

MATH 470: Communications and Cryptography**Homework 4***Due date: 20 September 2023**Name: Huy Lai*

Problem 1. Alice and Bob agree to use the prime $p = 1373$ and the base $g = 2$ for a Diffie–Hellman key exchange. Alice sends Bob the value $A = 974$. Bob asks your assistance, so you tell him to use the secret exponent $b = 871$. What value B should Bob send to Alice, and what is their secret shared value? Can you figure out Alice’s secret exponent?

Solution:

Bob’s sends $B = g^b \bmod p \equiv 805 \bmod 1373$ to Alice.

Their shared secret is $A^b \bmod p \equiv 397 \bmod p$

There is no really easy way to determine Alice’s secret exponent, but with a computer or even a programmable calculator, it does not take long to compute all of the powers of $2 \bmod 1373$.

Problem 2. Alice and Bob agree to use the prime $p = 1373$ and the base $g = 2$ for communications using the Elgamal public key cryptosystem.

Subproblem 1. Bob chooses $b = 716$ as his private key, so his public key is

$$B \equiv 2^{716} \equiv 469 \pmod{1373}$$

Alice encrypts the message $m = 583$ using the random element $k = 877$. What is the ciphertext (c_1, c_2) that Alice sends to Bob?

Solution:

$$c_1 = g^k \pmod{p} \equiv 719 \pmod{1373}$$

$$c_2 = m \cdot B^k \pmod{p} \equiv 623 \pmod{1373}$$

$$(c_1, c_2) = (719, 623)$$

Subproblem 2. Alice decides to choose a new private key $a = 299$ with associated public key $A \equiv 2^{299} \equiv 34 \pmod{1373}$. Bob encrypts a message using Alice's public key and sends her the ciphertext $(c_1, c_2) = (661, 1325)$. Decrypt the message.

Solution:

Decrypting the message is as follows

$$\begin{aligned} m &= (c_1^{-1})^a c_2 \pmod{p} \\ &= (661^{299})^{-1} \cdot 1325 \pmod{1373} \\ &= 645^{-1} \cdot 1325 \pmod{1373} \\ &= 794 \cdot 1325 \pmod{1373} \\ &= 332 \end{aligned}$$

Problem 3. Suppose that Eve is able to solve the Diffie–Hellman problem. More precisely, assume that if Eve is given two powers g^u and $g^v \bmod p$, then she is able to compute $g^{uv} \bmod p$. Show that Eve can break the ElGamal PKC.

Solution:

In the Engamal PKC, Eve knows Alice’s public key $A \equiv g^u \bmod p$ and Eve knows the ciphertext consisting of two integers $c_1 \equiv g^v \bmod p$ and $c_2 \equiv m \cdot A^v \bmod p$.

Given Eve knows g^u and $g^v \bmod p$, she can compute $g^{uv} \bmod p$.

With this, she can recover the encrypted message by computing $(g^{uv})^{-1} \cdot c_2 \equiv m \bmod p$

Problem 4. Use Shanks’s babystep–giantstep method to solve the following discrete logarithm problem.

$$11^x = 21 \text{ in } \mathbb{F}_{71}$$

Solution:

Since $\gcd(11, 71) = 1$, we can apply Fermat’s Little Theorem to it to calculate the order. The number 11 has order 70 in \mathbb{F}_{71} .

Set $n = \lceil \sqrt{70} \rceil = 9$.

By Fermat’s Little Theorem we can calculate $g^{-n} \equiv g^{n \cdot (p-2)} \bmod p = 7$

i	$g^i \bmod 71$	$hg^{-in} \bmod 71$
0	1	21
1	11	5
2	50	35
3	53	32
4	15	11
5	23	6
6	40	42
7	14	10
8	12	70
9	61	64

$i = 1, j = 4$

Therefore, the solution is $x = 4 \cdot 9 + 1 = 37$

Problem 5. Let $p \geq 3$ be a prime and suppose that the congruence $X^2 \equiv b \pmod{p}$ has a solution. Prove that the congruence $X^2 \equiv b \pmod{p^2}$ also has a solution.

Solution:

To prove that if the congruence $X^2 \equiv b \pmod{p}$ has a solution, then the congruence $X^2 \equiv b \pmod{p^2}$ also has a solution, we can use the Chinese Remainder Theorem (CRT) and a bit of algebraic manipulation.

Proof. Let x_1 be a solution to $X^2 \equiv b \pmod{p}$.

Assume that $X = x_1 + kp, k \in \mathbb{Z}$

Now, we'll square X and see what we get:

$$\begin{aligned} X^2 &= (x_1 + kp)^2 \\ &= x_1^2 + 2kpx_1 + k^2p^2 \end{aligned}$$

Now, let's consider the congruence $X^2 \equiv b \pmod{p^2}$. Substituting our expression for X^2 , we have:

$$x_1^2 + 2kpx_1 + k^2p^2 \equiv b \pmod{p^2}$$

Now, notice that $x_1^2 \equiv b \pmod{p}$ by our assumption. Additionally, k^2p^2 is a multiple of p^2 . So, we can simplify the congruence to:

$$b + 2kpx_1 \equiv b \pmod{p^2}$$

By proposition, this implies that $p^2 \mid (2kpx_1 + b - b) \rightarrow p^2 \mid 2kpx_1$.

