INTEGER FACTORING

The problem of distinguishing prime integers from composite ones is reasonably well understood. Agrawal, Kayal and Saxena have presented in [1] a deterministic polynomial time algorithm for this. The probabilistic test of Miller [4] and Rabin [5] is very convenient for practical purposes. On the other hand there is no known polynomial time algorithm for factoring integers with a deterministic Turing machine nor with a probabilistic one. Peter Shor [7] has proposed a polynomial time algorithm for factoring integers with a quantum computer if ever such a computer exists. For example if $n = pq$ where p and q are two distinct large primes, it is easy to multiply p and q to compute n. But recovering p and q from n is a challenge. This observation provides a candidate asymmetric function. There is a competition called RSA-challenge for factoring such composite numbers

`https://en.wikipedia.org/wiki/RSA_Factoring_Challenge`

The largest integer factored in this list has 232 digits. Its factorization was achieved by a team of computer scientists led by Kleinjung [3]. It used many hundreds of machines and took almost two years. "On a single core 2.2 GHz AMD Opteron processor with 2 GB RAM, sieving would have taken about fifteen hundred years."

We will revew a few algorithms of various nature for factoring integers. Some are very elementary and some are more sophisticated. Understanding the theoretical complexity and the practical efficiency of these algorithms is crucial to the security of many cryptographic schemes, the most famous beeing RSA.

1. THE RSA SCHEME

This is a public key encryption scheme invented by Rivest, Shamir and Adleman [6] in 1978 and a few years before by Cocks who did not disclose it because he was working for a British intelligence agency. The security of this cryptosystem relies on the difficulty of computing e-th roots modulo a composite number n when the factorization of n is not known. RSA is a generalization of a cryptosystem invented by Rabin that relied on the difficulty of computing square roots.

Let p and q be two distinct large primes and set $n = pq$. Let $\lambda = \mathrm{ppcm}(p-1, q-1)$. According to Chinese theorem, the group $((\mathbb{Z}/n\mathbb{Z})^*, \times)$ is isomorphic to

$$((\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*, \times) \sim ((\mathbb{Z}/(p-1)\mathbb{Z}) \times (\mathbb{Z}/(q-1)\mathbb{Z}), +)$$

so it has **exponent** λ. In words λ is the smallest positive integer such that for every x in $(\mathbb{Z}/n\mathbb{Z})^*$ one has $x^\lambda = 1 \bmod n$.

We describe RSA on a toy example. The primes involved in this example are not large enough to ensure security. We just want to illustrate the method.

Alice first generates her keys. She chooses two large enough prime integers p and q at random.

```
gp >p=nextprime(random(2^20))
%1 = 761669
gp >q=nextprime(random(2^20))
%2 = 341729
```

Alice now computes the product $n = pq$ and the exponent $\lambda = \mathrm{ppcm}(p-1, q-1)$.

```
gp > n=p*q
%3 = 260284385701
gp > L=lcm(p-1,q-1)
%4 = 65070820576
```

She chooses an integer e in $[2, \lambda]$ such that e is prime to λ. It is safer to pick a random e with uniform probability. Alice may prefer to select a small e to accelerate ciphering. We assume that she takes $e = 3$ this time :

```
gp > e=3
%5 = 3
gp > gcd(e,L)
%6 = 1
```

Alice computes the inverse f of $e \bmod \lambda$

```
gp > f=(1/e)% L
%7 = 43380547051
```

Alice publishes (n, e) in the phone-book for example. This is called the ciphering key or the public key. Alice keeps f secret. This is called the unciphering key or the secret key.

Assume now that Bob wants to send a message to Alice. The clear text of this message is $m = 1234567$.

Bob finds Alice's public key in the phonebook : $n = 260284385701$ and $e = 3$. He computes the cipher-text $c = m^e \bmod n$

```
gp > m=1234567
%1 = 1234567
gp > n=260284385701
%2 = 260284385701
gp > e=3
%3 = 3
gp > c=Mod(m,n)^e
%4 = Mod(214733022870, 260284385701)
```

He sends $c = 214733022870$ to Alice.

