Cryptographie et Factorisation

> $with(numtheory)$:

Méthode RSA

```
codage :=proc(x, m, N);
 return modp(x^m, N);
 end:


codage2 :=proc(a, m, p);
 local y, x, n;
 y := 1;
 x := a;
 n := m;
 while n > 1 do
 if irem(n, 2) =0 then
   y := y;
   x := x^2 mod p;
   n := n/2;
 else y := y·x mod p;
        x := x^2 mod p;
        n := (n − 1)/2;
 fi;
 od;
 return y·x mod p;
 end:


decodage :=proc(y, m, N);
 local M, i, s, t;
 M := phi(N);
 i := igcdex(m, M, 's', 't');
s;
 if s < 0 then s := s + M;fi;
 return codage2(y, s, N);
 end;
```

proc(y, m, N) **(1.1)**

```
    local M, i, s, t;
    M := numtheory:-phi(N);
    i := igcdex(m, M, 's', 't');
    s;
    if s < 0 then s := s + M end if;
    return codage2(y, s, N)
end proc
```

Theoreme chinois : un exemple d'application

```
> M[1] := 485; M[2] := 621; M[3] := 47; igcd(M[1], M[2]); igcd(M[1], M[3]);
      igcd(M[2], M[3]); P := M[1]·M[2]·M[3]; x := (452, 1, 41);
```

$$M_1 := 485$$

$$M_2 := 621$$

$$M_3 := 47$$

$$1$$

$$1$$

$$1$$

$$P := 14155695$$

$$x := 452, 1, 41 \tag{1.1.1}$$

```
> for i from 1 to 3 do
  N[i] := P/M[i];
  igcdex(M[i], N[i],'s','t');
  t;
  X[i] := x[i]·t·N[i];
  od; X := sum(X[j], j=1..3);
  X mod M[1]; X mod M[2]; X mod M[3];
```

$$N_1 := 29187$$

$$1$$

$$223$$

$$X_1 := 2941932852$$

$$N_2 := 22795$$

$$1$$

$$58$$

$$X_2 := 1322110$$

$$N_3 := 301185$$

$$1$$

$$21$$

$$X_3 := 259320285$$

$$X := 3202575247$$

$$452$$

$$1$$

$$41 \tag{1.1.2}$$

Factorisation

naive :=**proc**(N);
 local M, i;
 $M :=$ floor(*evalf* (*sqrt*(N)));
 if $M^2 = N$ **then**
 return M, M;
 fi;
 for i **from** 2 **to** $N - 1$ **do**
 if *irem*(N, i) $= 0$ **then**
 return $\left(i, \frac{N}{i}\right)$;
 fi;
 od;
 return N *est premier*;
 end;

proc(N) **(2.1)**

local M, i;

$M :=$ floor(*evalf* (sqrt(N)));

if M^2 $= N$ **then return** M, M **end if**;

for i **from** 2 **to** $N - 1$ **do if** *irem*(N, i) $= 0$ **then return** $i, N/i$ **end if end do**;

return N* *est** *premier*

end proc

pollard :=**proc**(N);
 local $a, f,$ alpha, beta, c, p, g, rr;
 $rr :=$ *rand* (1 ..10); $a := rr(\ )$;
 $f := x \rightarrow x^2 + 1$; $g := x \rightarrow x^4 + 2 \cdot x^2 + 2$;
 alpha := *modp*($f(a), N$); beta := *modp*($g(a), N$); $p :=$ *gcd* (beta − alpha, N);
 if $p \neq 1$ **then return** $p, \frac{N}{p}$;**fi**;
 $c := 0$;
 while $c < 2 \cdot N$ **do**
 alpha := *modp*(f (alpha), N);

```
        beta := modp(g(beta), N);
        p := gcd(alpha-beta, N);
        c := c + 1;
    if p ≠ 1 and p ≠ N then return p, N/p;
    fi;
od;
return N est surement premier;
end:
```

Un exemple pour illustrer

$N := 19073522567805023543 9$; $x := 164451689468567$; $m := 97$;

$$19073522567805023543 9$$

$$164451689468567$$

$$97 \tag{3.1}$$

FACTORISATION

> *naive*(N);

Warning, computation interrupted

> *pollard*(N);

$$10000019,\ 19073486328181 \tag{3.2}$$

> *METHODE RSA*

> $c := codage2(x, m, N)$;

$$c := 15195389033611796321 5 \tag{3.3}$$

> $d := decodage(c, m, N)$;

$$d := 164451689468567 \tag{3.4}$$

> $x - d$;

$$0 \tag{3.5}$$

>