Cryptography Homework 6

2024 Spring Semester

21 CST H3Art

Exercise 7.1

Implement SHANKS' ALGORITHM for finding discrete logarithms in $\mathbb{Z}_p^*$, where p is prime and α is a primitive element modulo p. Use your program to find $\log_{106} 12375$ in $\mathbb{Z}_{24691}{}^*$ and $\log_6 248388$ in $\mathbb{Z}_{458009}{}^*$.

Solution:

My Python program can give answers to the above two questions:

- For $\log_{106} 12375$ in $\mathbb{Z}_{24691}{}^*$, the answer is 22392.
- For $\log_6 248388$ in $\mathbb{Z}_{458009}{}^*$, the answer is 232836.

The source code is as follows:

```python
import math


def find_order(p, a):
    n = 1
    while True:
        if pow(a, n, p) == 1:
            return n
        n += 1


def egcd(a, b):
    if a == 0:
        return b, 0, 1
    else:
        g, y, x = egcd(b % a, a)
        return g, x - (b // a) * y, y


def multiplicative_inverse(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('Modular inverse does not exist')
    return x % m


def shanks(p, a, b):
    n = find_order(p, a)
    m = math.ceil(math.sqrt(n))
    a_m = pow(a, m, p)

    giant_steps = {pow(a_m, j, p): j for j in range(m)}
    a_inv = multiplicative_inverse(a, p)
    for i in range(m):
        value = (b * pow(a_inv, i, p)) % p
        if value in giant_steps:
            return (m * giant_steps[value] + i) % n

    raise ValueError("Discrete logarithm not found")


if __name__ == "__main__":
    print(shanks(24691, 106, 12375))
    print(shanks(458009, 6, 248388))
```

Exercise 7.9 (show the basic idea)

Decrypt the ElGamal ciphertext presented in **Table 7.4**. The parameters of the system are $p = 31847$, $\alpha = 5$, $a = 7899$ and $\beta = 18074$. Each element of $\mathbb{Z}_n$ represents three alphabetic characters as in Exercise 6.13.

TABLE 7.4 : ElGamal Ciphertext

$(3781, 14409)$	$(31552, 3930)$	$(27214, 15442)$	$(5809, 30274)$
$(5400, 31486)$	$(19936, 721)$	$(27765, 29284)$	$(29820, 7710)$
$(31590, 26470)$	$(3781, 14409)$	$(15898, 30844)$	$(19048, 12914)$
$(16160, 3129)$	$(301, 17252)$	$(24689, 7776)$	$(28856, 15720)$
$(30555, 24611)$	$(20501, 2922)$	$(13659, 5015)$	$(5740, 31233)$
$(1616, 14170)$	$(4294, 2307)$	$(2320, 29174)$	$(3036, 20132)$
$(14130, 22010)$	$(25910, 19663)$	$(19557, 10145)$	$(18899, 27609)$
$(26004, 25056)$	$(5400, 31486)$	$(9526, 3019)$	$(12962, 15189)$
$(29538, 5408)$	$(3149, 7400)$	$(9396, 3058)$	$(27149, 20535)$
$(1777, 8737)$	$(26117, 14251)$	$(7129, 18195)$	$(25302, 10248)$
$(23258, 3468)$	$(26052, 20545)$	$(21958, 5713)$	$(346, 31194)$
$(8836, 25898)$	$(8794, 17358)$	$(1777, 8737)$	$(25038, 12483)$
$(10422, 5552)$	$(1777, 8737)$	$(3780, 16360)$	$(11685, 133)$
$(25115, 10840)$	$(14130, 22010)$	$(16081, 16414)$	$(28580, 20845)$
$(23418, 22058)$	$(24139, 9580)$	$(173, 17075)$	$(2016, 18131)$
$(19886, 22344)$	$(21600, 25505)$	$(27119, 19921)$	$(23312, 16906)$
$(21563, 7891)$	$(28250, 21321)$	$(28327, 19237)$	$(15313, 28649)$
$(24271, 8480)$	$(26592, 25457)$	$(9660, 7939)$	$(10267, 20623)$
$(30499, 14423)$	$(5839, 24179)$	$(12846, 6598)$	$(9284, 27858)$
$(24875, 17641)$	$(1777, 8737)$	$(18825, 19671)$	$(31306, 11929)$
$(3576, 4630)$	$(26664, 27572)$	$(27011, 29164)$	$(22763, 8992)$
$(3149, 7400)$	$(8951, 29435)$	$(2059, 3977)$	$(16258, 30341)$
$(21541, 19004)$	$(5865, 29526)$	$(10536, 6941)$	$(1777, 8737)$
$(17561, 11884)$	$(2209, 6107)$	$(10422, 5552)$	$(19371, 21005)$
$(26521, 5803)$	$(14884, 14280)$	$(4328, 8635)$	$(28250, 21321)$
$(28327, 19237)$	$(15313, 28649)$		

The plaintext was taken from *The English Patient*, by Michael Ondaatje, Alfred A. Knopf, Inc., New York, 1992.

In order to translate the plaintext back into ordinary English text, you need to know how alphabetic characters are "encoded" as elements in $\mathbb{Z}_n$. Each element of $\mathbb{Z}_n$ represents three alphabetic characters as in the following examples:

$$\begin{aligned}\text{DOG} &\rightarrow 3\times 26^2+14\times 26+6=2398\\ \text{CAT} &\rightarrow 2\times 26^2+0\times 26+19=1371\\ \text{ZZZ} &\rightarrow 25\times 26^2+25\times 26+25=17575\end{aligned}$$

You will have to invert this process as the final step in your program.

