

Given

- ▶ a secret  $f_0$  in  $\mathbb{F}_p$ ,
- ▶  $n$  players,
- ▶ and a threshold  $t \leq n$ , which indicates how many players are required to recover the secret.

Choose  $n + t - 1$  random elements

$f_1, \dots, f_{t-1}, u_0, \dots, u_{n-1} \xleftarrow{\text{random}} \mathbb{F}_p$  uniformly and independently with all  $u_i$  nonzero and pairwise distinct,

set  $f = f_{t-1}x^{t-1} + \dots + f_1x + f_0 \in \mathbb{F}_p[x]$ , and give to player number  $i$  the values  $u_i$  and  $f(u_i)$  in  $\mathbb{F}_p$ .

Then

- ▶ each subset of  $t$  (or more) players can recover the secret  $f_0$ , by interpolating  $f$ ,
- ▶ but fewer than  $t$  players cannot recover the secret.

## Theorem

There are infinitely many primes.

ALGORITHM 1. Fermat test.

Input: A number  $N \in \mathbb{Z}$  with  $N \geq 2$ .

Output: Either “ $N$  is composite”, or “ $N$  is possibly prime”.

1.  $x \leftarrow \{1, \dots, N - 1\}$ .
2.  $g \leftarrow \gcd(x, N)$ . If  $g \neq 1$ , then Return “ $N$  is composite”.
3.  $y \leftarrow x^{N-1}$  in  $\mathbb{Z}_N$ .
4. If  $y \neq 1$ , then Return “ $N$  is composite”.
5. Return “ $N$  is possibly prime”.

ALGORITHM 2. Strong pseudoprimality test.

Input: A number  $N \in \mathbb{Z}$  with  $N \geq 2$ .

Output: Either “ $N$  is composite”, or “ $N$  is probably prime”.

1. Write  $N - 1 = 2^e m$ , where  $m$  is odd.
2.  $x \xleftarrow{\text{rand}}$   $\{1, \dots, N - 1\}$ .
3. If  $\gcd(x, N) \neq 1$  then Return “ $N$  is composite”.
4.  $y \leftarrow x^m$  in  $\mathbb{Z}_N$ .
5. If  $y = 1$  then Return “probably prime”.
6. For  $i$  from 0 to  $e - 1$  do steps 7.-8.
7. If  $y = -1$  then Return “probably prime”,
8. otherwise  $y \leftarrow y^2$  in  $\mathbb{Z}_N$ .
9. Return “is composite”.

$N$	$N - 1 = 2^e \cdot m$	$y_0$	$y_1$	$y_2$	$y_3$	$y_4$
553	$2^3 \cdot 69$	407	302	512	22	
557	$2^2 \cdot 139$	556	1			
$561 = 3 \cdot 11 \cdot 17$	$2^4 \cdot 35$	56	331	116	67	1

Table: Testing 553, 557, and 561.

The *strong pseudoprimality test* has the following properties.

- (i) If  $N$  is prime, the test returns “probably prime”.
- (ii) If  $N$  is composite and not a Carmichael number, the test returns “composite” with probability at least  $1/2$ .
- (iii) If  $N$  is a Carmichael number, the test returns a proper factor of  $N$  with probability at least  $1/2$ .
- (iv) For an  $n$ -bit input  $N$ , the test uses  $O(n^3)$  bit operations.

The strong pseudoprimality test , repeated  $t$  times independently, has the following properties on input  $N$ .

- (i) If it outputs “composite”, then  $N$  is composite.
- (ii) If it outputs “probably prime”, then  $N$  is prime with probability at least  $1 - 2^{-t}$ .

### ALGORITHM 3. Finding a pseudoprime.

Input: An integer  $n$  and a confidence parameter  $t$ .

Output: A number  $N$  in the range  $[2^{(n-1)/2}, 2^{n/2}]$ .

1.  $x \leftarrow 2^{(n-1)/2}$ .
2. Repeat steps 3 and 4 Until some  $N$  is accepted.
3.  $N \leftarrow \{ \lceil x \rceil, \dots, \lfloor \sqrt{2}x \rfloor \}$ .
4. Call the strong pseudoprimality test with input  $N$  for  $t$  independently chosen  $x \leftarrow \{1, \dots, N-1\}$ . Accept  $N$  if and only if all these tests Return “probably prime”. Goto step 3 if the test answers “composite”.
5. Return  $N$ .

