Maple-MPSI

Mr. Mamouni

myismail@altern.org

Cryptographie RSA

Programmation en Maple

```
> restart:
```

On se donne d'abord des nombres premiers, plus qu'ils sont grand plus que c'est mieux, on comprendra dans la suite pourquoi.

```
> p:=13:q:=17:
```

On factorise le produit (p-1)(q-1) pour en déduire un nombre premier avec.

```
> ifactor((p-1)*(q-1));
```

$$(2)^6\ (3)$$

```
> e:=5:
```

On cherche les coéfficient de Bezout u,d tels que u(p-1)(q-1)+ed=1.

```
> igcdex((p-1)*(q-1),e,'u','d'):d;
```

$$77$$

On se donne le message à coder.

```
> Message:=162:
```

On code le message.

```
> Code:=Message^e mod p*q;
```

$$Code := 93$$

On décode le message codé.

```
> Decodage:=Code^d mod p*q;
```

$$Decodage := 162$$

Oh ça marche,... mais si on prend le message assez grand ?

```
> Message:=1452:
```

```
> Code:=Message^e mod p*q;
```

$$Code := 198$$

```
> Decodage:=Code^d mod p*q;
```

$$Decodage := 126$$

Oh ça ne marche plus, la raison c'est que le message dépasse pq, alors que le décodage non , parceque c'est son reste mod (pq), vérifions.

```
> Message mod p*q;
```

Pour y rémedier on découpera le message initial en blocs tous inférieurs à pq.

On définit une fonction appellée, hachage qui extrait le 1ér bloc.

```
> hachage:=proc(M,p,q) local n,a,b;
> for n from 1 to nops(M)  do
> a:=sum(M[k]*10^(k-1),k=1..n):b:=a-M[n]*10^(n-1):
> if a>=p*q then break fi:
> od:return([b,n-1]):
> end proc:
```

Puis on définit une fonction appellée, décomposition qui extrait le 2ème bloc à partir du 1ér et ansi de suite jusqu'à extraire tous les blocs..

```
> decomposition:=proc(M,p,q) local M1,n,m,Blocs,M2;
> M1:=M;n:=0;m:=0;Blocs:=NULL;
> while m<nops(M) do
> M2:=op(1,hachage(M1,p,q)):
> Blocs:=M2,Blocs:
> n:=op(2,hachage(M1,p,q)):
> m:=m+n:
> M1:=[seq(M[i],i=m+1..nops(M))]:
> od:
> return(convert(Message,base,10)[1],Blocs);
> end proc:
```

Vérifions sur un exemple.

```
> p:=13;q:=17;Message:=2395478955412;M:=convert(Message,base,10):
```

$$p := 13$$

$$q := 17$$

$$Message := 2395478955412$$

```
> Blocs:=decomposition(M,p,q);
```

$$Blocs := 2, 0, 39, 54, 78, 95, 54, 12$$

```
> Code:=seq([Blocs][i]^e mod p*q,i=1..nops([Blocs]));
```

$$Code := 32, 0, 65, 175, 91, 23, 175, 207$$

```
> Decodage:=seq([Code][i]^d mod p*q,i=1..nops([Code]));
```

$$Decodage := 2, 0, 39, 54, 78, 95, 54, 12$$

Ah Enfin ça marche, c'est joli la programmation non, pour terminer je vous laisse un exercice.

Ecrire un programme qui transforme les lettres en chiffres et un autre qui fait l'inverse. C'est pas difficile, il suffit de s'y mettre.

Source disponible sur :

http://www.chez.com/myismail