Factoring Integers

Primes

- Natural (counting) numbers: $\mathcal{N} = \{1, 2, 3, \ldots\}$
- A number $p \in \mathcal{N}$ is called prime if it is divisible only by 1 and itself
- 1 is not considered prime
- 2 is the only even prime
- Primes: 2, 3, 5, 7, 11, 13, ...
- There are infinitely many primes
- Every natural number n is factored into prime powers uniquely:

$$n = p_1^{k_1} \cdot p_2^{k_2} \cdots p_m^{k_m}$$

For example: $1960 = 2^3 \cdot 5^1 \cdot 7^2$

Primes

- The number of primes less than or equal to n is $\frac{n}{\log_e(n)}$

n	$n/\log_e(n)$	exact
10^2	21.7	25
10^3	144.8	168
10^6	72382.4	78498
10^9	$4.8 \cdot 10^7$	50847534

- As we can see, primes are in abundance: we do not have scarcity
- The odds of selecting a prime is high for small numbers: if we select a 2-digit integer, the probability that it is prime is $25/100 = 25\%$
- The odds of selecting a prime less than 10^6 is $78498/10^6 \approx 7.8\%$
- If we make sure that this number is not divisible by 2 or 3, (which makes up 2/3 of integers), the odds increase to 23.5%

Primes

- As the numbers get larger, which would be the case for cryptographic applications, the ratio becomes less and less
- The ratio of 1024-bit (308-digit) primes to the 308-digit numbers is

$$\frac{1}{\log_e(2^{1024})} \approx \frac{1}{714}$$

- Therefore, if we randomly select a 308-digit integer, the probability that it is prime is $1/714$
- If we remove the multiples of 2 and 3 from this selected integer, the odds of choosing a 308-digit prime at random is improved by a factor of 3 to $1/238$

Checking for Primality vs Factoring

- Primality testing: Is $n \in \mathcal{N}$ prime?
  The answer is yes or no (we may not need the factors if n is composite)
- Factoring: What is the prime factorization of $n \in \mathcal{N}$?
  The answer is $n = p_1^{k_1} \cdots p_m^{k_m}$
- Is $2^{101} + 81 = 2535301200456458802993406410833$ prime?
  The answer: Yes
- Is $2^{101} + 71 = 2535301200456458802993406410823$ prime?
  The answer: No
- Factor $n = 2^{101} + 61 = 2535301200456458802993406410813$
  The answer: $n = 3 \cdot 19 \cdot 1201 \cdot 37034944570408560161757109$

Factorization by Trial Division

- Trial division (exhaustive search): Find a prime factor of $n \in \mathcal{N}$ by dividing n by numbers that are smaller than n
- Observation 1: We do not need to divide n by composite numbers
- It is sufficient that we only try primes, for example, if n is divisible by 6, then we could have discovered earlier that it was divisible by 2
- Observation 2: One of the factors of n must be smaller than $\sqrt{n}$, otherwise if $n = pq$ and $p > \sqrt{n}$ and $q > \sqrt{n}$ implies $pq > n$

Factorization by Trial Division

- Trial division finds a prime factor of $n \in \mathcal{N}$ by dividing n by k for $k = 2, 3, \ldots, \sqrt{n}$
- Trial division requires $O(\sqrt{n})$ divisions (in the worst case)
- If n is a k-bit number, then $n = O(2^k)$ and the number of divisions is $O(2^{k/2})$ which is exponential in k

Factorization by Trial Division

- For example, factoring $2^{101} + 61$ requires about 2^{50} divisions
- Assuming one division requires 1 μs, this would take 35 years!
- However, this is the worst case analysis, which assumes a prime divisor is as large as it can be $\approx \sqrt{n}$
- If n has a small divisors, they will be found more quickly
- For example, $2^{101} + 61$ has smaller factors such as 3, 19, and 1201, and thus, the trial division algorithm would quickly find them
- Therefore, we conclude that if $n = p \cdot q$ such that $p, q \approx \sqrt{n}$, then the trial division would take the longest time