Alice receives the cipher-text c from Bob. She computes $c^f \bmod n$ using the secret key f

```
gp > c=214733022870
%1 = 214733022870
gp > f=43380547051
%2 = 43380547051
gp > n=260284385701
%3 = 260284385701
```

```
gp > Mod(c,n)^f
%4 = Mod(1234567, 260284385701)
```

The security of the RSA crypto-system relies on the difficulty of computing e-th roots modulo n without knowing the factorization of n. Practical implementations of RSA are not so simple and require at least the addition of a random suffix to the clear-text before enciphering. The complexity of enciphering is $O((\log n)^3)$ using ordinary arithmetic if the exponent e is any integer between 1 and λ.

2. TRIAL DIVISION

A simple minded method to factor an integer n is to first check that n is not a prime and then compute the euclidean divisions (quotient and remainder) of n by 2, 3, 5, 7, 9, 11, 13, 15, ... It is possible to accelerate a little bit using a sieve or a table of small primes. The first found factor is a prime divisor p of n. We then rerun the algorithm with n replaced by n/p. The complexity is $n^{1/2+o(1)}$ which is very bad unless n is small.

3. THE METHOD $p-1$ OF POLLARD

This method is efficient only when the integer n to be factored has a very specific form. However the ideas underlying this method will be useful in a wider context and deserve to be presented.

Assume n is the product of two primes p and q and that $p-1$ is a **smooth** integer. This means that $p-1$ has only small prime factors. For example consider the case $n = 713 = pq$ with $p = 31$ and $q = 23$. Then $p - 1 = 2 \times 3 \times 5$ is y-smooth with $y = 5$.

We pick a random residue a modulo n. For example set $a = 2 \bmod n$. We define a sequence $a_1 = a$, $a_2 = a_1^2 \bmod n$, $a_3 = a_2^3 \bmod n$, $a_4 = a_3^4 \bmod n$, $a_5 = a_4^5 \bmod n$, ... So a_{k+1} is the $k+1$ power of a_k modulo n. So $a_k = a^{k!}$. For every k one computes the gcd of $a_k - 1$ and n. If $p-1$ is smooth then most likely $p-1$ divides $k!$ for some small k and we will find a non trivial $\gcd(a_k - 1, n)$. Indeed $a^{p-1} = 1 \bmod p$ so $a_k = x^{k!} = 1 \bmod p$ and there is no reason why $x^{k!}$ should be 1 modulo q.

k	1	2	3	4	5	6	7
$a_k = a^{k!} \bmod n$	2	4	64	326	311	32	280
$\gcd(a_k - 1, n)$	1	1	1	1	31	31	31

This method is very interesting if n has a small prime factor p because $p-1$ is much likely to be smooth in that case.

There exists a variant to the $p-1$ method of Pollard, called the *large prime variation*. It succeeds whenever $p-1$ is the product of a y_1-smooth number by a single prime in the interval $]y_1, y_2]$. Such an integer is said to be (y_1, y_2)-smooth. One starts as before and computes $a_{k+1} = a_k^{k+1} \bmod n$ for $1 \leqslant k \leqslant y_1 - 1$. We set $b_0 = a_{y_1}$. We then compute $b_1 = b_0^{l_1} \bmod n$, $b_2 =$

$b_0^{l_2} \bmod n$, $b_3 = b_0^{l_3} \bmod n$, $b_4 = b_0^{l_4} \bmod n$, ..., where the l_i are the successive primes in the interval $]y_1, y_2]$. The trick is that in order to compute b_2 we do not need to exponentiate. Indeed

$$b_2 = b_0^{l_2} = b_0^{l_1+(l_2-l_1)} = b_1 b_0^{l_2-l_1} \bmod n.$$

Since the difference between two consecutive primes is expected to be very small, we precompute the first powers of b_0 and set $c_j = b_0^j$. We then compute

$$b_{k+1} = b_k c_{l_{k+1}-l_k} \bmod n$$

at the expense of a single multiplication.

For example, if $n = 2721749 = pq$ where $p = 2671$ and $q = 1019$ then $p - 1 = 2 \times 3 \times 5 \times 89$ is (y_1, y_2)-smooth with $y_1 = 7$ and $y_2 = 100$.

