Fundamentals of Cryptography

Homework 5

Dr. Mohammad Dakhilalian

Fall 2024

Theory Part

Thoroughly review **Chapters 8 & 9** of the book *Understanding Cryptography* to confidently address the questions.

Question 1

Encrypt the following messages using the Elgamal scheme with $p = 751$ and $\alpha = 3$:

1. $k_{\text{pr}} = 123, i = 320, x = 71$
2. $k_{\text{pr}} = 123, i = 210, x = 45$
3. $k_{\text{pr}} = 500, i = 120, x = 500$

Now decrypt every ciphertext and show all steps.

Question 2

Find the primitive root (generator) for each of the numbers below.

1. $n = 11$
2. $n = 11^2$
3. $n = 2 * 11^2$
4. $n = 11^{100}$

Question 3

Bob wants to encrypt the plaintext $m = 10101$ using the Elgamal algorithm with parameters $p = 44927$, $a = 7$, and $d = 22105$. Find the public key, ciphertext, and ciphertext decryption.

Question 4

Let E be an elliptic curve defined over $\mathbb{Z}_7$:

$$E : y^2 \equiv x^3 + 3x + 2$$

1. Compute all points on E over $\mathbb{Z}_7$.
2. What is the order of the group?
3. Given the element $\alpha = (0, 3)$, determine the order of α. Is α a primitive element?

Question 5

Given the elliptic curve:

$$y^2 \equiv x^3 + 2x + 2 \pmod{17},$$

and the point $P = (6, 3)$, calculate $13P$ using the Double-and-Add Algorithm. Please ensure that you write out all calculations and equations for each step.

Question 6

Your task is to compute a session key in a Diffie–Hellman Key Exchange protocol based on elliptic curves. Your private key is $a = 6$. You receive Bob's public key $B = (5, 9)$. The elliptic curve being used is defined by

$$y^2 \equiv x^3 + x + 6 \pmod{11}$$

Programming Part

Question 7

The Elgamal encryption scheme is widely used in cryptography and consists of three phases: the **setup phase**, the **encryption phase**, and the **decryption phase**. Write a Python program that implements the Elgamal encryption protocol based on **section 8.5.2 in the reference book** and the following steps:

1. Setup Phase

- Create a function `elgamal_setup(p, α, private_key=None)`:
  - Takes as input a large prime number p, a primitive element α, and an optional private key d.
  - If d is not provided, generate it randomly as $d \in \{2, \ldots, p-2\}$.
  - Compute β, where β is Bob's public key.
  - Return the public key (p, α, β) and private key d.

2. Encryption Phase

- Create a function `elgamal_encrypt(public_key, message)`:
  - Takes as input the public key (p, α, β) and the message m to be encrypted.
  - Select a random ephemeral key $i \in \{2, \ldots, p-2\}$.
  - Compute k_E, k_M, y
  - Return the ciphertext as (k_E, y).

3. Decryption Phase

- Create a function `elgamal_decrypt(private_key, p, ciphertext)`:
  - Takes as input Bob's private key d, the prime p, and the ciphertext (k_E, y).
  - Compute the masking key k_M.
  - Compute the modular inverse of k_M using the extended Euclidean algorithm.
  - Decrypt the message as m.
  - Return the decrypted message.

Requirements

- Include a helper function `mod_exp(base, exp, mod)` for modular exponentiation.
- Include a helper function `mod_inverse(a, p)` to calculate the modular inverse of $a \mod p$ using the extended Euclidean algorithm.
- Demonstrate the correctness of your implementation by testing the program with the following scenarios:

Test Scenarios

1. **Key Generation**
   - Use $p = 467$ (a prime number) and $g = 2$ (a primitive element).
   - Generate Bob's public and private keys.
2. **Encryption**
   - Encrypt the following messages:
     - $m = 33$,
     - $m = 105$,
     - $m = 248$,
     - $m = 300$.
   - For each message, generate a random ephemeral key and compute the ciphertext (k_E, y).
3. **Decryption**
   - Use Bob's private key to decrypt the ciphertext and verify that the decrypted message matches the original message.

Output

Your program should print:

1. The generated public and private keys.
2. The original message, ciphertext, and decrypted message for each test case.
3. Validate that the decrypted message matches the original message for all test cases.

Deliverables

1. Submit your Python code in a single .py file.
2. Take screenshots from your code showing its correct execution for the above tests.