Number theory in cryptography

– Exercise set 9 –

The exercises T9.1 a), b), c), P9.1 have to be handed in on Tuesday, 30th April 2024, 8:30 at latest.

THEORETICAL QUESTIONS

T 9.1 This is an exercise on basic probability theory that will help you understand the complexity analysis of Pollard's rho method.

a) Suppose that you have an urn of N balls, each of which is colored in one of N distinct colors. Suppose that you draw a random ball, record its color and return the ball back to the urn. You then repeat this process. Given an integer $k \geqslant 1$, what is the probability that after k draws, the colors that you have recorded are all distinct?

b) What is the minimal number of people in a room so that the probability of having two of them born on the same day exceeds 50%? (Here a numerical computation with $N = 365$ suffices).

c) Now, for the setting in (a), show that the expected number of draws needed to record some color twice is $1 + Q(N)$, where

$$Q(N) = \sum_{k=1}^{N} \frac{N!}{(N-k)!N^k}.$$

d) Finally, to get an approximation of $Q(N)$, use the following formula due to the famous indian mathematician Srinivasa Ramanujan:

$$Q(N) = \sqrt{\frac{\pi N}{2}} - \frac{1}{3} + \frac{1}{12}\sqrt{\frac{\pi}{2N}} - \frac{4}{135N} + \frac{1}{288}\sqrt{\frac{\pi}{2N^3}} + \frac{8}{2835N^2} + \mathcal{O}(N^{-5/2})$$

to get what people call the *birthday bound* for the expected number of steps to obtain a collision. This is a very useful bound in analyzing the expected run time of some algorithms such as Pollard rho.

T 9.2 (Cycle-finding). Let S be a finite set and let $f\colon S \to S$ be a function. Consider the sequence $x_0 \in S$, $x_1 = f(x_0), ..., x_{n+1} = f(x_n)$. Let μ be the minimal index such that x_μ repeats in the sequence and let λ be the smallest integer such that $x_{\mu+\lambda} = x_\mu$.

Show that there exists an integer $i > 0$ such that $x_{2i} = x_i$. Use this observation to design an algorithm to compute the simple cycle in the above sequence (that is, to compute the integers μ and λ).

T 9.3 *A zero-knowledge proof protocol via discrete logarithms in $\mathbb{F}_q^*$.*

Let $q = p^n$, let $x \in \{1, ..., q-2\}$ be your secret, let $g \in \mathbb{F}_q^*$ be a generator and let $h = g^x$. Consider (q, g, h) as your public key. Computing x from these data is equivalent to solving the discrete logarithm problem on input $(g, h = g^x)$. Suppose now that someone (Bob) wants to check that you really know x. One way of doing this is to reveal the secret key x and then everyone can check that $h = g^x$.

Yet, we would like to do that without revealing the secret. You can proceed as follows:

- Choose an integer $y \in [1, q-2]$ uniformly at random and send $k = g^y \in \mathbb{F}_q^*$ to Bob;

- Bob chooses a bit $b \in \{0, 1\}$ and sends it to you;
- Send $z = y + bx \bmod q - 1$ to Bob;
- Bob computes $g^z \in \mathbb{F}_q^*$ and tests that it is equal to $kh^b \in \mathbb{F}_q^*$.

a) Explain why it is not a good idea for you to reuse the same y.

b) Explain why it is important that you send k *before* Bob sends you b.

c) Implement your protocol and in Sage and show how to use it multiple times to prove (in zero-knowledge) that you possess the secret key x. Test this protocol with one of your classmates.

Programming exercises

P 9.1 Write the function `PollardRho`(g, h, p) in SAGE that solves the discrete logarithm problem $g^x \equiv h \bmod p$ in $\mathbb{F}_p^\times$ for a generator g of $\mathbb{F}_p^\times$, using the base point $x_0 = 1$ and the partition $\mathbb{F}_p^\times = P_0 \sqcup P_1 \sqcup P_2$ where $P_j := \{\, x \pmod p \;:\; jp/3 \leqslant x < (j+1)p/3 \,\}$ for $j \in \{0, 1, 2\}$. You may want to use exercise T9.2.

P 9.2 Implement in SAGE the function `IndexCalc`(g, h, p) that solves the discrete logarithm problem $g^x \equiv h \bmod p$ in $\mathbb{F}_p^\times$ for a generator g of $\mathbb{F}_p^\times$, assuming that $p - 1$ is square-free (to simplify the linear algebra part).

P 9.3 Here, you will design and implement (in SAGE) an algorithm to check whether a given positive integer n is a perfect power, i.e., you will check whether there are integers x and $b > 1$ such that $n = x^b$. This is for instance useful in the quadratic sieve (where we need the square root of some integer of the form $x^2 \in \mathbb{Z}$ found using linear algebra over $\mathbb{F}_2$).

a) Given an integer $b > 1$, find an efficient algorithm (working in time polynomial in $\log n$) that decides whether there exists an integer x such that $x^b = n$ and if so, outputs x.

b) Using the algorithm from (a), find an efficient algorithm (working in time polynomial in $\log n$) that tests whether n is a perfect power and if so, returns a pair of positive integers (x, b) with $b > 1$ such that $x^b = n$.

c) Implement your algorithm in SAGE.

P 9.4 Write a small program in SAGE to plot the powers $627^i \bmod 941$ as a function of i. Do you see any patterns emerge? What about if you replace 941 by other (larger) prime numbers? *(Hint: use e.g., the Fermat prime $p = 65537$ and plot $7^i \bmod p$. Then plot $7^{1024 \cdot i}$).*