We compute $a_1 = 2$, $a_2 = a_1^2 = 4$, $a_3 = 64$, $a_4 = 446722$, $a_5 = 1416863$, $a_6 = 715291$, $a_7 = 795854$.

We set $b_0 = a_7$. We then raise b_0 to the powers l_i for l_i a prime in $[11, 97]$. The difference between two consecutive primes in this interval is at most 8.

We precompute $c_1 = b_0 = 795854$, $c_2 = (b_0)^2 = 2657777$, $c_3 = (b_0)^3 = 664706$, $c_4 = 1628037$, $c_5 = 2034144$, $c_6 = 1664270$, $c_7 = 1281471$, $c_8 = (b_0)^8 = 2696942$.

We now compute $b_1 = (b_0)^{11} = 1715449$, $b_2 = (b_0)^{13} = b_1 c_2 = 216252$, $b_3 = (b_0)^{17} = b_2 c_4 = 2580676$, ..., $b_{20} = (b_0)^{89} = b_{19} c_6 = 1279410$.

We find $\gcd(b_{20} - 1, n) = 2671$.

4. Fermat's method

Fermat noticed that if one manages to write an integer n as a difference of two squares then one gets a chance to factor n. Indeed if $n = x^2 - y^2$ then $n = (x - y)(x + y)$.

Assume for example that we want to factor $n = 1524157896661027288525081$. We add to n the succcessive squares 1, 4, 9, ... and we check if the sum is a square :

```
gp > n=1524157896661027288525081
%1 = 1524157896661027288525081
gp > for(k=1,20,if(issquare(n+k^2),print([k,sqrt(n+k^2)])))
[12, 1234567898765.000000000000000]
```

So $n = -12^2 + 1234567898765^2 = 1234567898753 \times 1234567898777$.

This method only works if one of the squares is small. In that case, the other square will be close to $\sqrt{n}$. So we may compute the smallest integer r bigger than $\sqrt{n}$ and check if $r^2 - n$ is a square also.

```
 gp > round(sqrt(n))^2-n
%2 = 144
```

If we manage to write a multiple of n as a difference of two squares then again we can hope to factor n. Indeed, if $x^2 = y^2 \bmod n$ we may hope that the gcd of n and $x - y$ is non-trivial. So we can apply the computation above to the first multiples of n. This method only succeeds for very special values of n. It is sometimes efficient for small n.

For general values of n, the best known method to produce a congruence

$$x^2 - y^2 = 0 \bmod n$$

is to collect many congruences between a square and a smooth number that is

$$\prod_i p_i = x^2 \bmod n$$

where x is any integer and the p_i are prime integers smaller than or equal to a given bound B. Once collected enough such congruences we can, using linear algebra, find a multiplicative combination of these congruences with a square on the right hand side also. The next two sections present two simple algorithms based on this idea.

5. DIXON LINEAR SIEVE

Pick a random a modulo n and let b be the unique integer congruent to a^2 modulo n in the interval $[-(n-1)/2, (n-1)/2]$. We thus have $a^2 = b \bmod n$. We hope that b is B-smooth. If this is the case we obtain a congruence between a square and a B-smooth integer. We collect enough such relations and we combine them to obtain a congruence between two squares.

Assume for example that we want to factor $n = 7081$. We choose $B = 3$. The B-smooth integers are of the form $\pm 2^a 3^b$.

After a few random trials we find

$$\begin{aligned} 4486^2 &= -2.3 \bmod n, \\ 1857^2 &= 2 \bmod n, \\ 2645^2 &= -3 \bmod n. \end{aligned}$$

We collect the valuations of the right hand sides in a 3×3 matrix M

	-1	2	3
4486	1	1	1
1857	0	1	0
2645	1	0	1

If we reduce M modulo 2 we obtain a matrix with coefficients in the field with two elements. We then look for the kernel of this matrix M modulo 2. We check that the line $r = [1,1,1] \in \mathbb{F}_2^3$ verifies $rM = [0,0,0]$. We deduce the congruence

$$(4486.1857.2645)^2 = (-2.3)^2 \bmod n.$$

The gcd of n and $4486.1857.2645 + 6$ is 73. We have found a non-trivial factor of n.