This shows that for $k \in \mathbb{Z}$, $X = x_1 + kp$ satisfies the congruence $X^2 \equiv b \pmod{p^2}$. Therefore, we have found a solution to $X^2 \equiv b \pmod{p^2}$, given that a solution exists for $X^2 \equiv b \pmod{p}$. \square

Problem 6. Let p be a large prime and let g be a primitive root mod p . Suppose p and g are publicly known to everybody in a network. Alice chooses a secret integer a and publishes $A = g^a \bmod p$ as her public key by broadcasting it over the network. Bob wants to send Alice a message m using Elgamal encryption. Suppose however that an active adversary Eve has the ability to intercept and modify messages sent over the network (including Alice's broadcasting of her public key) without being detected. Show how Eve can perform a Man-In-The-Middle attack.

Solution:

Alice first publishes $A = g^a \bmod p$ as her public key.

Eve intercepts this and chooses a secret integer e and publishes $E = g^e \bmod p$ as Alice's fake key.

Using this, Bob computes $c_1 = g^b \bmod p$ and $c_2 = m \cdot E^b \bmod p$ sends this message to Eve.

Eve then decrypts the message received from Bob, modifies it to her liking and then sends the new message c_{1e}, c_{2e} to Alice.

Alice then computes $(c_{1e}^a)^{-1} \cdot c_{2e} \equiv m_e \bmod p$.

This completes the Man-In-The-Middle attack.

Problem 7. You may assume that $p = 123456789001907$ is prime and that 2 is a primitive root modulo p . Find an integer x such that $2^x \equiv 3 \pmod{p}$.

Solution:

```
from math import ceil, sqrt

def fast_pow(base: int, power: int, modulo: int) -> int:
    result = 1
    base %= modulo

    while power > 0:
        if power & 1:
            result = (result * base) % modulo
            base = (base * base) % modulo
            power >>= 1

    return result

def lsbs(g: int, h: int, p: int) -> int:
    N = ceil(sqrt(p - 1))

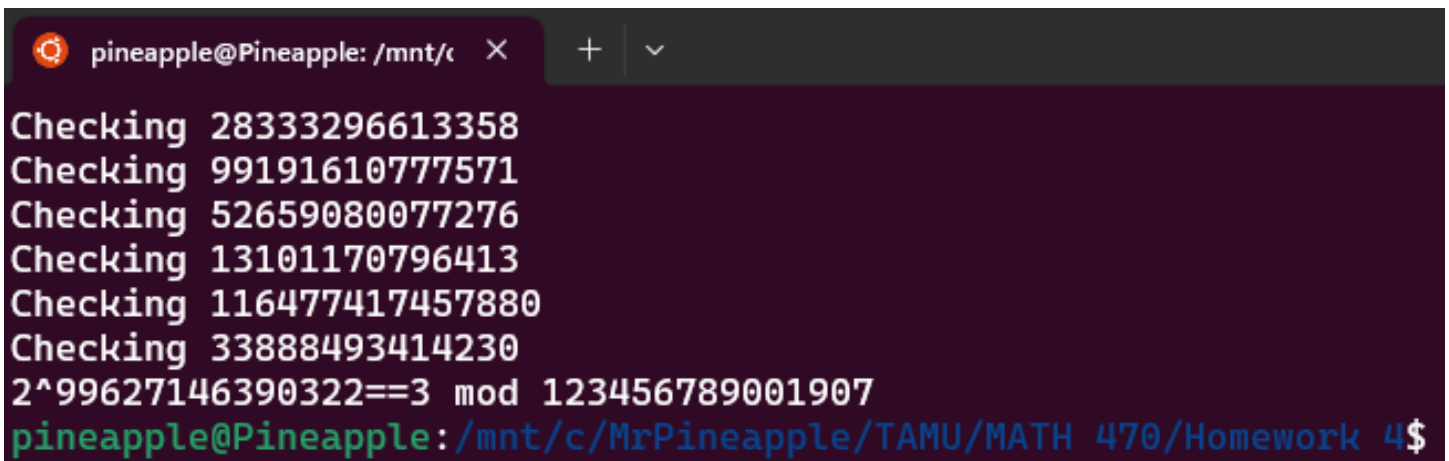
    # Little Step
    little = {fast_pow(g, i, p): i for i in range(N + 1)}

    # Use Fermat's Little Theorem to Calculate  $g^{-n}$ 
    g_inv = fast_pow(g, N * (p - 2), p)

    # Search for an equivalence in the table. Giant step.
    for j in range(N):
        y = (h * fast_pow(g_inv, j, p)) % p
        print(f"Checking {y}")
        if y in little:
            return j * N + little[y]

    # Solution not found
    return None
```

```
def main():  
    p = 123456789001907  
    h = 3  
    g = 2  
    print(f"{g}^{{lsbs(g, h, p)}}=={{h}} mod {{p}}")  
  
if __name__ == "__main__":  
    main()
```

A terminal window with a dark purple background. The title bar shows 'pineapple@Pineapple: /mnt/c' and window control buttons. The output of the script is displayed in white text: 'Checking 28333296613358', 'Checking 99191610777571', 'Checking 52659080077276', 'Checking 13101170796413', 'Checking 116477417457880', 'Checking 33888493414230', and '2^99627146390322==3 mod 123456789001907'. The prompt 'pineapple@Pineapple:/mnt/c/MrPineapple/TAMU/MATH 470/Homework 4\$' is shown at the bottom in green and blue.

```
pineapple@Pineapple: /mnt/c  X + v  
Checking 28333296613358  
Checking 99191610777571  
Checking 52659080077276  
Checking 13101170796413  
Checking 116477417457880  
Checking 33888493414230  
2^99627146390322==3 mod 123456789001907  
pineapple@Pineapple:/mnt/c/MrPineapple/TAMU/MATH 470/Homework 4$
```

Figure 1: Output