Homework 4

Collaborators :

Practice Questions

These are practice questions. They will **NOT** be graded. We have also provided the final answers. However, it is up to you to understand how or why the given solution is correct.
You do not need to submit these on Gradescope. However, you may find it easier to just include them in the pdf. In that case, please do not mark these questions on Gradescope.

1. **An Example of Extended GCD Algorithm (20 points).** Recall that the extended GCD algorithm takes as input two integers a, b and returns a triple (g, α, β), such that

$$g = \gcd(a, b), \text{ and } g = \alpha \cdot a + \beta \cdot b.$$

Here $+$ and $\cdot$ are integer addition and multiplication operations, respectively.

Find (g, α, β) when $a = 2025, b = 373$.

Solution.

$$(g, \alpha, \beta) = (1, 7, -38)$$

2. **Finding Inverse Using Extended GCD Algorithm (20 points).** In this problem, we shall work over the group $(\mathbb{Z}_{509}^*, \times)$. Note that 509 is a prime. The multiplication operation $\times$ is "integer multiplication $\mod 509$."

   Use the Extended GCD algorithm to find the multiplicative inverse of 20 in the group $(\mathbb{Z}_{509}^*, \times)$.

   Solution.

   The inverse of 20 in the group $(\mathbb{Z}_{509}^*, \times)$ is 280.

Homework Questions

These are homework questions and will be graded. Please make sure to clearly mark each problem on Gradescope.

1. **Asymptotics and Efficient Algorithms (20 points).** Suppose a cryptographic protocol P_n is implemented using αn^2 CPU instructions. We expect the protocol to be broken with $\beta 2^{n/10}$ CPU instructions. α and β are some positive constants, while n is the parameter of the protocol (such as key length in bits). That is, when we set the parameter $n = \sigma$, to use the protocol, the "good guys" need to run $\alpha\sigma^2$ CPU instructions. While to break the protocol, the "bad guys" need to run $\beta 2^{\sigma/10}$ CPU instructions.

   Suppose, today, everyone in the world uses the primitive P_n using $n = n_0$, a constant value such that even if the entire computing resources of the world were put together for 8 years, we cannot compute $\beta 2^{n_0/10}$ CPU instructions.

   Assume Moore's law holds. That is, every two years, the amount of CPU instructions a CPU can run per second doubles.

   Remark: This problem explains why we demand that our cryptographic algorithms run in polynomial time and it is exponentially difficult for adversaries to break the cryptographic protocols.

   (a) (5 points) Assuming Moore's law, how much faster will the CPUs be 8 years into the future compared to now?

   Solution.

   (b) (5 points) At the end of 8 years, what choice of n_1 will ensure that setting $n = n_1$ will ensure that the protocol P_n for $n = n_1$ cannot be broken for another 8 years? (Recall that currently, setting $n = n_0$ ensures that the adversaries need to run $\beta 2^{n_0/10}$ instruction to break the protocol, which they are unable to do even in 8 years.)

   Intuition: Since future computers (8 years later from today) are now faster (based on your answer in part (a)), we need to set our parameters to a larger value to ensure that securities still hold. Your task is to determine how large this new parameter n_1 needs to be compared to the current parameter n_0. Your answer should be an equation for n_1, in terms of n_0 and/or other variables.

   Hint: Start by assuming that the old computers are able to run Γ instruction over an 8-year period. And the new computers are able to run $x \cdot \Gamma$ instruction over an 8-year period.

   Solution.

(c) (5 points) What will be the run-time of the protocol P_n using $n = n_1$ on the new computers as compared to the run-time of the protocol P_n using $n = n_0$ on today's computers? (Recall that P_n is implemented using αn^2 CPU instructions.)

Hint: Start by assuming that today computers are able to run Γ instructions per second.

Your answer should be a ratio of the new run time divided by the old run time.

Solution.

(d) (5 points) What will be the run-time of the protocol P_n using $n = n_1$ on today's computers as compared to the run-time of the protocol P_n using $n = n_0$ on today's computers? (Recall that P_n is implemented using αn^2 CPU instructions.)

Your answer should be a ratio of the new run time divided by the old run time.

Solution.

2. **Another Application of Extended GCD Algorithm (20 points).** Use the Extended GCD algorithm to find $x \in \{0, 1, 2, \dots, 671\}$ that satisfies the following two equations.

$$\begin{aligned} x &= 3 \mod 21 \\ x &= 4 \mod 32 \end{aligned}$$

Note that 21 and 32 are not primes. However, we have the guarantee that 21 and 32 are relatively prime, that is, $\gcd(21, 32) = 1$. Also, note that the number $671 = 21 \cdot 32 - 1$.

Solution.

3. **Square Root of an Element (20 points).** Let p be a prime such that $p = 3 \mod 4$. For example, $p \in \{3, 7, 11, 19 \dots\}$.

   We say that x is a square-root of a in the group $(\mathbb{Z}_p^*, \times)$ if $x^2 = a \mod p$. We say that $a \in \mathbb{Z}_p^*$ is a quadratic residue if $a = x^2 \mod p$ for some $x \in \mathbb{Z}_p^*$. Prove that if $a \in \mathbb{Z}_p^*$ is a quadratic residue then $a^{(p+1)/4}$ is a square-root of a.

   (Remark: This statement is only true if we assume that a is a quadratic residue. For example, when $p = 7$, 3 is not a quadratic residue, so $3^{(7+1)/4}$ is not a square root of 3.)

   Solution.

4. **Lagrange Interpolation (10 points)** Consider distinct $x_1, x_2, \ldots, x_d \in F$, where F is a finite field. Prove the identity:

$$\sum_{i=1}^{d} \prod_{j \in \{1,2,\ldots,d\} \setminus \{i\}} \frac{x_j}{(x_j - x_i)} = 1.$$

Hint: We know that there is a unique polynomial of degree at most $d-1$ that passes through d distinct points $(x_1, 1), (x_2, 1), \ldots, (x_d, 1)$ where $1 \in F$. Can you think of a (simple) polynomial that passes through all these points? What if you try to interpolate this polynomial using Lagrange interpolation? What does the formula give you?

Solution.