## Example

The smallest Carmichael number is  $N = 561 = 3 \cdot 11 \cdot 17$ , for which  $N - 1 = 560 = 2^4 \cdot 5 \cdot 7$  and  $\phi(N) = 2 \cdot 10 \cdot 16 = 320$ . For any  $a \in \mathbb{Z}_N^\times$ , we have  $a^2 = 1$  in  $\mathbb{Z}_3$ ,  $a^{10} = 1$  in  $\mathbb{Z}_{11}$ , and  $a^{16} = 1$  in  $\mathbb{Z}_{17}$ , hence  $a^{80} = 1$  in  $\mathbb{Z}_3, \mathbb{Z}_{11}$ , and  $\mathbb{Z}_{17}$ , and by the Chinese Remainder Theorem also in  $\mathbb{Z}_N$ . Then also  $a^{N-1} = x^{560} = (x^{80})^7 = 1$  in  $\mathbb{Z}_N^\times$ .

## Prime Number Theorem

We have approximately

$$\pi(x) \approx \frac{x}{\ln x}, \quad \vartheta(x) \approx x, \quad p_n \approx n \ln n,$$

and more precisely

$$\frac{x}{\ln x} \left(1 + \frac{1}{2 \ln x}\right) < \pi(x) < \frac{x}{\ln x} \left(1 + \frac{3}{2 \ln x}\right) \text{ for } x \geq 59,$$

$$\frac{3x}{5 \ln x} < \pi(2x) - \pi(x) < \frac{7x}{5 \ln x} \text{ for } x \geq 21,$$

$$n \left(\ln n + \ln \ln n - \frac{3}{2}\right) < p_n < n \left(\ln n + \ln \ln n - \frac{1}{2}\right) \text{ for } n \geq 20,$$

$$x \left(1 - \frac{1}{2 \ln x}\right) < \vartheta(x) < x \left(1 + \frac{1}{2 \ln x}\right) \text{ for } x \geq 563.$$

## Theorem

On input  $n \geq 11$  and  $t$ , the output of the Algorithm *Finding a pseudoprime* is prime with probability at least  $1 - 2^{-t+1} n^{-1}$ . It uses an expected number of  $O(tn^4)$  bit operations.

$N$	$c_N/N^2$
10	0.63
100	0.6087
1000	0.608383
10 000	0.60794971
100 000	0.6079301507

**Table:** The probabilities that two random positive integers below  $N$  are coprime.

Find  $n/2$ -bit primes at random  $O(n^4 \log n)$ ,  
Find  $e$   $O(n^2 \log n)$ ,  
Calculate  $N$  and  $d$   $O(n^2)$ ,  
Calculate powers modulo  $N$   $O(n^3)$ .

We consider as an attacker a (random) polynomial-time computer  $\mathcal{A}$ .  $\mathcal{A}$  knows  $pk = (N, e)$  and  $y = \text{enc}_{pk}(x)$ . There are several notions of “breaking RSA”.  $\mathcal{A}$  might be able to compute from its knowledge one of the following data.

$B_1$ : the plaintext  $x$ ,

$B_2$ : the hidden part  $d$  of the secret key  $S = (N, d)$

$B_3$ : the value  $\phi(N)$  of Eulers totient function,

$B_4$ : a factor  $p$  (and  $q$ ) of  $N$ .

1. If  $A$  and  $B$  are two computational problems (given by an input/output specification), then a *random polynomial-time reduction* from  $A$  to  $B$  is a random polynomial-time algorithm for  $A$  which is allowed to make calls to an (unspecified) subroutine for  $B$ . The cost of such a call is the combined input plus output length in the call. If such a reduction exists, then  $A$  is *random polynomial-time reducible to  $B$* , and we write

$$A \leq_p B.$$

If such a reduction exists that does not make use of randomization, then  $A$  is *polynomial-time reducible to  $B$* .

