Factoring of Large Integers

Prof. Ashok K Bhateja

IIT Delhi

Factoring large Integers

- The security of RSA public-key, Rabin public-key cryptographic techniques depends upon the intractability of the integer factorization problem
- Some factoring techniques
  - Trial Division
  - Pollard's $p - 1$
  - Pollard's rho
  - Quadratic Sieve
  - Number Field Sieve

Trial division

- For each integer d greater than 1 and not larger than $\sqrt{n}$, check if d divides n.
- It takes roughly $\sqrt{n}$ divisions in the worst case when n is a product of two primes of the same size.
- Good if n is less than 240
- The density of primes is $n/\ln n$, and since primes go up to $\sqrt{n}$ the complexity is $O(\sqrt{n}/\ln\sqrt{n})$.

Pollard's p -1 factoring

- Due to J. M. Pollard 1974.
- B-smooth: Let B be a positive integer. An integer n is B-smooth, or smooth with respect to a bound B, if all its prime factors are $\leq B$.

  Ex. 153 is 17-smooth, because $153 = 3^2 \times 17$

  $15750 = 2 \times 3^2 \times 5^3 \times 7$ is 7-smooth

  $702 = 2 \times 3^3 \times 13$ is not 7-smooth.

- Pollard's $p - 1$ algorithm with bound B will succeed to find a factor of n if n has a factor p such that $p - 1$ is B-smooth.

Pollard's p -1 factoring: Concept

- The idea is that if $p \mid n$, and p is prime, then $a^{p-1} \equiv 1 \bmod p$, or $x = a^{p-1} - 1 \equiv 0 \bmod p$, for any a relatively prime to p, so that $\gcd(x, n) = p$, the factor of n.
- How can we compute x without knowing p ?
- Compute a^m with an exponent $m = 1, 2, 3, \ldots$ until $\gcd(a^m - 1, n) = p$, i.e., until $m = p - 1$.
- It requires to execute $p - 1$ operations, each involving exponentiation to a big power.
- Better way to choose m:
  - if m is such that $p - 1 \mid m$, i.e., $m = k(p - 1)$, then

$$a^m - 1 = a^{k(p-1)} - 1 = (a^{p-1})^k - 1 \equiv 0 \bmod p$$

so that $\gcd(a^m - 1, n) = p$.

Pollard's p -1 factoring

- Let B be a smoothness bound.
- If $q^{l} \leq B$, then $l \ln q \leq \ln B$ i.e., $l \leq \lfloor \ln B / \ln q \rfloor$

$$\text{Compute } m = \prod_{\substack{q \text{ is prime} \\ q \leq B}} q^{\left\lfloor \frac{\ln B}{\ln q} \right\rfloor}$$

- If p is a prime factor of n such that $p - 1$ is B-smooth, then p - 1 divides m
- For any a satisfying gcd $(a, p) = 1$, $a^{p-1} \equiv 1 \pmod{p}$.

  $\therefore a^{m} \equiv 1 \pmod{p} \Rightarrow p \mid \gcd(a^{m} - 1, n)$.
- gcd $(a^{m} - 1, n)$ is a non-trivial factor of n

Pollard's p-1 Algorithm

Choose a smoothness bound B, usually about $10^5 - 10^6$

Select a random integer a, $2 < a < n - 1$, and compute $d = \gcd(a, n)$. If $d > 2$ then return(d).

for each prime $q \leq B$

compute $l = \lfloor \ln B / \ln q \rfloor$

compute $a \equiv a^{q^l} \bmod n$

Compute $d = \gcd(a - 1, n)$.

If $d = 1$, go to 1 and increase B

If $d = n$, go to 3 and change a

If $d \neq 1$ & $d \neq n$, return d (It is a non-trivial factor of n)

Example: Pollard's p –1 method

To factor the number $n = 299$.

Select $B = 5$.

