Criptografia

22/01/2021

Autor: *Luiz Fernando Bittencourt*

A criptografia, que consiste na troca de informações sem que haja o comprometimento do sigilo, talvez seja tão antiga quanto a própria escrita.

Para criptografarmos algo, fazemos o uso de chaves, que permitem a codificação e a decodificação da informação.

Um exemplo de criptografia foi usado por Júlio César, que foi um grande Imperador de Roma, participou do primeiro triunvirato (junto com Pompeu e Crasso, em 60 a.C.) e foi assassinado no Senado em 44 a.C.. A chave utilizada por Júlio César era muito simples: desloca-se o alfabeto 3 letras e troca-as entre si.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Por exemplo, se ele quisesse enviar a mensagem "Veni, vidi, vici", a mensagem criptografada seria "YHQL YLGL, YLFL".

Este tipo de criptografia é classificada como **criptografia de chave simétrica**, ou seja, que utiliza a mesma chave para codificar e decodificar a informação.

Outra história famosa, e real, sobre utilização da criptografia é a história de Billy, que durante a guerra pela independência dos Estados Unidos, entregou ao dono de um hotel onde se hospedara um envelope. Billy disse a ele, que era um homem de conhecida confiança na época, que se ele, Billy, não retornasse em 10 anos, o dono do hotel receberia um telegrama com instruções de o que fazer com aquele envelope. Além disso, Billy também disse que havia 3 folhas dentro do envelope, cada qual contendo uma parte da mensagem: a primeira dizia como, a segunda dizia o que e a terceira dizia a quem. Passaram-se 10 anos e nada de o dono do hotel receber o telegrama. Este homem, como era um homem de honestidade inquestionável, resolveu esperar mais tempo. Passaram-se mais 5 anos, então ele resolveu abrir o envelope. Dentro dele, foram encontradas 3 folhas, como Billy havia dito. Nessas folhas, havia números. Muitos números, separados por vírgulas. Era uma mensagem criptografada. Como decifrá-las, se

as instruções nunca foram conhecidas? Muitas tentativas foram feitas, até que se conseguiu decifrar a segunda folha. Nela estava escrito que as folhas diziam respeito a valores de U\$20.000.000,00. As outras duas folhas nunca foram decifradas.

O método que Billy utilizou para criptografar esta segunda folha foi o seguinte:

> a partir de um texto qualquer, numeram-se as palavras;

1- 2 ---3 -4--- 5----- 6----- 7----- 8---- 9

> cada número corresponde à primeira letra da palavra correspondente;

> cada letra pode ter mais de um número, onde pode ser usado qualquer um deles.

Neste caso, se quisermos enviar a mensagem "panda", devemos então enviar os números 9,1,7,3,1.

Billy utilizou, na ocasião, o texto da Declaração de Independência dos Estados Unidos. Quem conseguir decifrar as outras duas folhas pode ser contemplado com um tesouro de U\$20.000.000,00.

Criptografia RSA

A **criptografia RSA** foi pensada por Rivest, Shamir e Adleman, sendo os dois primeiros da área da computação e o último, da área da matemática. Ela é uma **criptografia de chave assimétrica**, ou seja, utiliza uma **chave pública** e uma **chave privada**. A chave pública é utilizada para codificar e a chave privada para decodificar. O nome *chave pública* vem do fato de que você pode distribuir esta chave a qualquer pessoa sem comprometer a segurança dos dados ou da chave privada. Toda mensagem criptografada via RSA é decodificável sem a chave. Porém, a decodificação é inviável devido ao tempo que esta operação demoraria.

Por exemplo, se você usar SSH (Secure SHell), você pode mandar sua chave pública (por exemplo, o conteúdo do arquivo `~/.ssh/identity.pub`) para um administrador de algum sistema que você queira acessar para colocar em seu arquivo `~/.ssh/authorized_keys`. Para qualquer um que realmente queira acesso, é necessário que este alguém tenha a chave privada correspondente (por exemplo, o conteúdo decodificado de `~/.ssh/identity`) para se identificar. Para proteger a chave privada, você deve entrar com uma *passphrase* para codificar a chave quando esta é guardada no sistema de arquivos. Isto previne que

VOS.

```
$ ssh-keygen
Initializing random number generator...
Generating p: .++ (distance 6)
Generating q: .....++ (distance 110)
Computing the keys...
Testing the keys..
Key generation complete.
Enter file in which to save the key ($HOME/.ssh/identity): [RETURN]
Enter passphrase (empty for no passphrase): uma frase2boa
Enter same passphrase again: uma frase2boa
Your identification has been saved in /u/kim/.ssh/identity.
Your public key is: 1024 37 [vários números]
login@localhost Your public key has been saved in $HOME/.ssh/identity.pub
```

