Practical Exercises 2 - Primality testing and factorisation

September 22, 2022

The goal of these exercises is to familiarize withclassical arithmetical problems such as primality testing or factorisation, and their impact on public key cryptograhy. For implementations, it is advised you do a first draft using SageMath before translating it into C with Gmp.

Exercise 1: Primality testing

One of the main algorithms used to verify whether a given integer n is a prime number is *Miller-Rabin* algorithm[1]. It consists in successively drawing integers from $[\![2, n-1]\!]$ and checking if they are *witnesses* for the compositness of n. If no witness is found, then n is deemed to be prime.

1. Verify experimentally the prime number theorem : compute the average number of primality trials required to generate a key for RSA-1024 and compare this value to the theoretical one.
2. Implement the Miller-Rabin primality test.
3. Under the Generalised Riemann Hypothesis (GRH), the smallest witness for a composite n is known to be less than $2\ln n^2$. Deduce from this a deterministic primality test, called Miller test.
4. Compare the performances of your implementations to the ones of functions given by SageMath or Gmp.

Exercise 2: Pollard's $p-1$

Pollard's $p-1$ algorithm[2] is a factorisation algorithm based on the following idea. If $p \mid n$ then there is $a \in [\![2, n-1]\!]$ such that $a^p \equiv 1 \bmod p$.

1. Implement Pollard's $p-1$ algorithm and use it to factorise

$$n = 1178276814202715840174329035223273033253449480506652339565458 63.$$

2. The success of the algorithm depends heavily on the shape of the factors of n. After explaining which are such weak factors, write a function generating numbers n for which Pollard's method would not retrieve a factor in reasonable time.

Exercise 3: Fermat's method

Fermat's method[3] tries to factorise an integer n by writing it as the diffence between two squares.

1. Implement Fermat method and use it to factorise

$$n = 443363497731795997718971635197891857229652767733117521088 1861.$$

2. Can you generate composite integers for which Fermat method fails to compute a factor in reasonable time ?

[1] https://en.wikipedia.org/wiki/Miller-Rabin_primality_test
[2] https://en.wikipedia.org/wiki/Pollard's_p_-_1_algorithm
[3] https://en.wikipedia.org/wiki/Fermat's_factorization_method