Select $a = 2$

$m = 2^2 \times 3^1 \times 5^1$

$q = 2, l = 2, \; q^l = 4, \; a = 2^4 \mod 299 = 16$

$q = 3, l = 1, \; q^l = 3, \; a = 16^3 \mod 299 = 209$

$q = 5, l = 1, \; q^l = 5, \; a = 209^5 \mod 299 = 170$

$g = \gcd(a - 1, n) = \gcd(169, 299) = 13$

Since $1 < 13 < 299$, thus return 13

Another factor of 299 is $299 / 13 = 23$ is prime

Thus, it is fully factored: $299 = 13 \times 23$

Pollard's p –1 method

- Conditions of success of the algorithm
  - Pollard's p −1 algorithm with bound B will succeed to find a factor of n if n has a factor p such that $p - 1$ is B-smooth
  - If this B is too big, then Pollard's $p - 1$ algorithm will not be faster than trial division.
- Time Complexity
  - The running time of this algorithm is $O(B \times \log B \times \log^2 n)$; larger values of B make it run slower but are more likely to produce a factor.
- Selection of B
  - B must be as large as the largest prime factor of p - 1.
  - Let p is the smallest prime factor of n, assume that $p - 1$, be a random number of size less than $\sqrt{n}$. By Dixon's theorem, the probability that the largest factor of such a number is less than $(p-1)^{1/k}$ is roughly k^{-k}.
    So, the probability that $B \approx n^{1/6}$ will yield factorization is $\approx 3^{-3} = 1/27$.

Pollard's rho factoring algorithm

- Pollard's rho algorithm is a special-purpose factoring algorithm for finding small factors of a composite integer.
- It was invented by John Pollard in 1975.
- It uses only a small amount of space, and its expected running time is proportional to the square root of the size of the smallest prime factor of the composite number being factorized.

Pollard's rho algorithm

- Let p be a prime factor of a composite integer n.
- Consider the following experiment:

  Pick some numbers $x_0, x_1, x_2, \ldots, x_l$ uniformly at random in Z_n.

  Assume that all the numbers are distinct .

  Suppose that there exists some $0 \leq i < j \leq l$ such that

  $$x_i \equiv x_j \pmod{p}.$$

  Then $p \mid (x_i - x_j)$, and since $p \mid n$ also $\therefore\ p \mid \gcd(x_i - x_j, n)$.
- Since $-n < x_i - x_j < n$ and $x_i \neq x_j$, $\gcd(x_i - x_j, n) < n$.
  Thus $\gcd(x_i - x_j, n)$ provides a nontrivial factor of n.

Probability of collision

Let S be a set of N elements, after l distinct elements have been selected then the probability that the next element selected is also distinct from the previous ones is $(1 - l/\mathrm{N})$. Hence

$$P(\text{all diffrent}) = \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\ldots\left(1 - \frac{l-1}{n}\right)$$

$$\leq e^{-1/n} \cdot e^{-2/n} \cdot e^{-(l-1)/n}$$

$$= e^{-l(l-1)/2n} \approx e^{-l^2/(2n)}$$

So, if $l = \sqrt{n}$, the probability is roughly $e^{-1/2} \leq 2/3$, and so there is a substantial probability that there exists some $0 \leq i < j \leq l$ such that $x_i \equiv x_j \pmod{p}$.

Pollard's rho factoring: requirements

- Generation of $x_0 , x_1 , x_2 , \ldots , x_l$ uniformly at random in Z_n
- Finding x_i and x_j ; $0 \leq i < j \leq l$ such that $x_i \equiv x_j \pmod{p}$.

  Since p divides n but is unknown, this can be done by computing the terms $x_i \bmod n$, instead of $x_i \bmod p$ and searching x_i and x_j s.t., $\gcd(x_i - x_j , n) > 1$.
- If $\gcd(x_i - x_j , n) > 1$, then a non-trivial factor of n is $\gcd(x_i - x_j , n)$.
- Pollard's idea was to choose $x_0, x_1, x_2, \ldots , x_l$ in a recursive manner, choosing $x_0 \in Z_n$ at random and then computing $x_m = f(x_{m-1}) \bmod n$ for some appropriate function f.

Pollard's rho factoring

- Generation of random numbers: The clever trick here is not to pick the x_i's randomly, but instead in a way that "looks" random.
- Let $f : Z_n \rightarrow Z_n$ be defined by $f(x) = (x^2 + 1) \bmod n$. The sequence

  x_0 , $x_1 = f(x_0)$, $x_2 = f(x_1)$, … , $x_i = f(x_{i-1})$, ... looks random
- Finding x_i and x_j ; $0 \leq i < j \leq l$ s.t., $x_i \equiv x_j \pmod{n}$ can be done by Floyd's cycle detection algorithm.

