

Pràctiques Integrades

1er de Matemàtiques

Pràctica 19

curs 2002–03

19 Codificació de missatges amb RSA (II)

En la pràctica anterior heu pogut veure com funciona la base matemàtica per a poder encriptar missatges amb el sistema RSA. La finalitat d'aquesta pràctica és aplicar aquestes matemàtiques en un cas real (a més també veureu algunes funcions noves de Maple que poden resultar molt útils).

El primer que farem serà definir tots les mateixes funcions per a generar les claus públiques i privades i per a encriptar i desencriptar missatges. Per a generar un joc d'aquestes claus pública–privada es pot utilitzar la funció `generarcodi` següent:

```
> generarcodi:=proc(xifres)
> local p1,p2,m,n,phin,e,d;
> randomize();
> m:=floor(xifres/2+1);
> p1:=1;
> while p1<10^(m-1) do p1:=nextprime(rand(10^m())); end do;
> p2:=p1;
> while (p2=p1) or (p1*p2<10^xifres) do p2:=nextprime(rand(10^m())); end do;
> n:=p1*p2;
> phin:=(p1-1)*(p2-1);
> e:=phin;
> while igcd(e,phin)<>1 do e:=rand(10^6)() mod n; end do;
> d:=e^(-1) mod phin;
> [n,phin,e,d];
> end proc;
```

Aquesta funció té un únic argument `xifres` i produeix com a resultat una llista (delimitada amb `[ ]`) amb un nombre `n` que té, com a mínim, tantes xifres com el valor de l'argument `xifres` i és el producte de dos primers diferents `p1` i `p2` amb `xifres/2` xifres cada un; el producte `phin` de `(p1-1)` per `(p2-1)`; un exponent públic `e` que és un nombre arbitrari menor que 10^6 i coprimer amb el `phin`; i un exponent privat `d` que és l'invers de `e` mòdul `phin`. Observeu que s'utilitza la comanda `floor` (que calcula la part entera d'un nombre) per a poder ajustar la variable `m` i que la línia `while p1<10^(m-1) do p1:=...` faci que en `p1` s'hi trobi un nombre primer de més de `xifres/2` xifres; que la línia `while (p2=p1) or (p1*p2<10^xifres) do...` genera un segon primer `p2` que multiplicat per `p1` produeix un nombre de més de `xifres` xifres; i que `while igcd(e,phin)<>1 do...` genera un nombre de 6 xifres (o menys) que és coprimer amb el valor `phin`.

Les funcions `encriptarbloc` i `desencriptarbloc` següents són les que corresponen a la feina de la pràctica anterior. La primera té com arguments el missatge que es vol encriptar `missatge` i l'exponent `e` i el mòdul `n` públics per a donar com a resultat un nombre que representa el missatge encriptat. La funció de desencriptar té com arguments el nombre `numencript` que representa un missatge encriptat, l'exponent privat `d` i el mòdul públic `n`, i el resultat és el missatge que volem veure.

```

> encriptarbloc:=proc(missatge,e,n)
> local i,l1listanum,num,numencrypt;
> l1listanum:=convert(missatge,bytes);
> num:=sum(l1listanum[i+1]*256^i,i=0..nops(l1listanum)-1);
> numencrypt:=Power(num,e) mod n;
> numencrypt;
> end proc;

> descriptarbloc:=proc(numencrypt,d,n)
> local i,num,missatge,l1listanum;
> num:=Power(numencrypt,d) mod n;
> l1listanum:=convert(num,base,256);
> missatge:=convert(l1listanum,bytes);
> missatge;
> end proc;

```

Recordeu que aquestes funcions no permeten encriptar i descriptar missatges de qualsevol longitud. Si el missatge que es vol enviar és massa llarg (té més de $\log_{256}(n) = \frac{\ln(n)}{\ln(256)}$ caràcters), el nombre que el representa serà més gran que el mòdul n i en el moment de descriptar no el podrem recuperar. Si es volen enviar missatges de qualsevol longitud s'ha de dissenyar un mecanisme que parteixi el missatge en parts prou petites per a poder encriptar-les i descriptar-les de manera fiable.