6. THE QUADRATIC SIEVE

This algorithm is due to Carl Pomerance. We illutrate it on an example. Let

$$n = 21311 = 101.211$$

be the number to be factored. We find an integer m close to the square root of n

$$m = \lfloor n^{1/2} \rceil = 146.$$

We produce lots of congruences modulo n by observing that for any integer a,

$$(m+a)^2 \equiv (m^2 - n) + a^2 + 2am \pmod{n} = 5 + a^2 + 292a \pmod{21311},$$

where $m^2 - n$ has order of magnitude $\sqrt{n}$.

We choose a smoothness bound $B = 13$ and look for small integers a such that $5 + a^2 + 292a$ is B-smooth. For example for a between -60 and 60 we find

a	$5 + 292a + a^2$
-27	$-2.5^2.11.13$
-5	$-2.5.11.13$
-1	$-2.11.13$
0	5
60	$5^3.13^2$

We report the parity of the valuations in a matrix with coefficients in $\mathbb{Z}/2\mathbb{Z}$:

	-1	2	5	11	13
-27	1	1	0	1	1
-5	1	1	1	1	1
-1	1	1	0	1	1
0	0	0	1	0	0
60	0	0	1	0	0

Each row in the kernel of this matrix produces a congruence modulo n between two squares. The kernel here has dimension three and a basis for it consists of the three rows below

-27	-5	-1	0	60
1	0	1	0	0
1	1	0	1	0
1	1	0	0	1

The first row produces the congruence

$$(2.5.11.13)^2 \equiv (146-27)^2.(146-1)^2 \pmod{21311}.$$

We compute the gcd of $2.5.11.13-(146-27).(146-1) = -15825$ and 21311. We find the non-trivial factor $p = 211$ of $n = 21311$ and its cofactor $q = 101$. A primality test shows that p and q are prime integers.

Which is the advantage of this algorithm compared to Dixon's linear sieve ? The only difference concerns the way we produce congruences modulo n. In Dixon's sieve the residue x^2 modulo n is an integer in the range $[-(n-1)/2, (n-1)/2]$. We hope that this interger is smooth. In the quadratic sieve the number that we hope to be smooth has order of magnitude $\sqrt{n}$. The probability of success is higher.

7. Density of smooth integers

The efficiency of sieving algorithms relies on the expectation that the proportion of smooth integers is not too small. We state a theorem of Canfield, Erdös, Pomerance about this and sketch a proof of it.

Let $\mathbb{Z}^*$ be the set of non-zero integers. We consider the function $t : \mathbb{Z}^* \to \mathbb{R}$ defined by

$$t(n) = \log(|n|).$$

We say that $t(n)$ is the **size** of n. The size a product of non-zero integers is the sum of their sizes.

We say that an integer n is y-smooth if it is a product of integers $\leqslant y$. Equivalently all prime divisors of b are $\leqslant y$. For example $330 = 2 \times 3 \times 5 \times 11$ is 13-smooth and even 11-smooth.

Let $x \geqslant 2$ and $y \geqslant 2$ be two integers. We want to estimate the proportion of y-smooth integers among all integers $\leqslant x$. We give a simplistic argument.

Let

$$\ell = \lfloor t(x)/t(y) \rfloor = \lfloor \log(x)/\log(y) \rfloor$$

be the biggest integer lower than or equal to the quotient $t(x)/t(y)$ of the sizes of x and y.

Pick ℓ positive prime integers a_1, a_2,..., a_ℓ, all $\leqslant y$.

The product $a_1 a_2 \dots a_\ell$ is $\leqslant x$ and it is y-smooth.