Floyd's Cycle Detection

- Finding a cycle in a sequence of iterated function values.
- For any function f that maps a finite set S to itself, and any initial value x_0 in S, the sequence of iterated function values

  $$x_0,\ x_1 = f(x_0),\ x_2 = f(x_1),\ \ldots,\ x_i = f(x_{i-1})$$

  there must be some pair of distinct indices i and j such that $x_i \equiv x_j$.

Robert W. Floyd's cycle detection Algorithm

- Floyd’s Cycle-Finding Algorithm is like the Tortoise Hare Story.
- Floyd’s Cycle-Finding Algorithm uses two pointers that move at different speeds. If there is a cycle, both pointers would point to the same value at some point in the future.
- The Tortoise and the Hare are both pointers, and both start at the top of the list. For each iteration, the Tortoise takes one step, and the Hare takes two. If there is a loop, the hare will go around that loop (possibly more than once) and eventually meet up with the turtle. If there is no loop, the hare will get to the end of the list without meeting up with the turtle.
- This algorithm takes O(N) time.

Pollard's rho factoring

- Let n, the integer to be factored
- Consider polynomial $f(x) = x^2 + 1 \pmod{n}$ and find x_m and x_{2m} s.t., $\gcd(x_m - x_{2m}, n) \neq 1$
- If $\gcd(x_m - x_{2m}, n) > 1$, then a non-trivial factor of n is $\gcd(x_m - x_{2m}, n)$.
- The situation $\gcd(x_m - x_{2m}, n) = n$ occurs with negligible probability.

Pollard's rho algorithm for factoring integers

INPUT: a composite integer n that is not a prime power.

Set $a \leftarrow 2$, $b \leftarrow 2$, $d \leftarrow 1$

while $d = 1$

$a \leftarrow f(a) \bmod n$

$b \leftarrow f(f(b)) \bmod n$

$d \leftarrow \gcd(|a - b|, n)$

If $d = n$ then terminate the algorithm with failure

else return(d) and terminate with success

Note: (options upon termination with failure) try again using a starting value other than 2 or with a different polynomial instead of $f(x) = x^2 + 1$. i.e. $f(x) = x^2 + c$, $c \neq 1$.

Pollard's rho algorithm for factoring

- The expected time for Pollard's rho algorithm proportional to the square root of the smallest prime factor p of n i.e., $O(p^{1/2})$ modular multiplications. i.e., the expected time to find a non-trivial factor of n is $O(n^{1/4})$ modular multiplications.
- Example: Let $n = 8051$ and $f(x) = x^2 + 1 \bmod 8051$.

a	b	$\lvert a - b \rvert$	$\gcd(\lvert a - b \rvert, 8051)$
2	2	0	1
5	26	21	1
26	7474	7450	1
677	871	194	97

97 is a non-trivial factor of 8051.

Fermat's factorization method

- Let n be an odd integer as the difference of two squares:

$$n = a^2 - b^2$$

then $n = (a - b)\ (a + b)$ if neither factor equals one, it is a proper factorization of n.

- Each odd number (not prime) has such a representation.

If $n = cd$ is a factorization of n, then

$$n = \left(\frac{c+d}{2}\right)^2 - \left(\frac{c-d}{2}\right)^2$$

Since n is odd, then c and d are also odd, so those halves are integers.

Dixon's factorization method

- It is a general-purpose integer factorization algorithm.
- It was designed by John D. Dixon, a mathematician and published in 1981.
- (Basic idea) find a congruence of squares modulo the integer n, i.e., select random or pseudorandom x values, s.t. $x^2 \bmod n$ is a perfect square.

  i.e., If $x^2 \bmod n \equiv y^2$, then gcd $(x - y, n)$ is a factor of n.
- Approach: start with $x = \lceil \sqrt{n} \rceil$ and counting up so that $x^2 \bmod n$ is a perfect square.

Theorem: Let n be a positive integer. Suppose there exist integers x, y such that $x^2 \equiv y^2 \pmod{n}$ but $x \neq \pm y \pmod{n}$. Then $\gcd(x - y, n)$ gives a non-trivial factor of n.

