Number theory in cryptography

– Exercise set 3 –

The exercises T3.1 a), b) and T3.5 have to be handed in on Tuesday, 12th March 2024, 8:30 at latest. As usual, 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 have to be done in the relevant file in the `CoCalc` project.

THEORETICAL QUESTIONS

T 3.1 For an integer $n \geqslant 1$, let $\ell(n)$ be the shortest length of an addition chain $a_0 = 1 < a_1 < ... < a_{\ell(n)} = n$ (i.e., for every integer k such that $1 \leqslant k \leqslant \ell(n)$ there are indices $0 \leqslant i, j < k$ such that $a_i + a_j = a_k$).

a) Show that if $2^s \leqslant n < 2^{s+1}$ and $s \geqslant 1$ then $s \leqslant \ell(n) \leqslant 2s$.

b) Prove that if $r \geqslant 1, s \geqslant 0$ are integers such that $2^{rs} \leqslant n < 2^{r(s+1)}$ then there is an addition chain[1] for n of length at most $(r+1)s + 2^r - 2$ and which starts with $a_i = i$ for all $i \in \{0, ..., 2^r - 2\}$, that is:

$$a_0 = 1, \;\; a_1 = 2, \;\; a_2 = 3, \;\; ..., \;\; a_{2^r-2} = 2^r - 1.$$

Hints: proceed by induction on s. In the induction step, you can work with the euclidean division of n by 2^r (and use the induction hypothesis on the quotient).

c) By choosing $r = \lceil \log(\log(n)) \rceil$ for $n \geqslant 3$ in b), where $\log = \ln$ is the logarithm in base e, deduce that for large enough n we have $\ell(n) \leqslant \log_2(n)(1 + f(n))$ where f is a function such that $\lim_{n \to +\infty} f(n) = 0$. Note: we also denote this by $\ell(n) \leqslant \log_2(n)(1 + o(1))$.

T 3.2 (optional) Find an addition chain $a_0 = 1 < a_1 < ... < a_8 = 63$ of length 8 for $n = 63$.

T 3.3 Let p be a prime. We refer to a set of integers $\{b_1, \ldots, b_{p-1}\}$ as a *complete residue system* mod p if the set $\{b_1 \mod p, \ldots, b_{p-1} \mod p\}$ is a permutation of $\{1, \ldots, p-1\}$.

a) Let $\{b_1, \ldots, b_{p-1}\}$ be a complete residue system. Show that if a is an integer that is relatively prime to p then $\{ab_1, \ldots, ab_{p-1}\}$ is again a complete residue system.

b) Use (a) to deduce Fermat's little theorem, i.e., if $\gcd(a, p) = 1$ then $a^{p-1} \equiv 1 \pmod{p}$.

c) How can you prove Euler's theorem with a similar idea?

T 3.4 Let p be a prime. For $0 \leqslant k \leqslant p$, consider the binomial coefficients

$$\binom{p}{k} = \frac{p!}{k!(p-k)!}.$$

a) Prove that if $0 < k < p$ then $\binom{p}{k}$ is divisible by p.

b) Use (a) to show that if x and y are arbitrary integers then

$$(x + y)^p \equiv x^p + y^p \mod p.$$

[1] To be very precise for the case $s = 0$, we should say "there is an addition chain for $\max\{n, 2^r - 1\}$".

c) Use (b) and induction to show that if $x_1, \ldots, x_k$ are integers then

$$(x_1 + x_2 + \cdots + x_k)^p \equiv x_1^p + \cdots + x_k^p \bmod p.$$

Use that to give another proof of Fermat's little theorem.

T 3.5 Let $\varphi(n)$ be the Euler totient function (i.e., the number of integers in $\{1, 2, \ldots, n\}$ that are coprime to n). Show that $\sum_{d|n} \varphi(d) = n$.

T 3.6 Recall how the RSA algorithm works. Moreover, given an odd integer N which is known to be a product of two distinct primes p, q, prove that computing $\phi(N) := (p-1)(q-1)$ is equivalent to factoring N. Note: this does *not* mean that breaking RSA is equivalent to factoring N.

Programming exercises

P 3.1 Here, you will do some basic SAGE exercises related to some bit operations:

a) Write a simple one-line command (in SAGE) that calculates $3^{4324324}$ modulo $2^{1000000}$ using the Python/SAGE generic `%` (modulo) operator. Time the calculation. You probably notice that it is quite slow. Using the Python "AND" operator `&`, show how to write another one-line command that speeds this up. Give a short justification of why you are getting the same answer. Now, implement a function `myLSB(N, k)` that takes as input integers N and k and outputs the k least significant bits in the binary representation of N.

b) Recall Python's right-shift (`>> k`) and left-shift (`<< k`) operators (look them up in the Python manual or on a search engine). Implement a function `myMSB(N, k)` that takes an integer N and an integer k and returns the k most significant bits of the binary representation of N.