We want to count the number of y-smooth integers we have constructed this way. According to the prime numbers theorem the proportion of prime integers among the positive integers $\leqslant y$ is

$$\frac{1}{\log(y)}(1+\epsilon(y))$$

where ϵ is a function in the class $o(1)$. We even know that this proportion is

$$\geqslant \frac{1}{\log(y)}$$

as soon as $y \geqslant 52$. See [8, Chapitre I.1]. So the number of ℓ-uples $(a_1, ..., a_\ell)$ is

$$\geqslant \frac{y^\ell}{(\log y)^\ell} \geqslant \frac{x}{y} \times \frac{1}{(\log(y))^\ell} \geqslant x^{1-\frac{1}{\ell}} \times \frac{1}{(\log(y))^\ell},$$

as soon as $y \geqslant 52$.

Since multiplication is commutative, all the permutations of a given ℓ-uple $(a_1, \ldots, a_\ell)$ give rise to the same product. In fact two ℓ-uples have the same product if and only if they are equal up to permutation of their coordinates. This is the fundamental theorem of arithmetic. So we have at most $\ell!$ different ℓ-uples that give rise to the same product. So we have constructed at least

$$\frac{x}{\ell! x^{\frac{1}{\ell}} (\log(y))^\ell}$$

y-smooth elements in the interval $[1, x]$. The proportion of y-smooth integers in $[1, x]$ is thus at least

$$\frac{1}{\ell! x^{\frac{1}{\ell}} (\log(y))^\ell}$$

The most significant factor in the denominator is the $\ell!$. A careful study [2] proves the following theorem.

Theorem 7.1 (Canfield, Erdös, Pomerance)**.** *Let $\Psi(x, y)$ be the number of y-smooth integers in the interval $[1, x]$. Set $u = \frac{\ln x}{\ln y}$. Let $\varepsilon \in]0, 1[$ be fixed. There exists a function μ in the $o(1)$ class such that*

$$\Psi(x, y) = xu^{-u(1+\mu(u))}$$

whenever $(\ln x)^\varepsilon < u < (\ln x)^{1-\varepsilon}$.

In the typical case when $\log y = A.\sqrt{\log(x)\log\log(x)}$ for some positive constant A, we find $u = \frac{1}{A}\sqrt{\frac{\log x}{\log\log x}}$ and $-u \log u = -\frac{1+o(1)}{2A}\sqrt{\log(x)\log\log(x)}$.

Theorem 7.2. *Let A be a positive real number. Let x be a large enough integer and set*

$$y = \exp\left(A.\sqrt{\log(x)\log\log(x)}\right).$$

The proportion of y-smooth integers in the interval $[1, x]$ is at least

$$\exp\left(-\frac{1+o(1)}{2A}\sqrt{\log(x)\log\log(x)}\right).$$

The proportion of y-smooth integers in the interval $[1, x]$ is neither too large nor too small. There are many more smooth integers than squares for example. But much less than primes.

8. Complexity analysis of the linear sieve

We first study the complexity of Dixon's linear sieve. In this algorithm we consider random integers in the range $[1, n]$ and we hope to find enough B-smooth ones. Assume

$$B = \exp(A.\sqrt{\log(n)\log\log(n)})$$

for some positive real constant A. Then the probability that an integer in the range $[1, n]$ is B-smooth is

$$P = \exp\left(-\frac{1+o(1)}{2A}\sqrt{\log(n)\log\log(n)}\right).$$

The time needed to find one congruence between a square and a B-smooth integer is thus

$$\frac{1}{P} = \exp\left(\frac{1+o(1)}{2A}\sqrt{\log(n)\log\log(n)}\right).$$

We need to find as many such congruences as the number $\pi(B)$ of prime integers in the range $[1, B]$. So the total running time of the algorithm is

$$\frac{\pi(B)}{P} \sim \frac{B}{P} = \exp\left((A + \frac{1}{2A} + o(1))\sqrt{\log(n)\log\log(n)}\right).$$