Proof: Let $x^2 \equiv y^2 \pmod{n}$

$n \mid (x - y)(x + y)$

Since $x \neq \pm y \pmod{n}$

implies neither $n \mid (x - y)$ nor $n \mid (x + y)$

Therefore $\gcd(x - y, n)$ is a non-trivial factor of n.

Practicality of Dixon's factorization method

- Selecting random x values will take long time to find a congruence of squares, since there are $\sqrt{n}$ squares less than n.
- Better approach: To factorize n,
  - choose an integer B (bound) and a set

    $P = \{ p : \text{prime} \leq B\}$, called factor base
  - Search for integer z s.t. $z^2 \bmod n$ is B-smooth i.e.

$$z^2 \bmod n = \prod_{p_i \in P} p_i^{e_i} (\bmod n)$$

  - Generate enough such relations so that the exponents of the primes on RHS are all even i.e.

$$z_1^2 \cdot z_2^2 \cdots z_k^2 \equiv \prod_{p_i \in P} p_i^{e_{i,1} + e_{i,2} + \cdots e_{i,k}}$$

where $e_{i,1} + e_{i,2} + \ldots + e_{i,k}$ is even. This gives $x^2 \bmod n \equiv y^2$

Algorithm: Dixon's factorization

choose bound B

repeat

for $i = 1$ to $k + 1$

choose z_i between 1 and n s.t. $z_i^2 \bmod n$ is B-smooth

$$i.e. \quad z_i^2 \; mod \; n \equiv \prod_j p_j^{e_{ij}}$$

find nonempty set S such that $\sum_{z_i \in S} e_{ij}$ is even

$$x = \prod_{z_i \in S} z_i \bmod n, \qquad y = \prod_{p_j \in P} p_j^{\left(\sum_{z_i \in S} e_{ij}\right)/2} \bmod n$$

while $x \equiv \pm y \bmod n$

return gcd $(x + y, n)$

Example: Dixon's factorization

n = 23449 and factor base P = {2, 3, 5, 7}

$[\sqrt{n}\,]$ = 154. Starting here for z_i

The first related squares are:

$970^2 \bmod (23449) = 2940 = 2^2 \times 3 \times 5 \times 7^2$

$8621^2 \bmod (23449) = 11760 = 2^4 \times 3 \times 5 \times 7^2$

So, $(970 \times 8621)^2 \equiv (2^3 \times 3 \times 5 \times 7^2)^2 \pmod{23449}$

i.e., $14526^2 \equiv 5880^2 \pmod{23449}$

Now, find gcd(14526 -5880, 23449) = 131

gcd (14526 + 5880, 23449) = 179

Factors are $n = 131 \times 179$

Quadratic sieve factoring

- Invented by Carl Pomerance in 1981
- Fastest algorithm for factoring a numbers up to 110 digits long.
- An improvement to Dixon's factorization method.
- Basic idea
  - Tries to set up a congruence of squares modulo *n* to find a factor.

Sieve of Eratosthenes for Prime Number

1. Start with all numbers greater than 1
2. Divide all by the first number & delete the number which is divisible.
3. Repeat until no numbers are left to divide by
4. What remains are the prime numbers.

Prime Number Sieve

Initial Sieve Space

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46

47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67

68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88

89 90 91 92 93 94 95 96 97 98 99 100 101

Prime Number Sieve

After Divide by two

2 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 | 25 | 27 | 29 | 31 | 33

| 35 | 37 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 |

63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91

| 93 | 95 | 97 | 99 | 101

Prime Number Sieve

After Divide by three

2 | 3 | 5 | 7 | | | 11 | 13 | | | 17 | 19 | | | 23 | 25 | | | 29 | 31 | | | 35 | 37 | |

| 41 | 43 | | | 47 | 49 | | | 53 | 55 | | | 59 | 61 | | | 65 | 67 | | | 71 | 73 | |

| 77 | 79 | | | 83 | 85 | | | 89 | 91 | | | 95 | 97 | | | 101

Prime Number Sieve

After Divide by five

2 | 3 | 5 | 7 | | | 11 | 13 | | | 17 | 19 | | | 23 | | | | | 29 | 31 | | | | | 37 | | | 41

| 43 | | | 47 | 49 | | | 53 | | | | | 59 | 61 | | | | | 67 | | | 71 | 73 | | | 77 | 79

| | | 83 | | | | | 89 | 91 | | | | | 97 | | | 101

Prime Number Sieve

After all possible divisions

2 | 3 | 5 | 7 | | | 11 | 13 | | | 17 | 19 | | | 23 | | | | | 29 | 31 | | | | | 37 | | |