2. If also  $B \leq_p A$ , we call  $A$  and  $B$  *random polynomial-time equivalent* and write

$$A \equiv_p B.$$

## Wiener's attack

In the RSA notation, suppose that  $p < q < 2p$ ,  $1 \leq e < \phi(N)$ , and  $1 \leq d \leq N^{1/4}/\sqrt{12}$ . Then  $d$  can be computed from the public data in time  $O(n^2)$ .

We can take the remainder  $a_p \in \mathbb{Z}_{p-1} = \{0, \dots, p-2\}$  of any  $a \in \mathbb{Z}_{\varphi(N)}$ . Then we can simplify the computation of  $x = y^d$  in  $\mathbb{Z}_N$  by keeping  $p$  and  $q$ , computing

$$\begin{aligned}d_p = d \text{ in } \mathbb{Z}_{p-1}, & \quad x_p = y^{d_p} \text{ in } \mathbb{Z}_p, \\d_q = d \text{ in } \mathbb{Z}_{q-1}, & \quad x_q = y^{d_q} \text{ in } \mathbb{Z}_q,\end{aligned}$$

$$x = \begin{cases} x_p \text{ in } \mathbb{Z}_p, \\ x_q \text{ in } \mathbb{Z}_q. \end{cases}$$

Then  $x = y^d$  in  $\mathbb{Z}_N$ , but the computation becomes cheaper.

method	year	time
trial division	$-\infty$	$2^{n/2}$
Pollard's $p - 1$ method	1974	$2^{n/4}$
Pollard's $\rho$ method	1975	$2^{n/4}$
Pollard's and Strassen's method	1976	$2^{n/4}$
Morrison's and Brillhart's continued fractions	1975	$\exp(n^{1/2})$
Dixon's random squares, quadratic sieve	1981	$\exp(n^{1/2})$
Lenstra's elliptic curves	1987	$\exp(n^{1/2})$
number field sieve	1990	$\exp(n^{1/3})$

## Birthday paradox

We consider random choices, with replacement, among  $p$  labeled balls. The expected number of choices until a collision occurs is  $O(\sqrt{p})$ .

We want to factor  $N = 82123$ . Starting with  $x_0 = 631$ , we find the following sequence:

$i$	$x_i \bmod N$	$x_i \bmod 41$
0	631	16
1	69670	11
2	28986	40
3	69907	2
4	13166	5
5	64027	26

$i$	$x_i \bmod N$	$x_i \bmod 41$
6	40816	21
7	80802	32
8	20459	0
9	71874	1
10	6685	2

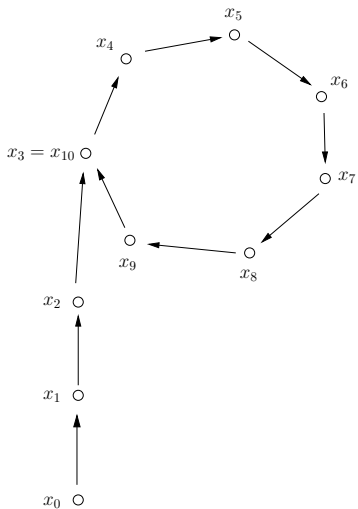


Figure: Pollard's  $\rho$  method.

ALGORITHM 4. Pollard's  $\rho$  algorithm.

Input:  $N \in \mathbb{N}_{\geq 3}$ , neither a prime nor a perfect power.

Output: Either a proper divisor of  $N$ , or "failure".

1. Pick  $x_0 \xleftarrow{\$} \mathbb{Z}_N$  at random,  $y_0 \leftarrow x_0$ ,  $i \leftarrow 0$
2. Repeat
3.  $i \leftarrow i + 1$ ,  $x_i \leftarrow x_{i-1}^2 + 1$  in  $\mathbb{Z}_N$ ,  
 $y_i \leftarrow (y_{i-1}^2 + 1)^2 + 1$  in  $\mathbb{Z}_N$
4.  $g \leftarrow \gcd(x_i - y_i, N)$   
If  $1 < g < N$  then Return  $g$   
If  $g = N$  then Return "failure"

## Theorem

Let  $N \in \mathbb{N}$  be a composite  $n$ -bit integer,  $p$  its smallest prime factor, and  $f(x) = x^2 + 1$ . Under the assumption that the sequence  $(f^i(x_0))_{i \in \mathbb{N}}$  behaves modulo  $p$  like a random sequence, the expected number of iterations in Pollard's algorithm for returning a proper factor of  $N$  is  $O(\sqrt{p})$ . By applying the algorithm recursively,  $N$  can be completely factored with an expected number of  $O(N^{1/4}n^2)$ , bit operations.