The minimum of $A + \frac{1}{2A}$ is reached for $A = 1/\sqrt{2}$. We thus take

$$B = \exp\left(\frac{1}{\sqrt{2}}.\sqrt{\log(n)\log\log(n)}\right)$$

and find a time complexity

$$T = \exp\left((\sqrt{2} + o(1))\sqrt{\log(n)\log\log(n)}\right).$$

9. Complexity analysis of the quadratic sieve

We now study the complexity of the quadratic sieve. In this algorithm we consider integers $(m^2 - n) + a^2 + 2am$ of size $\sqrt{n}$ and we hope to find enough B-smooth ones. Assume

$$B = \exp(A.\sqrt{\log(n)\log\log(n)})$$

for some positive real constant A. Then the probability that an integer of size $\sqrt{n}$ is B-smooth is $P = \exp(-(1 + o(1))u \log u)$ with

$$u = (\log\sqrt{n})/\log B = \frac{1}{2A}\sqrt{\frac{\log(n)}{\log\log(n)}}.$$

So

$$P = \exp\left(-\frac{1+o(1)}{4A}\sqrt{\log(n)\log\log(n)}\right).$$

The time needed to find one congruence between a square and a B-smooth integer is thus

$$\frac{1}{P} = \exp\left(\frac{1+o(1)}{4A}\sqrt{\log(n)\log\log(n)}\right).$$

We need to find as many such congruences as the number $\pi(B)$ of prime integers in the range $[1, B]$. So the total running time of the algorithm is

$$\frac{\pi(B)}{P} \sim \frac{B}{P} = \exp\left((A + \frac{1}{4A} + o(1))\sqrt{\log(n)\log\log(n)}\right).$$

The minimum of $A + \frac{1}{4A}$ is reached for $A = 1/2$. We thus take

$$B = \exp\left(\frac{1}{2}.\sqrt{\log(n)\log\log(n)}\right)$$

and find a time complexity

$$T = \exp\left((1 + o(1))\sqrt{\log(n)\log\log(n)}\right).$$

10. EXERCISE : AN EXAMPLE OF QUADRATIC SIEVE

We want to factor the integer $n = 32399$ using the quadratic sieve.

1 . We notice that $\sqrt{n} \simeq 179.9$. Write a congruence modulo n of the type

$$(a+m)^2 \equiv a^2 + u_1 a + u_0 \bmod n$$

depending on an integer parameter a. Here m, u_0, u_1 are well chosen integer constants.

