INTRODUÇÃO À CRIPTOGRAFIA RSA E CÓDIGO EM LINGUAGEM C

O algoritmo de criptografia Rivest-Shamir-Adleman (RSA) é um algoritmo de criptografia assimétrico amplamente utilizado em muitos produtos e serviços. A criptografia assimétrica usa um par de chaves matematicamente vinculado para criptografar e descriptografar dados. Duas chaves, uma pública e outra privada são criadas, sendo a chave pública acessível a qualquer pessoa e a chave privada sendo um segredo conhecido apenas pelo criador do par de chaves. Com o RSA, a chave privada ou pública pode criptografar os dados, enquanto a outra chave os descriptografa. Essa é uma das razões pelas quais o RSA é o algoritmo de criptografia assimétrica mais usado.

Algoritmo -

1 - Geração das chaves:

Selecione dois números primos grandes, a e b.Os números primos precisam ser grandes para que seja difícil para alguém descobrir.

Calcule n = a*b

Calcular a função totiente: $\phi(n)=(a-1)(b-1)$.

Selecione um inteiro e, tal que e seja coprimo com $\phi(n)$ e $1 < e<\phi(n)$. O par de números (n,e) compõe a chave pública.

Nota: Dois inteiros são coprimos se o único inteiro positivo que os divide for 1.

(Em linguagem C - Portanto, é feito um teste com uma função mdc para comprovar que são coprimos)

Calcule d tal que $e*d = 1 \bmod \phi(n)$.

(Em linguagem C - É usada uma função para cálculo do inverso multiplicativo modular)

d pode ser encontrado usando o algoritmo euclidiano estendido. O par (n,d) compõe a chave privada.

2 – Criptografia:

Dado um texto simples, representado como um número, o texto cifrado é calculado como:

C=M^(e) mod (n) --- Onde "M" representa o número relacionado ao caractere original.

3 – Descriptografia:

Usando a chave privada (n,d), o texto simples pode ser encontrado usando:

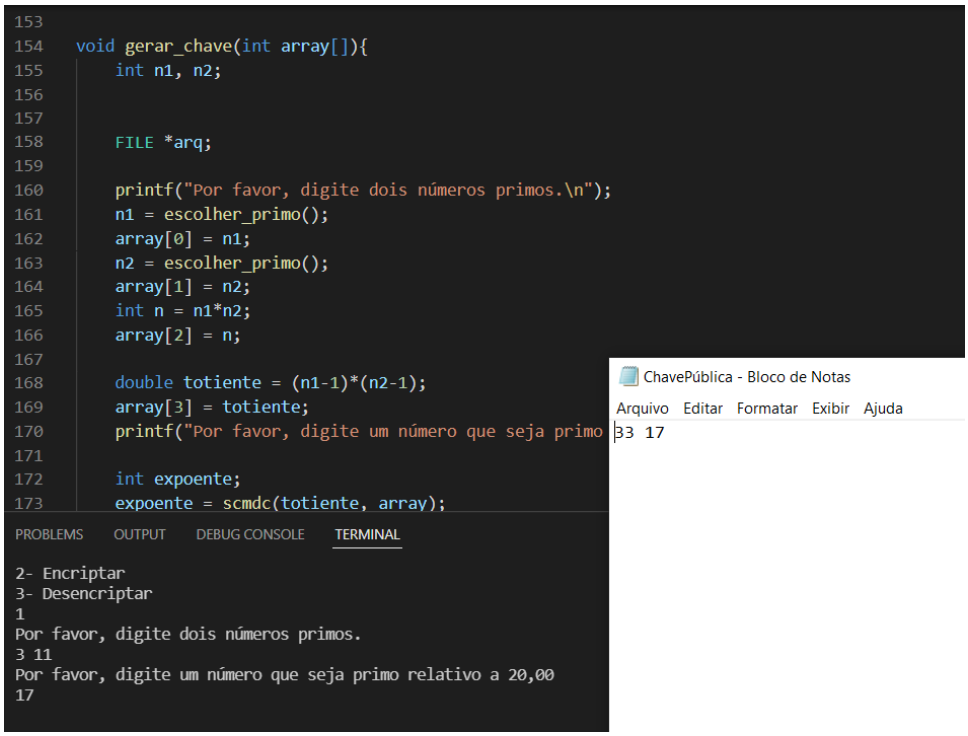
M=C^(d) mod (n)

Por que o algoritmo funciona?

Por definição temos que $(b^e)^d \equiv b^{ed} \bmod (n)$. Mas d é o inverso de e mod $\phi(n)$, então existe um inteiro k tal que $ed=1+k*\phi(n)$ portanto substituiremos ed por $1+k* \phi(n)$ em b^{ed} e teremos $[(b^\phi(n))^k]*b \equiv b \bmod(n)$. Sendo $n=p*q$ (o produto de dois primos distintos), $\phi(n)=(p-1)*(q-1)$, substituindo na equação anterior e aplicando o teorema de Fermat temos $b^{ed} \equiv b \bmod (p*q)$, como prova do funcionamento já que b^(ed) = b para qualquer b.

Para a aplicação do algoritmo em linguagem C é necessária a presença de algumas funções com o objetivo de determinar mdc, calcular inverso multiplicativo modular e exponenciação modular, além de funções que analisem e comprovem a primalidade dos números. É possível fazer a conversão de um caractere em número a partir da aplicação de loops de repetição que caminham por todos os caracteres de uma frase e, a partir do armazenamento destes caracteres em um array, a conversão é dada de forma sequencial. Para a descriptografia, o processo inverso é feito e o número é convertido em caractere ASCII até que seja encontrado o fim do arquivo.

CÓDIGO TESTE (FORAM UTILIZADOS 33 E 17 COMO PAR (N,E), ONDE OS PRIMOS SÃO OS NÚMEROS 3 E 11):



```c
void gerar_chave(int array[]){
    int n1, n2;


    FILE *arq;

    printf("Por favor, digite dois números primos.\n");
    n1 = escolher_primo();
    array[0] = n1;
    n2 = escolher_primo();
    array[1] = n2;
    int n = n1*n2;
    array[2] = n;

    double totiente = (n1-1)*(n2-1);
    array[3] = totiente;
    printf("Por favor, digite um número que seja primo
    int expoente;
    expoente = scmdc(totiente, array);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Selecione a opção desejada:
1- Gerar chave pública.
2- Encriptar
3- Desencriptar
2


Digite a chave pública previamente recebida.
33 17
Digite a mensagem que deseja encriptar: matematica discreta

Mensagem encriptada com sucesso!
```

código - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

20 29 21 30 20 29 21 10 16 29 19 14 10 26 16 13 30 21 29

```c
    int n1, n2;


    FILE *arq;

    printf("Por favor, digite dois números primos.\n");
    n1 = escolher_primo();
    array[0] = n1;
    n2 = escolher_primo();
    array[1] = n2;
    int n = n1*n2;
    array[2] = n;

    double totiente = (n1-1)*(n2-1);
    array[3] = totiente;
    printf("Por favor, digite um número que seja primo relativo a %.2lf\n", totiente);

    int expoente;
    expoente = scmdc(totiente, array);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Selecione a opção desejada:
1- Gerar chave pública.
2- Encriptar
3- Desencriptar
3
Digite o par de números primos escolhidos ao gerar a chave pública
3 11
Digite o número coprimo escolhido: 17
Mensagem desencriptada: matematica discreta
```