Number theory in cryptography

– Exercise set 1 –

The exercises T1.1, P1.6 have to be handed in on Wednesday, 28th February 2024, 8:30 at latest. Theoretical exercises have to be uploaded on **Moodle**, as a PDF file (e.g., a scan of a handwritten version or a PDF obtained from a LaTeX file).

Programming exercises (as P1.6 here) have to be completed in the file `HW_1_2024_SAGE.ipynb` to be found in the `CoCalc` project (see P1.1 below). We will then automatically collect this file online; you do *not* need to send/share it.

Theoretical questions

T 1.1

a) Prove in detail that if two maps $T, g : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ are bounded on any bounded interval, satisfy $T(x) = 2T(x/2) + g(x)$ for every real number $x \geqslant 1$ and $g(x) = \mathcal{O}(x)$, then $T(x) = \mathcal{O}(x \log(x))$.

b) Let $a, b \geqslant 2$ be integers. Prove that if a map $T : \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$ satisfies $T(n) \leqslant aT(\lceil n/b \rceil)$ for all $n \geqslant 1$, then $T(n) = \mathcal{O}(n^{\log_b(a)})$.

T 1.2

Using $(aX+b)(cX+d) = acX^2+(ad+bc)X+bd$ and $ad+bc = (a+b)(c+d)-ac-bd$, design a recursive multiplication algorithm for large integers that is faster than the traditional schoolbook method and analyze rigorously its runtime.

Programming exercises

P 1.1

a) Register on SageMathCloud through https://cocalc.com/, preferably with your EPFL email address. Then **please send an email to `gauthier.leterrier@epfl.ch` once you registered** by indicating which email address you used to register, so that you will be added to the course management system. You will then find the Sage worksheets for your programming assignments in the folder "Assignments" of the project "`Number theory in cryptography - Spring 2024`".

   Optional: if you want to install Sage locally, you can download it from https://www.sagemath.org/download.html.

b) Learn/recall some basic Python from the link https://en.wikibooks.org/wiki/A_Beginner's_Python_Tutorial.

c) Get familiar with the SAGE basic tutorial https://doc.sagemath.org/html/en/reference/. You should pay special attention to the sections on linear algebra, group theory, basic rings, number theory. (See also https://doc.sagemath.org/html/en/tutorial/ and https://doc.sagemath.org/html/en/reference/genindex.html for later use).

Last update : 18/02/2024, 22 : 37

P 1.2 Read about lists in Python/SAGE. Write a function called `myDivisors(N)` that takes as input an integer N and returns a list of all divisors of N (including 1 and N).

P 1.3 Write a function called `numPrimes(x)` that calculates the number of primes in the interval $[1, x]$. Write another function called `myPrimes(x)` that outputs a list of all primes in $[1, x]$.

P 1.4 Plot some values of the function `numPrimes(x)` and make a conjecture about an explicit function on x that approximates your graph. Hint: you may look up the *prime number theorem* (and also *Skewes' number*) on a search engine.

P 1.5

a) Write a function called `coprimes(N)` that takes as input an integer N and outputs a list of all integers m, $1 \leqslant m \leqslant N$ that are relatively prime to N. Write another function called `numCoprimes(N)` that computes the size of this list.

b) Given an integer N, write a function that calculates the sum of `numCoprimes(d)` over all divisors $d \mid N$ (including 1 and N). Calculate your function for several arbitrary values that you choose (e.g., $N = 100, 200, 1000, 2048$)? What do you observe?

P 1.6 Write a Sage function `Karatsuba(A, B)` that takes two integers A and B and returns their product using the algorithm from T1.2. Run and time the above algorithm on 100 pairs of uniformly random positive integers less than 2^{512}.