Quadro 1 - Exemplo de geração de chave RSA em uma shell linux

Toda comunicação é feita usando outros tipos de criptografia, como o IDEA ou outros (DES, RC4-128, TSS, Blowfish). A troca das chaves é feita usando RSA, e os dados usados para a troca da chave são destruídos a cada hora (as chaves não são salvas em lugar algum). Cada host tem uma chave RSA que é usada para autenticar o host quando a autenticação RSA é usada.

- IP spoofing, onde um host remoto envia pacotes fingindo ser outro host de confiança. SSH também protege contra um spoofer na rede local, que finge ser seu roteador para o lado de fora.
- IP source routing, onde um host finge que um pacote IP vem de outro host de confiança.
- DNS spoofing, onde o agressor forja os registros de servidores de nome.
- Interceptação de passwords em texto ou outros dados por hosts intermediários.
- Manipulação de dados por pessoas que estão controlando hosts intermediários.
- Ataques baseados em ouvir os dados da autenticação X e conexão com spoof ao servidor X11 (Linux/Servidor X).

Quadro 2: Tipos de ataques que o SSH pode prevenir

Mas como funciona a criptografia RSA? Se você olhar na geração da chave SSH, verá que são gerados 2 números (**p** e **q**). Estes números são a base da criptografia RSA e devem ser dois primos grandes e com uma grande diferença numérica entre si.

A verificação para saber se um número **n** é primo ou não pode ser feita tirando-

se a raiz quadrada de n e dividindo-o por todos os primos anteriores a este resultado.

Obs.: Número de primos menores que $x \sim x/\ln(x)$

Exemplo 1: Verificar se 101 é primo ou não.

Solução: basta dividir 101 pelos primos 2, 3, 5 e 7. Se alguma dessas divisões for exata (inteira), então 101 não é primo. Se todas as divisões não são exatas, 101 é primo.

Exemplo 2: Verificar se $r =$

123456789012345678901234567890123456789012347 é primo ou não.

Solução: a partir de um certo ponto, é muito trabalhoso estabelecer a seqüência dos primos menores que raiz quadrada de r . Como r tem 45 algarismos, resulta que $10^{44} < r < 10^{45}$

Então $10^{22} < \text{raiz quadrada de } r < 10^{23}$.

Em vez de dividir r pelos números primos menores do que 10^{23} podemos dividir r pelos números ímpares 1, 3, 5, 7, 9, 11 ... menores do que 10^{23} .

No caso de r ser primo, o número de divisões que é necessário efetuar está entre $(10^{22})/2$ e $(10^{23})/2$.

Suponhamos que um computador efetue 10^{10} divisões por segundo.

1 ano tem 31.536.000 segundos. Então esse computador efetua $31.536.000 \cdot 10^{10}$ divisões em um ano, o que é menor que 10^{18} .

Se o computador efetuasse as 10^{18} divisões em um ano, para efetuar $(10^{22})/2$ divisões ele demoraria 5000 anos.

Se aumentarmos este número de 45 para 49 algarismos, conseqüentemente o tempo que esse computador levaria para determinar se tal número é primo ou não seria de 500.000.000 anos.

Tempos como estes são classificados como tempos **não polinomiais**.

Em Z (inteiros): Definição: x

$\equiv$

$y \pmod{n}$ significa que $x-y$ é múltiplo de n , ou, x é **côngruo** com $y \pmod{n}$.

Notação: x^y significa x elevado a y.

$p*q$ significa q somado p vezes.

Obs.:

1) 3^7

$\equiv$

$3(\text{mod } 77)$

2) $p*q = n = 192343993140277293096491917$ (27 algarismos, produto de 2 primos)

43^{731}

$\equiv$

$103730879700159299993828112(\text{mod } n)$

$103730879700159299993828112^{134482257019077958238644371}$

$\equiv$

$43(\text{mod } n)$

onde:

43 é a mensagem.

$103730879700159299993828112$ é a mensagem criptografada.

$134482257019077958238644371$ é a chave privada.

731 é a chave pública.

É claro que para a utilização prática, o número de algarismos de cada chave é bem maior. Quanto maior a chave, mais tempo alguém levaria para decodificar a mensagem.