41 | 43 | | | 47 | | | | | 53 | | | | | 59 | 61 | | | | | 67 | | | 71 | 73 | | | | |

79 | | | 83 | | | | | 89 | | | | | | | 97 | | | 101

Quadratic sieve factoring

- Fact: Let x and y be integers. If $x^2 \equiv y^2 \bmod n$ but $x \neq \pm y \bmod n$, then gcd $(x - y, n)$ is a nontrivial factor of n.
- Fact: Let n be an odd composite integer that is divisible by k distinct odd primes. If $a \in Z_n^*$, then the congruence $x^2 \equiv a^2 \pmod{n}$ has exactly 2^k solutions modulo n, two of which are $x = a$ and $x = -a$.
- If x and y are two randomly selected integers, s.t. $x^2 \equiv y^2 \bmod n$, then with probability at least 1/2 it is the case that $x \neq \pm y \bmod n$.

Finding x & y at random, satisfying $x^2 \equiv y^2 \bmod n$

- (Factor Base) A set consisting of the first t primes $S = \{p_1, p_2, \ldots, p_t\}$ is chosen.
- Find pairs of integers (a_i, b_i) satisfying the following

(i) $a_i^2 \equiv b_i \pmod{n}$

(ii) $b_i = \prod_{j=1}^{t} p_j^{e_{ij}}$, $e_{ij} \geq 0$, i.e. b_i is p_t smooth

- Find a subset of the b_i's whose product is a perfect square.

Quadratic sieve factoring

- Let integer n is to be factorized
- Let $m = \lfloor \sqrt{n} \rfloor$, define polynomial $Q(x) = (x + m)^2 - n$

  $Q(x)$ is small, if x is small in absolute value.

  $x = \pm 1, \pm 2, \ldots$
- If $a_i = (x + m)$ & $b_i = (x + m)^2 - n$, then $a_i^2 = (x + m)^2 \equiv b_i \pmod{n}$
- Therefore, aim is to selects $a_i = (x + m)$ and tests whether

  $b_i = (x + m)^2 - n$ is p_t smooth.

Factor base

- A prime p divides b_i then $(x + m)^2 \equiv n \pmod{p}$ i.e., n is a quadratic residue modulo p.
- Thus, the factor base need only contain those primes p for which the Legendre symbol (n/p) is 1.
- Since b_i may be negative, - 1 should be included in the factor base.
- Example: For $n = 24961$, $t = 6$

  Factor base = {-1, 2, 3, 5, 13, 23}

Checking smoothness of b_i

- For each i, associate the binary vector $v_i = (v_{i1}, v_{i2}, \ldots, v_{it})$ with the integer exponent vector $(e_{i1}, e_{i2}, \ldots, e_{it})$ such that $v_{ij} = e_{ij} \bmod 2$.
- If $t + 1$ pairs (a_i, b_i) are obtained, then the t-dim vectors $v_1, v_2, \ldots, v_{t+1}$ must be linearly dependent over Z_2.

  i.e., there must exist a non−empty subset T □□ { 1, 2, . . . , t + 1) s.t.,

$$\sum_{i \in T} v_i = 0 \quad \text{and hence} \quad \prod_{i \in T} b_i \text{ is a perfect square.}$$

Sieving

- Instead of checking whether $Q(x)$ is divisible only by the primes in FB, find which values of $Q(x)$ inside the sieving interval p divides.
- $Q(x) = (x + m)^2 - n$, find the roots of modular polynomial

$$Q(x) = (x + m)^2 - n \equiv 0 \pmod{p}, \qquad \text{where } m = \lfloor \sqrt{n} \rfloor$$

- This equation usually leads to two integer solutions in Z_p, say x_p and $\hat{x}_p = p - x_p$.
- Knowing that $p \mid Q(x_p)$ and $p \mid Q(\hat{x}_p)$, it is very easy to find all the other values of $Q(x)$ divisible by p inside the sieving interval.

Sieving

- If p is an odd prime in the factor base and p divides $Q(x)$, then p also divides $Q(x + kp)$ for every integer k.

