Cryptography: Assignment 2

Aladdin Persson (gushijal@student.gu.se)

Innehåll

1 Assingment

1.1 Part 1: Analyzing the generator

a) We've been given the generator $g = 18$ and we are working in $\mathbb{Z}_{\not\models\not\models}{}^*$. The group which it generates is the set $< g >= S = \{g^i | 0 \leq i \leq 21\}$.

A table for the calculations that find this set:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...
g^i	1	18	2	13	4	3	8	6	16	12	1	18	2	13	4	3	8	6	...

where the '...' represent that the calculations continue, however in the same cyclical pattern as previously. The set $S = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$.

b) We note
$2 \cdot 3 = 6 \in S$,
$2 \cdot 4 = 8 \in S$,
$4 \cdot 6 \pmod{23} = 1 \in S$,
$3 \cdot 4 = 12, \in S$
$3 \cdot 16 \pmod{23} = 2 \in S$.

Proposition 1. *Let $q > 0$ and $g \in \mathbb{Z}_{\imath}{}^*$ and has the property $g^q = 1$ then $\{g^i | 0 \leq i < q\}$ is closed under scalar multiplication.*

Proof We want to show that for given any $a, b \in S = \{g^i | 0 \leq i < q\}$ then $a \cdot b \in S$.

Let $a = g^{k_1}$ and $b = g^{k_2}$, then $a \cdot b = g^{k_1+k_2}$. Let us distinguish between two cases that can happen.

Case 1: $k_1 + k_2 < q$, then the result is clear that $a \cdot b \in S$ by definition of S.

Case 2: $k_1+k_2 \geq q$. Note that $k_1+k_2 < 2q$ from defintion of S and hence we can write $k_1+k_2 = q+s$ where we know $0 \leq s < q$. $g^{k_1+k_2} = g^{q+s} = g^q \cdot g^s = 1 \cdot g^s = g^s \in S$.

In both cases we conclude that $a \cdot b \in S$ and this concludes the proof.

1.2 Part 2: Encrypting a message

We know that the public key is $6 = g^x$ and the message $m = 17$. By looking at our table from previous part we note that $x = 7$. To encrypt m we choose a random $k \in Z_{23}^*$, in this case we choose

$k = 3$. We encrypt by calculating $c = (c_1, c_2)$.

Calculate $c_1 = g^k = 13$ in Z_{23}^*. $c_2 = m \cdot g^{x^k} = 17 \cdot (6^3) = 15$ in Z_{23}^*. Hence $c = (13, 15)$.

To assure ourselves that we have encrypted the correct way we can try to decrypt the message in this case. We can do this since we know the fact that $x = 7$. We start with calculating $k = 13^7 = 9$ in Z_{23}^*. $m = c_2 \cdot k^{-1}$, where k^{-1} can be found using the extended euclidean algorithm and in this case we get since $18 \cdot 9 = 1$ in Z_{23}^*, that $k^{-1} = 18$ in Z_{23}^*. Performing this calculation gives us then that $m = 15 \cdot 18 = 17$ in Z_{23}^* and we obtain the original message.

1.3 Part 3: Decrypting a message

What is public is that we are using the group G $= Z_{23}^*$, $g = 18$, $q = 11$ (the order of the generated set), and also $h = g^x = 18^9 = 12 \pmod{23}$. So we can write public key pk $= (G, g, q, h)$

We have the message (13,11)(12,15)(9,10) and the private key $x = 9$. Since we have the private key we can follow the steps to decipher this message.

We use the formula that $k = c_1^x$ in Z_p^* and $m = c_2 \cdot k^{-1}$ in Z_p^*. These calculations in our case when $p = 23$ and we essentially have three different messages we wish to decrypt and then concatenate these into the secret message.

$k = 13^9 = 3 \pmod{23}$, $k^{-1} = 8$ since $3 \cdot 8 = 1 \pmod{23}$. We then get
$m = 11 * 8 = 19 \pmod{23}$. This letter is 'S'.

$k = 12^9 = 4 \pmod{23}$, $k^{-1} = 6$ since $4 \cdot 6 = 1 \pmod{23}$. We then get
$m = 15 * 6 = 21 \pmod{23}$. This letter is 'U'.

$k = 9^9 = 2 \pmod{23}$, $k^{-1} = 12$ since $2 \cdot 12 = 1 \pmod{23}$.We then get
$m = 10 * 12 = 5 \pmod{23}$. This letter is 'E'.

The concatenated decrypted word is then: 'SUE'.

In this example finding the inverses were quite straightforward and the solution were 'seen'. In the general way one would use the extended euclidean algorithm to find k^{-1}. This is how these calculations would look for the first case, i.e the letter 'S' using the extended euclidean algorithm.

$k = 3$, and we wish to find k^{-1} in (mod 23).

$23 = 3 \cdot 7 + 2$

$3 = 2 \cdot 1 + 1$

$2 = 1 \cdot 2 + 0$

Then we move backwards

$1 = 3 - 2 \cdot 1$

$1 = 3 - (23 - 3 \cdot 7) = 3 \cdot 8 - 23$

Which gives us that if we take (mod 23) on both sides, we obtain that $3 \cdot 8 = 1$ (mod 23)

Written in python-like code (where % is modulus and // is integer division) we can write the EEA algorithm recursively as follows

Algorithm 1 EEA(a, b)

```
if a == 0 then
    return $(b, 0, 1)$
else
    gcd, x, y = EEA(b % a, a)
    return (gcd, y - (b//a) * x, x)
```

I believe the steps in calculation is easier to show by example as we did above but the algorithm can also be used.