Factorization by Trial Division

- The number of divisions for factoring n with large prime factors is exponential in terms of the number of bits in n
- Trial division starts from $k = 2$ and increases k until $\sqrt{n}$, and thus, it is very successful on numbers which have small prime factors: these factors would be found first, reducing the size of the number to be factored
- For example, given $n = 1227331068230022428624 11$, we would find the smaller factors 17, 31, and 101 first, and divide them out

$$\frac{n}{17 \cdot 31 \cdot 101} = m = 23058430274673049 93$$

and then continue to factor m which is smaller in size than n

Fermat's Trial Division

- Fermat's idea was that if n can be written as the difference of two perfect squares:
$$n = x^2 - y^2$$
then, we can write
$$n = (x - y)(x + y)$$
and therefore, we can find two factors of n
- As opposed to the standard trial division algorithm, Fermat's method starts $x \approx \lceil \sqrt{n} \rceil$ and $y = 1$, and increases y until we find a y value such that $x^2 - y^2 = n$
- Since $x \approx \lceil \sqrt{n} \rceil$, Fermat's methods finds a factor that is closer to the size of $\sqrt{n}$ before it finds a smaller factor

Fermat's Trial Division

- For example, consider $n = 302679949$, we have $\lceil \sqrt{n} \rceil = 17398$
- We start with $x = 17398$ and $y = 1$, increase y as long as $x^2 - y^2 \leq n$
- We either find a y such that $x^2 - y^2 = n$ or the selected value of x does not work, i.e., we cannot find y such that $x^2 - y^2 = n$, then we increase x as $x = x + 1$ and start with $y = 1$ again
- It turns out for $x = 19015$, we find $y = 7674$ such that

$$x^2 - y^2 = 19015^2 - 7674^2 = 302679949 = n$$

therefore, n is factored as $n = (x - y)(x + y)$ such that

$$n = (19015 - 7674)(19015 + 7674) = 11341 \cdot 26689$$

Kraitchik's Method

- Instead of looking for x and y satisfying $x^2 - y^2 = n$, we can also search for "random" x and y such that

$$x^2 = y^2 \pmod{n}$$

- For such a pair (x, y), factorization of n is not guaranteed
- We only know the difference of the squares is a multiple of n:

$$x^2 - y^2 = (x - y)(x + y) = 0 \pmod{n}$$

- Since n divides $(x - y) \cdot (x + y)$, we have $1/2$ chance that prime divisors of n are distributed among the divisors of both of these factors
- The GCD$(n, x - y)$ will be a nontrivial factor, the GCD will be neither 1 nor n

Kraitchik's Method

- For $n = 221 = 13 \cdot 17$, we find $x = 4$ and $y = 30$, such that $4^2 = 16 \pmod{221}$ and $30^2 = 900 = 16 \pmod{221}$, and therefore,

$$\text{GCD}(221, 30 - 4) = \text{GCD}(221, 26) = 13$$

- In fact, there are many (x, y) such that $x^2 = y^2 \pmod{221}$, which gives us a higher chance of finding a pair (x, y):

$$(2, 15), (3, 88), (5, 73), \ldots, (11, 28), \ldots$$

- Note that we still perform an exhaustive search to find a pair (x, y)

Dixon's Method

- There is an algorithm due to Dixon to find such squares (x and y) which is slightly more efficient
- It expresses the numbers x and y into small prime powers, and then works with the exponents
- When the exponents of the small primes in the expression are all even, for example, if $x = 2^8 \cdot 3^6 \cdot 5^2 \cdot 7^0 \cdot 11^8$, then x is a square
- The algorithm starts with particular (random) x and y values (which have even powers in their small-prime factorizations), and creates other candidates for x and y which have even powers, and checks for equality $x^2 = y^2 \pmod{n}$ among all such squares

Modern Factorization Methods

- Factorization in general seems to require exhaustive search: modern factorization algorithms differ from one another slightly in the way this search is constructed
- There is no known deterministic or randomized polynomial time algorithm for finding the factors of a given composite integer n, particularly, when $n = p \cdot q$ with size of p and q about half of the size of n
- The best integer factorization algorithm called GNFS (generalized number field sieve) algorithm requires a time complexity of

$$O\left(\exp\left(\left(\frac{64}{9}b\right)^{\frac{1}{3}}(\log b)^{\frac{2}{3}}\right)\right)$$

where b is the number of bits in n

Complexity of Factorization

- It is not known exactly which complexity classes contain the decision version of the integer factorization problem
- It is known to be in $\mathcal{NP}$ since a YES answer can be verified in polynomial time by multiplication: Are p and q factors of n?
- However, it is not known to be in $\mathcal{NP}$-complete since no such reduction proof is discovered
- Many people have looked for a polynomial time algorithm for integer factorization, and failed
- On the other hand, factorization problem can be solved in polynomial time on a quantum computer, using Shor's algorithm