  $Q(x) = x^2 - n$

  $Q(x + kp) = (x + kp)^2 - n$

  $Q(x + kp) = x^2 + 2xkp + (kp)^2 - n$

  $Q(x + kp) = Q(x) + 2xkp + (kp)^2 \equiv Q(x) \pmod{p}$

- Thus, solving $Q(x) \equiv 0 \pmod{p}$ for x generates a whole sequence of $y = Q(x)$s which are divisible by p.
- x is a square root of n modulo a prime, i.e. $x^2 \equiv n \pmod{p}$

Example - Sieving

- $n = 15347$, $m = \lceil \sqrt{15347} \rceil = 124$, $S = \{-1, 2, 17, 23, 29\}$
- Polynomial $Q(x) = (x + 124)^2 - 15347$.
- Choose $0 \leq x < 100$ to sieve $Q(x)$:

$$Q = [Q(0) \quad Q(1) \quad Q(2) \quad Q(3) \quad Q(4) \quad Q(5) \quad \ldots \quad Q(99)]$$

$$= [\ 29 \quad 278 \quad 529 \quad 782 \quad 1037 \quad 1294 \quad \ldots \quad 34382]$$

- Sieving:
  - For $p = 2$, $(x + 124)^2 - 15347 \equiv 0 \pmod 2 \Rightarrow x \equiv 1 \pmod 2$.

    Starting at $x = 1$ and incrementing by 2, each entry will be divisible by 2. Divide them by 2

    $Q = [29 \quad 139 \quad 529 \quad 391 \quad 1037 \quad 647 \quad \ldots \quad 17191]$
  - For $p = 17$, $x = 4 \bmod 17$; $p = 23$, $x = 3 \bmod 23$; $p = 29$, $x = 13 \bmod 29$

    $Q = [1 \quad 139 \quad 23 \quad 1 \quad 61 \quad 647 \quad \ldots \quad 1 \quad \ldots \quad 17191]$
  - Q_0, Q_3, Q_{71} are p_t smooth.

Parallel Quadratic Sieve

- In the Quadratic Sieve the most laborious part of the algorithm is the sieving over the given interval.
- Divide the sieving interval into blocks, with the number of blocks corresponding to the number of processors available.
- The benefit of performing the parallel process in this manner is that it requires minimum communication between the processors,

Finding x & y

- $\prod_{i \in T} a_i^2$ is a perfect square.
- Thus, $x = \prod_{i \in T} a_i$ and y square root of $\prod_{i \epsilon T} b_i$

  satisfies $x^2 \equiv y^2 \bmod n$.

- If this pair also satisfies $x \neq \pm y \bmod n$, then gcd $(x - y, n)$ is a nontrivial factor of n.

  Otherwise, some of the (a_i, b_i) pairs may be replaced by some new such pairs.

- In practice, there is high probability that at least one (x, y) pair satisfying $x \neq \pm y \bmod n$.

Theorem: The probability that a number x, is B-smooth is approximately u^{-u}, where $u = \frac{\ln x}{\ln B}$.

Kechlibar, Marian, "The quadratic sieve-introduction to theory with regard to implementation issues." *Charles University in Prague* (2005).

Example: Let $n = 24961$

Select the factor base $S = \{-1, 2, 3, 5, 13, 23\}$ of size $t = 6$.

7, 11, 17 and 19 are omitted from S since $(n/p) = -1$ for these primes.

Compute $m = \lfloor \sqrt{24961} \rfloor = 157$.

collected for the first $t + 1$ values of x for which $Q(x)$ is 23-smooth.

i	x	$Q(x)$	Factors of $Q(x)$	a_i	v_i
1	0	-312	$-2^3 \cdot 3 \cdot 13$	157	(1, 1, 1, 0, 1, 0)
2	1	3	3	158	(0, 0, 1, 0, 0, 0)
3	-1	-625	-5^4	156	(1, 0, 0, 0, 0, 0)
4	2	320	$2^6 \cdot 5$	159	(0, 0, 0, 1, 0, 0)
5	-2	-936	$-2^3 \cdot 3^2 \cdot 13$	155	(1, 1, 0, 0, 1, 0)
6	4	960	$2^6 \cdot 3 \cdot 5$	161	(0, 0, 1, 1, 0, 0)
7	-6	-2160	$-2^4 \cdot 3^3 \cdot 5$	151	(1, 0, 1, 1, 0, 0)