Exercici 19.1

Definiu un procediment **particionar** que tingui com arguments una llista qualsevol **l1lista** i un nombre **m**, i doni com a resultat una nova llista formada per les llistes obtingudes considerant els diferents blocs consecutius de **m** elements de **l1lista** i per una última llista amb els elements que puguin quedar si el nombre d'elements de **l1lista** no és múltiple de **m**.

Per exemple, si **l1lista**:=**[1,2,3,4,5]** i **m**:= 2, el resultat de **particionar(l1lista,m)** hauria de ser la llista **[[1,2],[3,4],[5]]**, i si **m**:= 3 el resultat seria **[[1,2,3],[4,5]]**.

Un cop obtinguda aquesta funció que parteix un missatge en parts d'una longitud correcta, podeu veure com es pot utilitzar **particionar** i la comanda de Maple **map** per a modificar les funcions **encriptarbloc** i **descriptarbloc** per a obtenir les noves comandes **encriptar** i **descriptar** que poden encriptar i descriptar missatges de qualsevol longitud seguint el mateix mecanisme que estem utilitzant fins ara.

```

> encriptar:=proc(missatge,e,n)
> local long,l1listach,nums,l1listablocs;
> global particionar;
> l1listach:=convert(missatge,bytes);
> long:=floor(log(n)/log(256))-1;
> l1listablocs:=particionar(l1listach,long);
> nums:=map(x->sum(x[i+1]*256^i,i=0..nops(x)-1),l1listablocs);
> map(x->Power(x,e) mod n,nums);
> end proc;

```

```

> desencriptar:=proc(llistanumsencrpt,d,n)
> local llistanums, llistablocs, llistach;
> llistanums:=map(x->Power(x,d) mod n, llistanumsencrpt);
> llistablocs:=map(x->convert(x,base,256), llistanums);
> llistach:=map(op, llistablocs);
> convert(llistach,bytes);
> end proc;

```

Si mireu el que diu l'ajuda de Maple sobre la comanda `map` veureu que `map(funcio, llista)` aplica la funció `funcio` a cada un dels elements de la llista `llista`. Per exemple, si considereu `map(sin, [a,b,c])`; obtindreu com a resultat `[sin(a), sin(b), sin(c)]`. Per un altre costat, fixeu-vos també en que dins d'una comanda `map` es pot utilitzar una funció definida amb la notació `x -> expressio` (que és una funció sense cap nom assignat).

19.1 Cas pràctic

Amb les funcions de codificació i encriptació que teniu definides, ja es poden enviar i rebre missatges amb la seguretat de que, a part del destinatari i el remitent, ningú els pot llegir. Per a poder aplicar el mètode en un cas real disposeu d'una carpeta de la xarxa on hi podeu llegir i escriure tots vosaltres, dins aquesta carpeta podeu crear una subcarpeta amb el vostre nom (o qualsevol codi que us identifiqui) i posar dins aquesta subcarpeta un fitxer (creat amb qualsevol editor, per exemple NOTEPAD, i al que li podeu posar per nom `clau.txt`) que contingui la vostra *clau pública* (`e,n`) (generada amb la comanda de Maple corresponent). Quan algú us vulgui enviar un missatge només ha de recollir la vostra clau pública i utilitzar la dada per a encriptar el missatge, deixant el resultat (una llista de nombres més o menys llarga que per comoditat hauria de ser de la forma `[n1, n2, n3, ...]`) en la vostra carpeta en un altre fitxer de text que tingui un nom que identifiqui al remitent (o no!).

Recordeu que la seguretat del sistema està garantida pel temps que es necessita per a descompondre `n` en factors primers i que si s'utilitza una clau `n` massa petita qualsevol que tingui accés al contingut dels missatges encriptats i a les claus públiques podrà calcular les claus privades i desencriptar els missatges.

