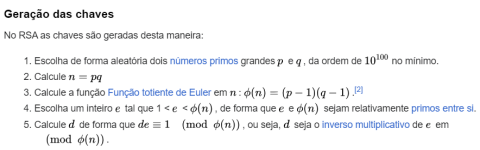
1 Introdução

Esse trabalho trata do RSA, sistema de criptografia assimétrica.Ele é um dos primeiros sistemas de criptografia de chave pública e foi também o primeiro algoritmo a possibilitar a assinatura digital, uma das grandes inovações em criptografia de chave pública.

2 Implementação

2.1 Geração de Chaves

Os primeiros passos para essa etapa são os da imagem abaixo:



Geração das chaves

No RSA as chaves são geradas desta maneira:

1. Escolha de forma aleatória dois números primos grandes p e q, da ordem de 10^{100} no mínimo.
2. Calcule $n = pq$
3. Calcule a função Função totiente de Euler em n: $\phi(n) = (p-1)(q-1)$.[2]
4. Escolha um inteiro e tal que $1 < e < \phi(n)$, de forma que e e $\phi(n)$ sejam relativamente primos entre si.
5. Calcule d de forma que $de \equiv 1 \pmod{\phi(n)}$, ou seja, d seja o inverso multiplicativo de e em $\pmod{\phi(n)}$.

Figure 1: Geracao de Chaves

Para a geração de chaves foram feitas algumas funções

1. eh_primo :
   Essa função foi feita para ver se um número é primo.

2. gera_primo :
   Essa função foi feita para gerar um número aleatório primo, foi utilizado random para gerar um número e enquanto a função eh_primo não falasse que ele era primo, gerava denovo até achar um primo.Essa função foi utilizada para achar p e q.

3. mod_inverso:
   Essa função foi feita para encontrar o inverso multiplicativo de e em (modϕ(n))). Foi usada para achar a chave privada

A chave pública foi feita usando o MDC de um número e, com as propriedades da imagem acima e o phi da função totiente de Euler. A chave privada foi calculada pela função mod_inverso também de e e do phi.

2.2 Cifração/decifração

Para a cifração foram convertidos os caracteres usando ascii e então foi calculado (m^e)mod n = c para cada caracter convertido. E para a decifração, o contrário, retorna os números convertidos para ascii e depois de ascii para caracter da mensagem decifrada.

```
def criptografar(mensagem,e,n):
    convertida = [ord(caracter) for caracter in mensagem]
    #(m^e) mod n = c
    criptograma = [pow(caracter,e,n) for caracter in convertida]
    return criptograma

def decriptografar(criptograma,d,n):
    convertida = [pow(caracter,d,n) for caracter in criptograma]
    mensagem = "".join(chr(caracter) for caracter in convertida)
    return mensagem
```

2.2.1 Exemplo teste1

Mensagem original: Attack at dawn
Mensagem criptografada: [77433, 1885475, 1885475, 1161711, 712486, 2530098, 224369, 1161711, 1885475, 224369, 936309, 1161711, 294964, 5080499]
Mensagem decriptografada: Attack at dawn

2.2.2 Exemplo teste2

Mensagem original: Attack at dawn
Mensagem criptografada: [538131, 225263, 225263, 590974, 778147, 1110422, 1363151, 590974, 225263, 1363151, 211695, 590974, 1355204, 633856]
Mensagem decriptografada: Attack at dawn

3 Conclusão

Embora não tenha conseguido implementar o OAEP(Optimal asymmetric encryption padding), usado para fortalecer a segurança do RSA, a cifração/ decifração foi feita com sucesso e o trabalho foi de muito aprendizado, não somente da parte feita com sucesso mas também das partes que falharam(OAEP e assinatura digital).