2 . Find values of a in the interval $[-40, 40]$ that produce a congruence between a square and a smooth number (in a sense to be made precise) modulo n. You may use the following data.

```
for(a=-40,40,print([a,factor(a^2+2*a*180+1)]))
[-40, [-1, 1; 12799, 1]]
[-39, [-1, 1; 2, 1; 11, 1; 569, 1]]
[-38, [-1, 1; 5, 1; 2447, 1]]
[-37, [-1, 1; 2, 1; 5, 2; 239, 1]]
[-36, [-1, 1; 107, 1; 109, 1]]
[-35, [-1, 1; 2, 1; 11, 2; 47, 1]]
[-34, [-1, 1; 11083, 1]]
[-33, [-1, 1; 2, 1; 5, 1; 13, 1; 83, 1]]
[-32, [-1, 1; 5, 1; 2099, 1]]
[-31, [-1, 1; 2, 1; 5099, 1]]
[-30, [-1, 1; 19, 1; 521, 1]]
[-29, [-1, 1; 2, 1; 4799, 1]]
[-28, [-1, 1; 5, 1; 11, 1; 13, 2]]
[-27, [-1, 1; 2, 1; 5, 1; 29, 1; 31, 1]]
[-26, [-1, 1; 19, 1; 457, 1]]
[-25, [-1, 1; 2, 1; 53, 1; 79, 1]]
[-24, [-1, 1; 11, 1; 733, 1]]
[-23, [-1, 1; 2, 1; 5, 3; 31, 1]]
[-22, [-1, 1; 5, 1; 1487, 1]]
[-21, [-1, 1; 2, 1; 3559, 1]]
[-20, [-1, 1; 13, 1; 523, 1]]
[-19, [-1, 1; 2, 1; 41, 1; 79, 1]]
[-18, [-1, 1; 5, 1; 1231, 1]]
[-17, [-1, 1; 2, 1; 5, 1; 11, 1; 53, 1]]
[-16, [-1, 1; 5503, 1]]
[-15, [-1, 1; 2, 1; 13, 1; 199, 1]]
[-14, [-1, 1; 29, 1; 167, 1]]
[-13, [-1, 1; 2, 1; 5, 1; 11, 1; 41, 1]]
[-12, [-1, 1; 5, 2; 167, 1]]
[-11, [-1, 1; 2, 1; 19, 1; 101, 1]]
[-10, [-1, 1; 3499, 1]]
[-9, [-1, 1; 2, 1; 1579, 1]]
[-8, [-1, 1; 5, 1; 563, 1]]
```

```
[-7, [-1, 1; 2, 1; 5, 1; 13, 1; 19, 1]]
[-6, [-1, 1; 11, 1; 193, 1]]
[-5, [-1, 1; 2, 1; 887, 1]]
[-4, [-1, 1; 1423, 1]]
[-3, [-1, 1; 2, 1; 5, 1; 107, 1]]
[-2, [-1, 1; 5, 1; 11, 1; 13, 1]]
[-1, [-1, 1; 2, 1; 179, 1]]
[0, matrix(0,2)]
[1, [2, 1; 181, 1]]
[2, [5, 2; 29, 1]]
[3, [2, 1; 5, 1; 109, 1]]
[4, [31, 1; 47, 1]]
[5, [2, 1; 11, 1; 83, 1]]
[6, Mat([13, 3])]
[7, [2, 1; 5, 1; 257, 1]]
[8, [5, 1; 19, 1; 31, 1]]
[9, [2, 1; 11, 1; 151, 1]]
[10, Mat([3701, 1])]
[11, [2, 1; 13, 1; 157, 1]]
[12, [5, 1; 19, 1; 47, 1]]
[13, [2, 1; 5, 2; 97, 1]]
[14, Mat([5237, 1])]
[15, [2, 1; 29, 1; 97, 1]]
[16, [11, 1; 547, 1]]
[17, [2, 1; 5, 1; 641, 1]]
[18, [5, 1; 1361, 1]]
[19, [2, 1; 13, 1; 277, 1]]
[20, [11, 1; 691, 1]]
[21, [2, 1; 4001, 1]]
[22, [5, 1; 41, 2]]
[23, [2, 1; 5, 1; 881, 1]]
[24, [13, 1; 709, 1]]
[25, [2, 1; 4813, 1]]
[26, Mat([10037, 1])]
[27, [2, 1; 5, 2; 11, 1; 19, 1]]
[28, [5, 1; 41, 1; 53, 1]]
[29, [2, 1; 5641, 1]]
[30, Mat([11701, 1])]
[31, [2, 1; 11, 1; 19, 1; 29, 1]]
[32, [5, 1; 13, 1; 193, 1]]
[33, [2, 1; 5, 1; 1297, 1]]
[34, Mat([13397, 1])]
[35, [2, 1; 31, 1; 223, 1]]
```

```
[36, [53, 1; 269, 1]]
[37, [2, 1; 5, 1; 13, 1; 113, 1]]
[38, [5, 3; 11, 2]]
[39, [2, 1; 31, 1; 251, 1]]
[40, Mat([16001, 1])]
```

3 . Write down all the congruences you have found. Report the signs and valuations in a matrix M with integer coefficients.

4 . Compute (a basis of) the kernel of the reduction modulo 2 of the matrix M.

5 . For each vector in this basis write a congruence between two squares modulo n. Deduce a factorization of n.

11. EXERCISE : AN EXAMPLE OF QUADRATIC SIEVE

We want to factor $n = 7747$ using the quadratic sieve.

1 . Note that $\sqrt{n} \simeq 88.01704$. Write a congruence modulo n of the type

$$(a+m)^2 \equiv a^2 + u_1 a + u_0 \bmod n$$

depending on an integer parameter a. Here m, u_0, u_1 are well chosen integer constants.

2 . Find values of a between -3 and 6 that provide a congruence between a square and a 29-smooth integer n.

3 . Write down all the interesting congruences thus obtained. Collect signs and valuations in a matrix M with integer coefficients.

4 . Compute the kernel of the reduction modulo 2 of the matrix M. Give a basis of this kernel.