Exercici 19.2

Definiu un procediment `trencarcodi` que tingui com argument una clau pública (`e,n`) i obtingui com a resultat la clau privada `d` corresponent. Un cop fet això haurieu de poder desencriptar un missatge encriptat amb una certa clau pública (`e,n`) utilitzant `desencriptar(missatge, trencarcodi(e,n), n)`.

Nota: Si intenteu utilitzar `trencarcodi` amb un `n` de més de 30 xifres és molt probable que deixeu l'ordinador un xic penjadet.

19.1.1 Missatges amb signatura

Segurament ja deveu haver notat que amb aquest sistema d'encriptació hi ha la seguretat de que els missatges que s'envien només els pot llegir el destinatari autèntic però que no hi ha cap garantia sobre el remitent. És a dir, és molt fàcil enviar un missatge utilitzant una identitat falsa.

Una forma d'afegir una *signatura digital a un missatge* consisteix en afegir al text que es vol enviar el resultat d'encriptar amb la clau privada del remitent un text que identifiqui tant al missatge com a aquest remitent (sovint es repeteix el missatge que es vol enviar, o una part del missatge). D'aquesta manera, quan el destinatari desencripta el missatge troba tot el text del missatge pròpiament dit i, a més, una llista de

nombres que correspon a la signatura. Observeu que aquesta signatura es descripta amb la clau pública del remitent i, per tant, el destinatari pot comprovar si el remitent és l'autèntic.

Exemple 19.1

L'Anna, que té com a clau pública $e := 234281$, $n := 20131133042706409921684601$ i clau privada $d := 8592557834410647663644921$, vol quedar amb la Jessica, que té com a clau pública $e1 := 183643$, $n1 := 10701535163116170996766471$, per anar a dinar a les 14 hores. Encripta amb la seva clau privada la signatura "Quedem per dinar a les 14. De l'Anna per la Jessica." i obté

```
[8568033446836833340239, 9617208344215122129119482, 3569295709510289113864968,
10191676509591032189424162, 7327887385199906740741770, 5144324498380469293535799]
```

Seguidament redacta el missatge

```
"Quedem per dinar a les 14. De l'Anna per la Jessica. [8568033446836833340239,
9617208344215122129119482, 3569295709510289113864968, 10191676509591032189424162,
7327887385199906740741770, 5144324498380469293535799]"
```

l'encripta amb la clau pública de la Jessica i envia

```
[3563711015752004319402424, 10302110634942014127686446, 5179879773797379577822967,
201999477580341703418242, 9999366564979028193721266, 8764097362618589741135968,
7683139254169211311340606, 9524083387533957137565194, 9347924217402170216685381,
645848098745112872494709, 7097746577892742359072760, 4709368691474043888219700,
3027831765049390860565394, 2019554872802334481328513, 3735030092110383645909035,
4570576624440291186973375, 10658666321535282632543875, 7180206697277055734198727,
5960888968783855178496638, 8305018049790205930420916, 1560125811123715010751873,
9011004426070128228498872, 2183985496715709988142589, 1306070483254503631042304]
```

Quan la Jessica descripta amb la seva clau privada el missatge, tornarà a recuperar

```
"Quedem per dinar a les 14. De l'Anna per la Jessica. [8568033446836833340239,
9617208344215122129119482, 3569295709510289113864968, 10191676509591032189424162,
7327887385199906740741770, 5144324498380469293535799]"
```

i per a comprovar si el missatge és realment de l'Anna descriptarà amb

```
descriptar( [8568033446836833340239, 9617208344215122129119482,
3569295709510289113864968, 10191676509591032189424162, 7327887385199906740741770,
5144324498380469293535799], 234281, 20131133042706409921684601 );
```

la part de la signatura utilitzant la clau pública de l'Anna i obtindrà un altre cop el mateix missatge de forma que pot estar segura de que el missatge prové de l'Anna.