

Algorithms

1 The Miller-Rabin primality test

```
> MillerRabin := proc(n, a)
> if igcd(n, a) > 1 then
> return 'unsuitable base';
> fi;
> m := n-1;
> s := 0;
> while (m mod 2) = 0 do
> m := m/2;
> s := s+1;
> od;
> b := a &^ m mod n;
> if b = 1 then
> return 'possible prime';
> fi;
> if b = n-1 then
> return 'possible prime';
> fi;
> for i from 1 to s-1 do
> b := b^2 mod n;
> if b = 1 then
> return 'composite';
> fi;
> if b = n-1 then
> return 'possible prime';
> fi;
> od;
> return 'composite';
> end;
```

This tests the odd number n for primality using the base a . The **while**-loop computes s and odd m with $n - 1 = 2^s m$. Then b is initialized so that $b \equiv a^m \pmod{n}$. If $b \equiv \pm 1 \pmod{n}$ then “possible prime” is returned. Otherwise the **for**-loop computes $a^{2m}, a^{4m}, \dots, a^{2^{s-1}m}$ modulo n in turn. If

any of these is -1 modulo n then “possible prime” is returned, but if any is 1 modulo n , “composite” is returned as then $a^{2^{j-1}m} \not\equiv \pm 1 \pmod{n}$ but $(a^{2^{j-1}m})^2 \equiv 1 \pmod{n}$. If the loop terminates, $a^{2^{s-1}m} \not\equiv \pm 1 \pmod{n}$. In this case either $a^{n-1} = (a^{2^{s-1}m})^2 \not\equiv 1 \pmod{n}$ and n is composite, or $(a^{2^{s-1}m})^2 \equiv 1 \pmod{n}$ and again n must be composite, so “composite” is returned.

2 Pollard’s $p - 1$ factorization algorithm

```
> Pollard := proc(n, a, kmax)
> b := a;
> for k from 2 to kmax do
> b := b &^ k mod n;
> g := igcd(b-1, n);
> if g = n then
> return 'fail';
> fi;
> if g > 1 then
> return g;
> fi;
> od;
> return 'no factors found';
> end;
```

This procedure inputs a number n , believed to be composite, a base a , and an iteration bound $kmax$. It computes in turn $b \equiv a^{k!} \pmod{n}$ for each k from 2 up to $kmax$, and then $g = \gcd(b-1, n)$. Usually $g = 1$, but if $g = n$ we have been unlucky and we abort the procedure with the output “fail”. But if $1 < g < n$, we have succeeded and we can output the factor g of n . If the loop terminates, again we have been unlucky and we output “no factors found”. If we like we can then repeat the algorithm with a different base or an increased $kmax$.

3 Pollard's rho factorization algorithm

```
> PollardRho := proc(n, c, s, kmax)
> x := s;
> y := s;
> for k from 1 to kmax do
> x := x^2+c mod n;
> y := y^2+c mod n;
> y := y^2+c mod n;
> g := igcd(x-y, n);
> if g = n then
> return 'fail';
> fi;
> if g > 1 then
> return g;
> fi;
> od;
> return 'no factors found';
> end;
```

This procedure inputs a number n , believed to be composite, a functional parameter c , a seed s , and an iteration bound $kmax$. Define $x_0 = s$ and $x_{k+1} \equiv x_k^2 + c \pmod{n}$. The **for**-loop is iterated up to $kmax$ times, and computes $x = x_n$ and $y = x_{2n}$. Let $g = \gcd(x - y, n)$. Usually $g = 1$, but if $g = n$ we have been unlucky and we abort the procedure with the output “fail”. But if $1 < g < n$, we have succeeded and we can output the factor g of n . If the loop terminates, again we have been unlucky and we output “no factors found”. If we like we can then repeat the algorithm with a different seed or parameter or an increased $kmax$.

RJC 25/11/2003