- Linearly dependence: $v_1 + v_2 + v_5 = 0$
- Compute $x = (a_1 a_2 a_5 \bmod n) = 936$.
- $y^2 = (-1)^2 \cdot 2^6 \cdot 3^4 \cdot 5^0 \cdot 13^2 \cdot 23^0$
- $y = (-1) \cdot 2^3 \cdot 3^2 \cdot 13 \bmod n = -936 \bmod 24961 = 24025$.
- Since $936 \equiv -24025 \pmod n$, consider another linear dependency.
- By inspection, $v_3 + v_6 + v_7 = 0$
- Compute $x = (a_3 a_6 a_7 \bmod n) = 23405$.
- Compute $y = (-2^5 \cdot 3^2 \cdot 5^3) \bmod n = 13922$.
- Since $23405 \neq \pm 13922 \pmod n$, compute $\gcd(x - y, n)$

  $\gcd(9483, 24961) = 109$.
- Hence, two non-trivial factors of 24961 are 109 and 229.

Speedup Sieving – avoid division

- Using Divisions Primes $p_1, \ldots, p_t < B$ divide $Q(x)$.
- Divide $Q(x)$ by all the p_i . Also, p_i^2, p_i^3, etc. until does not work.
- Use Subtraction Primes $p_1, \ldots, p_t < B$ divide $Q(x)$.

$$d = \lg(Q(x)) - \lg(p_1) - \lg(p_2) - \cdots - \lg(p_m)$$

If $d \sim 0$ implies $Q(x)$ is B-smooth.

Size of factor base

- The optimal selection of $t \approx L_n$ [1/2, 1/2] derived from knowledge concerning the distribution of smooth integers close to $\sqrt{n}$.

  Where $L_n [\alpha, c] = O(\exp((c (\ln n)^{\alpha} (\ln\ln n)^{1 - \alpha}))$

  where c is a +ve constant, and α is a constant satisfying $0 < \alpha < 1$.

- With this choice, the expected running time of Quadratic Sieve algorithm is L_n [1/2, 1], independent of the size of the factors of n.

Multiple polynomial quadratic sieve

- The rate at which we find *B*-smooth relations usually starts high but decreases rapidly over time.
- This happens because $Q(x) = (x + m)^2 - n$ and we start sieving with $x = 0, 1, 2, \ldots$, thus the values of $Q(x)$ keep increasing with increasing x.
- Smaller numbers are more likely to be *B*-smooth than bigger numbers for the same *B*. So, longer the sieving.
- Remedy: use more than one polynomial to generate *B*-smooth relations.

Multiple polynomial quadratic sieve

- Start the sieving process with some polynomial $Q_1(x)$ over fixed sieving interval, collect B-smooth relations, and when the values of $Q_1(x)$ get too high and the probability of finding B-smooth relations decreases too much, instead of moving to the next interval, switch to a slightly different polynomial $Q_2(x)$.
- Continue sieving and switching polynomials until enough B-smooth relations are obtained.

Number Field Sieve

- Number field sieve is an improvement to the quadratic sieve.
- To factor a large number n, it is necessary to search for smooth numbers (i.e., numbers with small prime factors) of order $n^{1/2}$. The size of these values is exponential in the size of n in QS.
- The general number field sieve searches smooth numbers that are subexponential in the size of n. Since these numbers are smaller, they are more likely to be smooth than the numbers inspected in previous algorithms. This is the key to the efficiency of the number field sieve.
- Time Complexity: $L_n[1/3, \sqrt[3]{64/9}]$

$$= O(\exp((\sqrt[3]{64/9}\ (\ln n)^{1/3}\ (\ln \ln n)^{2/3}))$$

Quadratic Sieve: Reference

- Pomerance, Carl. "The quadratic sieve factoring algorithm." Workshop on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1984.
- Landquist, Eric. "The quadratic sieve factoring algorithm." *Math* 488 (2001): 1-11.
- Lenstra, Arjen K., et al. "The number field sieve." *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990.
- Lenstra, Arjen K., and Hendrik W. Lenstra. *The development of the number field sieve*. Vol. 1554. Springer Science & Business Media, 1993.