5 . For each vector in this basis find a congruence between two squares modulo n. Deduce a (possibly trivial) factorization of n.

12. EXERCISE : ANOTHER EXAMPLE OF QUADRATIC SIEVE

We want to factor the integer $N = 20737$ using the quadratic sieve.

1 . We notice that $\sqrt{N} \simeq 144.0035$. We set $m = 144$. Write a congruence modulo N of the type

$$(a+m)^2 \equiv a^2 + u_1 a + u_0 \bmod N$$

depending on an integer parameter a. Here u_0, u_1 are well chosen integer constants.

2 . Find values of a in the interval $[-20, 20]$ that produce a congruence between a square and a smooth number (in a sense to be made precise) modulo N. You may use the following data.

```
? for(a=-20,20,print([a,factor(a^2+a*288-1)]))
[-20, [-1, 1; 3, 1; 1787, 1]]
[-19, [-1, 1; 2, 3; 3, 2; 71, 1]]
[-18, [-1, 1; 4861, 1]]
[-17, [-1, 1; 2, 9; 3, 2]]
[-16, [-1, 1; 3, 1; 1451, 1]]
```

```
[-15, [-1, 1; 2, 12]]
[-14, [-1, 1; 3, 1; 1279, 1]]
[-13, [-1, 1; 2, 3; 3, 1; 149, 1]]
[-12, [-1, 1; 3313, 1]]
[-11, [-1, 1; 2, 3; 3, 1; 127, 1]]
[-10, [-1, 1; 3, 3; 103, 1]]
[-9, [-1, 1; 2, 4; 157, 1]]
[-8, [-1, 1; 3, 3; 83, 1]]
[-7, [-1, 1; 2, 4; 3, 1; 41, 1]]
[-6, [-1, 1; 1693, 1]]
[-5, [-1, 1; 2, 3; 3, 1; 59, 1]]
[-4, [-1, 1; 3, 1; 379, 1]]
[-3, [-1, 1; 2, 3; 107, 1]]
[-2, [-1, 1; 3, 1; 191, 1]]
[-1, [-1, 1; 2, 5; 3, 2]]
[0, Mat([-1, 1])]
[1, [2, 5; 3, 2]]
[2, [3, 1; 193, 1]]
[3, [2, 3; 109, 1]]
[4, [3, 1; 389, 1]]
[5, [2, 3; 3, 1; 61, 1]]
[6, [41, 1; 43, 1]]
[7, [2, 4; 3, 1; 43, 1]]
[8, [3, 2; 263, 1]]
[9, [2, 4; 167, 1]]
[10, [3, 2; 331, 1]]
[11, [2, 3; 3, 1; 137, 1]]
[12, [59, 1; 61, 1]]
[13, [2, 3; 3, 1; 163, 1]]
[14, [3, 1; 1409, 1]]
[15, [2, 6; 71, 1]]
[16, [3, 1; 1621, 1]]
[17, [2, 6; 3, 4]]
[18, Mat([5507, 1])]
[19, [2, 3; 3, 6]]
[20, [3, 1; 2053, 1]]
```

3 . Write down all the congruences you have found. Report the signs and valuations in a matrix M with integer coefficients.

4 . Compute (a basis of) the kernel of the reduction modulo 2 of the matrix M.

5 . For each vector in this basis write a congruence between two squares modulo N. Deduce a factorization of N.

REFERENCES

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.

[2] Andrew Granville. Smooth numbers: computational number theory and beyond. In *Algorithmic number theory: lattices, number fields, curves and cryptography*, volume 44 of *Math. Sci. Res. Inst. Publ.*, pages 267–323. Cambridge Univ. Press, Cambridge, 2008.

[3] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman J. J. te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 333–350, 2010.

[4] Gary L. Miller. Riemann's hypothesis and tests for primality. *J. Comput. System Sci.*, 13(3):300–317, 1976. Working papers presented at the ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N.M., 1975).

[5] Michael O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1):128–138, 1980.

[6] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[7] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332, 1999.

[8] Gérald Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 163 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, third edition, 2015. Translated from the 2008 French edition by Patrick D. F. Ion.