The algorithm calculates in tandem  $x_i$  and  $y_i = x_{2i}$  and performs the gcd test each time. We have  $t = 3$ ,  $\ell = 7$ , and  $t + (-t\ell) = 7$ , and in fact, after seven 2-steps the algorithm catches up with the 1-steps:

$i$	$x_i \bmod N$	$x_i \bmod 41$	$y_i \bmod N$	$y_i \bmod 41$	$\gcd(x_i - y_i, N)$
0	631	16	631	16	82123
1	69670	11	28986	40	1
2	28986	40	13166	5	1
3	69907	2	40816	21	1
4	13166	5	20459	0	1
5	64027	26	6685	2	1
6	40816	21	75835	26	1
7	80802	32	17539	32	41

The factorization  $N = 41 \cdot 2003$  is found as  $\gcd(x_7 - y_7, N) = 41$ .

Let  $N = 2183$ . Suppose that we have found the equations

$$\begin{aligned}453^2 &= 7, \\1014^2 &= 3 \\209^2 &= 21.\end{aligned}$$

Then we obtain  $(453 \cdot 1014 \cdot 209)^2 = 21^2$  in  $\mathbb{Z}_N$ , or  $687^2 = 21^2$  in  $\mathbb{Z}_N$ . This yields the factors  $37 = \gcd(687 - 21, N)$  and  $59 = \gcd(687 + 21, N)$ ; in fact,  $N = 37 \cdot 59$  is the prime factorization of  $N$ .

ALGORITHM 5. Dixon's random squares method.

Input: An integer  $N \geq 3$ , and  $B \in \mathbb{N}_{\geq 2}$ .

Output: Either a proper divisor of  $N$ , or "failure".

1. Compute all primes  $p_1, p_2, \dots, p_h$  up to  $B$ .
2. If  $p_i$  divides  $N$  for some  $i \in \{1, \dots, h\}$  then Return  $p_i$ .
3.  $A \leftarrow \emptyset$ .
4. Repeat 5 - 10
5. Choose a uniform random number  $b \leftarrow \mathbb{Z}_N \setminus \{0\}$ .
6.  $g \leftarrow \gcd(b, N)$ , If  $g > 1$  then Return  $g$ .
7.  $a \leftarrow b^2 \in \mathbb{Z}_N$ .
8. For  $i = 1, \dots, h$  do
9.  $\alpha_i \leftarrow 0$ , While  $p_i$  divides  $a$  do  $a \leftarrow \frac{a}{p_i}$ ,  $\alpha_i \leftarrow \alpha_i + 1$ .
10. If  $a = 1$ , then  $\alpha \leftarrow (\alpha_1, \dots, \alpha_h)$ ,  $A \leftarrow A \cup \{(b, \alpha)\}$ .
11. Until  $\#A = h + 1$ .
12. Find distinct pairs  $(b_1, \alpha^{(1)}), \dots, (b_\ell, \alpha^{(\ell)}) \in A$  with  $\alpha^{(1)} + \dots + \alpha^{(\ell)} = 0$  in  $\mathbb{Z}_2^h$ , for some  $\ell \geq 1$ , by solving an  $(h + 1) \times h$  system of linear equations over  $\mathbb{F}_2$ .
13.  $(\delta_1, \dots, \delta_h) \leftarrow \frac{1}{2}(\alpha^{(1)} + \dots + \alpha^{(\ell)})$ .
14.  $x \leftarrow \prod_{1 \leq i \leq \ell} b_i$ ,  $y \leftarrow \prod_{1 \leq j \leq h} p_j^{\delta_j}$ ,  $g \leftarrow \gcd(x + y, N)$ .
15. If  $g < N$  then Return  $g$  Else Return "failure".