Inversos Módulo N

Teorema:

Se $\text{mdc}(a,n) = 1$, então existe um único **b** entre 1 e $n-1$ tal que

$$\begin{aligned} a \cdot b \\ \equiv \\ 1 \pmod{n} \end{aligned}$$

diz-se então que **b é o inverso de a módulo n**, e se escreve

$$b = a^{-1} \pmod{n}$$

Exemplo: $\text{mdc}(3,11) = 1$. Então existe um único **b** entre 1 e 10 tal que $3 \cdot b$

$$\begin{aligned} \equiv \\ 1 \pmod{11}. \end{aligned}$$

Por tentativa, $b = 4$, ou seja,

$$\begin{aligned} 3 \cdot 4 \\ \equiv \\ 1 \pmod{11} \\ 4 = 3^{-1} \pmod{11}. \end{aligned}$$

Função ϕ de Euler

Definição: ($n \in \mathbb{Z}$, $n \geq 1$)

$\phi(n)$ = número de inteiros s tais que $1 \leq s \leq n-1$ e $\text{mds}(s,n) = 1$.

Exemplos: $\phi(5) = 4$, $\phi(25) = 20$, $\phi(8) = 4$.

Propriedades:

- 1) $\phi(p) = p-1$ e $\phi(p^n) = p^{n-1}$ se p é primo,
- 2) $\phi(a \cdot b) = \phi(a) \cdot \phi(b)$, a e b quaisquer ≥ 1 .
- 3) $\phi(a_1 \cdot a_2 \cdot a_3 \dots a_n) = \phi(a_1) \cdot \phi(a_2) \cdot \phi(a_3) \dots \phi(a_n)$, $a_1 \dots a_n$ quaisquer ≥ 1 .

Teorema (Fundamentação para RSA):

Sejam **n**, **p**, **q**, **a**, **b** inteiros positivos, tais que

p e **q** são primos,

$$n = p \cdot q,$$

$$ab$$

$$\equiv$$

$$1 \pmod{\varnothing(n)}, \text{ isto é, } b = a^{(-1)} \pmod{\varnothing(n)}.$$

Nessas condições,

Se m^b

$$\equiv$$

m (mod n).

Para um usuário, digamos José, receber mensagens criptografadas, ele deve obter números inteiros positivos **n**, **p**, **q**, **a**, **b**, tais que:

p e **q** são primos

$$n = p \cdot q$$

$$a \cdot b$$

$$\equiv$$

$$1 \pmod{\varnothing(n)}, \text{ a e b entre 1 e } \varnothing(n).$$

n e **a** são chaves públicas de José.

b é a chave secreta de José.

p e **q** são secretos (somente José os conhece).

Para Alice codificar uma mensagem **m** pertencente a Z (inteiros), ($2 \leq m \leq m-2$), ela calcula

$$y = m^a \pmod{n}.$$

y é a mensagem codificada, a qual é enviada para José.

Para José decodificar **y**, ele usa sua chave secreta **b** e calcula

$$y^b \pmod{n}$$

e encontra **m**, conforme teorema anterior (Fundamentação para RSA).

Para algum intruso conseguir decifrar esta mensagem, ele deveria conhecer a chave **b**.

Para isto:

- 1) Ele precisa obter $\varnothing(n)$. Tendo $\varnothing(n)$ e usando a chave pública **a**, ele calcula $b = a^{-1} \pmod{\varnothing(n)}$.
- 2) Para obter $\varnothing(n)$, ele precisa conhecer **p** e **q**, pois $\varnothing(n) = (p-1)*(q-1)$.
- 3) Para obter **p** e **q**, ele precisa fatorar **n**, pois $n = p*q$.

Fato: com a tecnologia e os algoritmos conhecidos nos dias de hoje, a fatoração de **n** não poderá ser feita em tempo polinomial se:

- 1) **p** e **q** tiverem cerca de 130 algarismos cada um no sistema decimal.
- 2) **p** e **q** não podem estar próximos um do outro, isto é, a diferença entre **p** e **q** deve ser relativamente grande.
- 3) $p-1$ deve ter um fator primo grande.
- 4) $q-1$ deve ter um fator primo grande.

Se José escolhe **p** e **q** sob as condições acima, então um intruso não consegue fatorar **n** em tempo polinomial; logo, não obtém $\varnothing(n)$; logo, não obtém **b**; logo, não consegue decodificar **y**.

Referências:

A Course in Number Theory and Cryptography, Neal Koblitz.

Criptography - Theory and Practice, Douglas R. Stinson.