Problem 1: The Extended Euclidean GCD Algorithm

On input integers \(x\) and \(y\), the extended Euclidean GCD algorithm finds integers \(a\) and \(b\) such that \(ax + by = \gcd(x, y)\). This is done by careful bookkeeping throughout the execution of the Euclidean GCD algorithm, and is described in more detail in Chapter 4 of *A Computational Introduction to Number Theory and Algebra* by Victor Shoup.

Note that you need not include your work— but make sure to check your final answer!

a. Use the Euclidean algorithm to compute \(\gcd(1239, 735)\).

b. Use the Extended Euclidean Algorithm to find \(L, K\) such that \(27L + 62K = 1\).

c. Find \(27^{-1} \mod 62\).

d. Find \(1245^{-1} \mod 143\).

Problem 2: Practice with the Chinese Remainder Theorem

Recall the Chinese Remainder Theorem, which we saw in class. (It is also in Section 2.4 of Victor Shoup’s book mentioned in previous problem)

Let \(n = 7 * 19 * 43\). Suppose that we know that

\[
\begin{align*}
x &\equiv 4 \mod 7 \\
x &\equiv 11 \mod 19 \\
x &\equiv 5 \mod 43
\end{align*}
\]

Let’s use the Chinese remainder theorem to find \(x \mod n\).

a. Use the extended Euclidean algorithm to find \(a, b\) such that \(1 = a*7 + b*19*43\).

b. Use the extended Euclidean algorithm to find \(a, b\) such that \(1 = a*19 + b*7*43\).

c. Use the extended Euclidean algorithm to find \(a, b\) such that \(1 = a*43 + b*7*19\).

d. Use the CRT to find \(x \mod 7 * 19 * 43\).

Problem 3: The Blum TDP

The Blum TDP works as follows:

**Key generation** On an input of the security parameter \(1^k\) produce modulus \(N = pq\), where \(p\) and \(q\) are both random \(k\)-bit primes such that \(p \equiv q \equiv 3 \mod 4\). Let \(PK = N\) and \(SK = \{p, q\}\).

**Sampling the domain** The domain of the permutation consists of the squares \(\mod N\). To sample from the domain, pick a random \(z \in \mathbb{Z}_N^*\) and output \(x = z^2 \mod n\). We denote this domain by \(\text{QR}_N\), for “quadratic residues.”
**Computing the permutation** The permutation itself is simply squaring; i.e., \( f_N(x) = x^2 \mod N \).

In this problem, we will analyze the Blum TDP.

a. Prove that for all \( N \), \( f_N \) is a permutation over \( \text{QR}_N \). Here is a suggested outline for this proof (you may do it differently if you prefer):

   (a) Recall that for \( p \) a prime, \( x^{p-1} \equiv 1 \mod p \) if and only if \( x \) is a square modulo \( p \). Furthermore, recall that for \( p \equiv 3 \mod 4 \), \(-1\) is not a square mod \( p \).

   (b) Infer that if \( x \) is a square mod \( p \) and \( p \equiv 3 \mod 4 \), then \( x \) has a unique square root mod \( p \) that is itself a square.

   (c) Use the Chinese Remainder Theorem to show that it follows that \( f_N \) is a permutation.

b. Prove the following lemma: if \( p = 4m + 3 \) is prime and \( a \) is a quadratic residue modulo \( p \), then \( a^{m+1} \) is a square root of \( a \) modulo \( p \).

c. Give an algorithm for computing \( f_N^{-1} \) given the trapdoor \( \{p, q\} \). (Hint: use what you just proved and the CRT.)

d. Suppose that we are given \( x, y \), where \( x \neq \pm y \mod N \), and such that \( x^2 = y^2 \mod N \). What can you infer about the value \( x - y \mod N \)? Show how to use this value to factor \( N \).

e. Suppose that we have a polynomial-time adversary \( A \) that breaks the Blum TDP. We wish to show that this implies a polynomial-time algorithm for factoring the corresponding modulus. Consider the following reduction \( B \): on input \( N \), \( B \) sets \( x \leftarrow Z^*_N \) and runs \( A(N, x^2 \mod N) \) to obtain some \( y \). If \( y \neq \pm x \mod N \), then use part (d) above to factor \( N \).

Analyze this reduction.

**Problem 4: One-Way Functions Under XOR**

Let \( f_1 \) and \( f_2 \) be OWFs with the same-size output (i.e., if \( |x_1| = |x_2| \), then \( |f_1(x_1)| = |f_2(x_2)| \)). Now consider \( f(x) = f_1(x_1) \oplus f_2(x_2) \), where \( x = x_1 \circ x_2 \) so that \( |x_1| = \lceil |x| / 2 \rceil \), and \( |x_2| = \lfloor |x| / 2 \rfloor \), and when XORing strings of unequal length, you can pretend that blank characters at the end of the shorter strings are 0’s.

a. Assuming that length-preserving one-way functions exist, give an example of OWFs \( f_1 \) and \( f_2 \) such that \( f \) is a OWF, and prove that in this case \( f \) is a OWF. You must also show that your choices of \( f_1 \) and \( f_2 \) are OWFs.

b. Assuming that length-preserving one-way functions exist, give an example of OWFs \( f_1 \) and \( f_2 \) such that \( f \) is NOT a OWF, and show that in this case \( f \) is not a OWF. You must also show that your choices of \( f_1 \) and \( f_2 \) are OWFs.