Solution:

To decrypt the ElGamal ciphertext in $\textbf{Table 7.4}$ using the parameters $p=31847$, $\alpha=5$, $a=7899$, and $\beta=18074$, we follow these steps:

The ElGamal decryption formula involves calculating the plaintext message m using the ciphertext pair (c_1,c_2) and the private key a. The formula is:

$$m=c_2\cdot(c_1^a)^{-1}\bmod p$$

Here, c_1 and c_2 are the components of the ciphertext, and p is the prime modulus. Nextly we should calculate the multiplicative inverse of $c_1^a \bmod p$. This can be done using the $\text{EXTENDED EUCLIDEAN ALGORITHM}$, and we only decrypt the first ciphertext pair $(3781, 14409)$ to show the basic idea:

Firstly, we calculate the multiplicative inverse of $3781^{7899} \bmod 31847$, $(3781^{7899})^{-1}=6479$, then we can compute the plaintext x_1 as follows:

$$\begin{aligned}x_1&=14409\cdot(3781^{7899})^{-1}\bmod 31847\\ &=14409\cdot 6479 \bmod 31847\\ &=12354\end{aligned}$$

Finally, to convert 12354 back to characters, we use the encoding scheme:

$$12354=18\times 26^2+7\times 26+4\rightarrow \text{“SHE”}$$

By repeating the above steps for each ciphertext pair in the table, we can decrypt the entire ciphertext.

Exercise 7.17

Suppose $e: G_1\times G_2\rightarrow G_3$ is a bilinear pairing. Prove, for all $P\in G_1$ and $Q\in G_2$, that $e(aP,bQ)=e(P,Q)^{ab}$ for any positive integers a and b.

Solution:

Since e is bilinear pairing, and $P\in G_1$, $Q\in G_2$. By the bilinear propety, we have:

$$\begin{aligned}e(P_1+P_2,Q)=e(P_1,Q)e(P_2,Q)\\ e(P,Q_1+Q_2)=e(P,Q_1)e(P,Q_2)\end{aligned}$$

Therefore, for $e(aP,bQ)$, we have:

$$\begin{aligned}e(aP,bQ)&=e(P,bQ)e(\underbrace{P+\cdots+P}_{a-1\text{ times}},bQ)\\ &=e(P,bQ)e(P,bQ)e(\underbrace{P+\cdots+P}_{a-2\text{ times}},bQ)\\ &=\vdots\\ &=e(P,bQ)^a\end{aligned}$$

$$\begin{aligned}
&= [e(P,Q)e(P,\underbrace{Q+\cdots+Q}_{b-1 \text{ times}})]^a \\
&= [e(P,bQ)e(P,bQ)e(P,\underbrace{Q+\cdots+Q}_{b-2 \text{ times}})]^a \\
&= \vdots \\
&= [e(P,Q)^b]^a \\
&= e(P,Q)^{ab}
\end{aligned}$$

Exercise 7.23

The *ElGamal Cryptosystem* can be implemented in any subgroup $\langle\alpha\rangle$ of a finite multiplicative group $(G,\cdot)$, as follows: Let $\beta \in \langle\alpha\rangle$ and define (α,β) to be the public key. The plaintext space is $P = \langle\alpha\rangle$, and the encryption operation is $e_K(x) = (y_1, y_2) = (\alpha^k, x\cdot \beta^k)$, where k is random.

Here we show that distinguishing ElGamal encryptions of two plaintexts can be Turing reduced to **Decision Diffie-Hellman**, and vice versa.

(a) Assume that $\mathrm{ORACLEDDH}$ is an oracle that solves **Decision Diffie-Hellman** in $(G,\cdot)$. Prove that $\mathrm{ORACLEDDH}$ can be used as a subroutine in an algorithm that distinguishes ElGamal encryptions of two given plaintexts, say x_1 and x_2. (That is, given $x_1, x_2 \in \mathcal{P}$, and given a ciphertext (y_1, y_2) that is an encryption of x_i for some $i \in \{1, 2\}$, the distinguishing algorithm will determine if $i = 1$ or $i = 2$.)

(b) Assume that $\mathrm{ORACLE\text{-}DISTINGUISH}$ is an oracle that distinguishes ElGamal encryptions of any two given plaintexts x_1 and x_2, for any *ElGamal Cryptosystem* implemented in the group $(G,\cdot)$ as described above. Suppose further that $\mathrm{ORACLE\text{-}DISTINGUISH}$ will determine if a ciphertext (y_1, y_2) is not a valid encryption of either of x_1 or x_2. Prove that $\mathrm{ORACLE\text{-}DISTINGUISH}$ can be used as a subroutine in an algorithm that solves **Decision Diffie-Hellman** in $(G,\cdot)$.

Solution:

(a) We can calculate $u_i = y_2(x_i)^{-1}$. If $\mathrm{ORACLEDDH}(\alpha,\beta,y_1) = u_i$, then (y_1, y_2) is an encryption of x_i.

(b) Given an instance of the **Decision Diffie-Hellman** problem (α,β,y_1,y_2), set $x_1 = 1$ and let x_2 be any random number except 1. If the result of $\mathrm{ORACLE\text{-}DISTINGUISH}(x_1, x_2, (y_1, y_2))$ is $i = 1$, then the $\mathrm{ORACLE\text{-}DISTINGUISH}$ can be utilized.