

Homework 4

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 $a = 2021, b = 632$.

Solution.

2. **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.

3. **Finding Inverse Using Extended GCD Algorithm (20 points).** In this problem, we shall work over the group $(\mathbb{Z}_{503}^*, \times)$. Note that 503 is a prime. The multiplication operation \times is “integer multiplication mod 503.”

Use the Extended GCD algorithm to find the multiplicative inverse of 20 in the group $(\mathbb{Z}_{503}^*, \times)$.

Solution.

4. **Another Application of Extended GCD Algorithm (20 points).** Use the Extended GCD algorithm to find $x \in \{0, 1, 2, \dots, 779\}$ that satisfies the following two equations.

$$x = 10 \pmod{20}$$

$$x = 7 \pmod{39}$$

Note that 20 and 39 are not primes. However, we have the guarantee that 20 and 39 are relatively prime, that is, $\gcd(20, 39) = 1$. Also, note that the number $779 = 20 \cdot 39 - 1$.

Solution.

5. **Square Root of an Element (20 points).** Let p be a prime such that $p \equiv 3 \pmod{4}$. 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 $x^2 = a \pmod{p}$. We say that $a \in \mathbb{Z}_p^*$ is a quadratic residue if $a = x^2 \pmod{p}$ 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.

6. **Weak One-way Functions (20 points).** Define $S_n = \{0, 1\}^n \setminus \{0, 1\}$. That is, S_n is all n -bit numbers except 0 and 1. Let $h_n: S_n \times S_n \rightarrow \{0, 1\}^{2n}$ be the product function $f(x_1, x_2) = x_1 \cdot x_2$. Present an adversarial algorithm $\mathcal{A}: \{0, 1\}^{2n} \rightarrow S_n \times S_n$ that successfully inverts this function with a constant probability when $(x_1, x_2) \xleftarrow{\$} S_n \times S_n$. Compute the probability of your algorithm successfully inverting the function h_n .

Hint: Intuitively, to invert the function is equivalent to finding one factor of a number. Can you find a factor that shows up with constant probability?

Hint: Your algorithm is allowed to fail with constant probability. This also means you are allowed to design an algorithm that sometimes (with constant probability) “gives up” and outputs wrong/arbitrary/dummy values.

Solution.

7. **Extra-Credit** Consider distinct $x_1, x_2, \dots, x_d \in F$, where F is a finite field. Prove the identity:

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

Collaborators :