Homework-4 Report

Emre Can Tüzer

BGK-516

Enver ÖZDEMİR

1.)

The order of g is given as $g^3 \equiv 1\ (mod\ n)$. It indicates that whereas smaller powers do not form a unit element, g to the third power does.

Alice Side:

$A \equiv\ g^a\ (mod\ P)$.

Bob Side:

$B \equiv\ g^b\ (mod\ n)$.

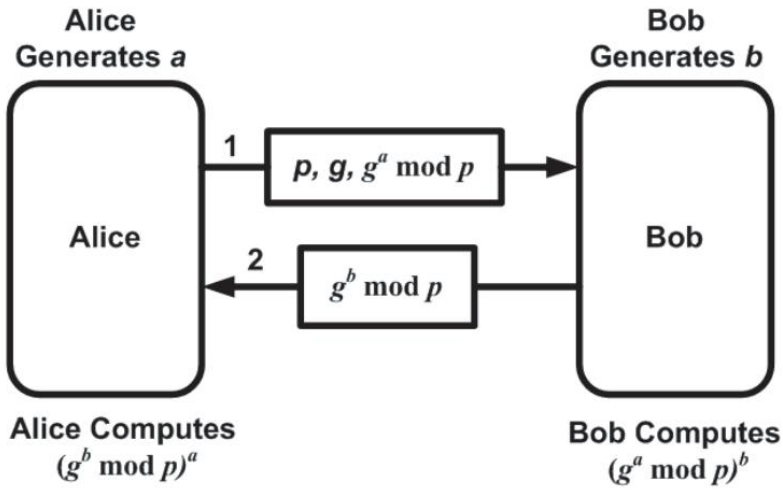


Figure 1: Diffie Hellman Protocol

The public key is computed as

Alice Side Public Key:

$K \equiv\ (g^b\ mod\ n)^a$

Bob Side Public Key:

$K \equiv\ (g^a\ mod\ n)^b$

An attacker looking at g^a and g^b should have an extremely hard time figuring out a or b. This often relies on the group's order G. Subtracting a or b from the values of g^a and g^b gets increasingly challenging as the group's rank increases. However, for whatever value of ab, the outcome of g^{ab} can only be one of these three values because the order of g is only three; g^1, g^2, g^3. The attacker may easily brute force all feasible values for g^a, g^b, and g^{ab} since she is aware of the pattern of g. An attacker can determine the public key by testing the three potential values of K.

2.)

When applying the Diffie-Hellman key exchange protocol to a five-person group, everyone in the group must first agree on a base (g) and a large prime integer (p). Every individual A, B, C, D, E, each choose their private key at random. These secret keys, which are not disclosed to other participants, are designated a, b, c, d, e, respectively.Each participant calculates a value of the form $g^x \ (mod\ p)$ and shares this value with the other four participants. Likewise, B, C, D and E also transmit their calculated values to all group members. Once this step is completed, everyone has received these shared values from others. Every participant uses their private key to calculate the identical public key. Every participant receives the identical public key (K) due to mathematical properties.

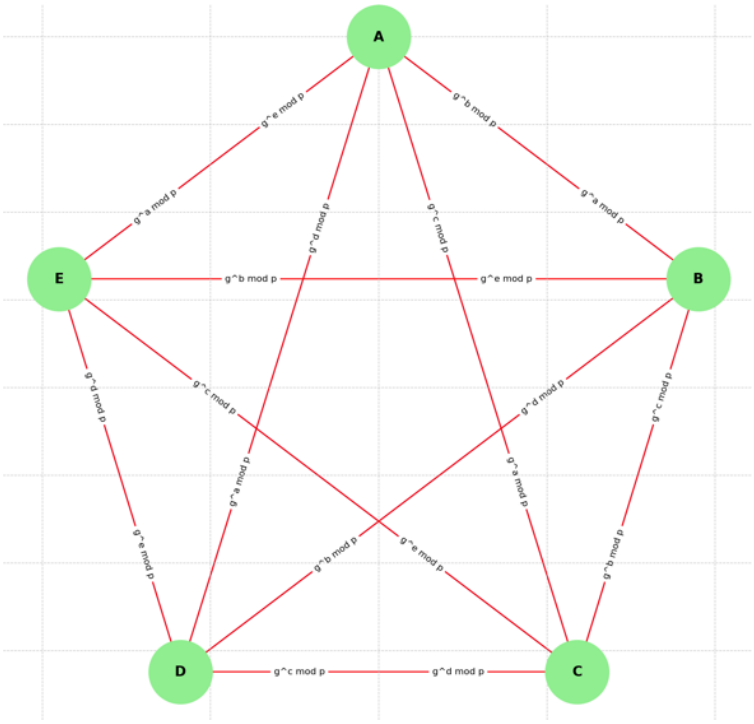
$$\boldsymbol{K_a} \equiv (g^b \cdot g^c \cdot g^d \cdot g^e \cdot mod\ n)^a$$

$$\boldsymbol{K_b} \equiv (g^a \cdot g^c \cdot g^d \cdot g^e \cdot mod\ n)^b$$

$$\boldsymbol{K_c} \equiv (g^a \cdot g^b \cdot g^d \cdot g^e \cdot mod\ n)^c$$

$$\boldsymbol{K_d} \equiv (g^a \cdot g^b \cdot g^c \cdot g^e \cdot mod\ n)^d$$

$$\boldsymbol{K_e} \equiv (g^a \cdot g^b \cdot g^c \cdot g^d \cdot mod\ n)^e$$



3.)

The multiplicative group of all prime numbers with regard to n is this group. This group has inverse components and is closed under modular multiplication. Euler's Totient function is used to determine the number of elements of the group because this function satisfies the divisibility relation in modular arithmetic. In particular, when working with non-prime numbers under mod n, $\varphi(n)$ directly gives the number of these non-prime numbers.

$$n = p \cdot \mathrm{q} = 101 \cdot 1621 = 163921$$

$$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (\mathrm{q}-1) = (101-1) \cdot (1621-1) = 162000$$

This means we can say that the group contains exactly 162000 elements under it. The question asks us to find the element a.

$$a^2 = 14641\ (mod\ 163921) = k_2 \cdot 163921 + 14641$$

$$a^3 = 75244\ (mod\ 163921) = k_3 \cdot 163921 + 75244$$

$$a^5 = 88789\ (mod\ 163921) = k_5 \cdot 163921 + 88789$$

Finding the a values that fulfill the provided equations requires the use of the brute-force approach, which entails checking all possible values of a between 1 and $n-1$, calculating each value in mod n, and comparing it with the target values.

```
for (mpz_set_ui(a2, 0); mpz_cmp(a2, mod) < 0; mpz_add_ui(a2, a2, 1)) {
    mpz_powm(result2, a2, exponent2, mod);
    if (mpz_cmp(result2, target2) == 0) {
        break;
    }
```

The expression mpz_cmp(a^2, mod) < 0 checks whether the value of a^2 is less than 163921.

If a^2< 163921, returns true (1);

Else a^2 >= 163921, returns false (0).

The mpz_powm(result, a, exponent, mod) function performs a^2 (mod 163921) and assigns the result to the result variable. The modular exponent is computed using the mpz_powm function in the GMP package.

$$result = a^2\ (mod\ 163921)$$

In the if loop, we obtain our result when the result value and our value are equal.

We do the same for other exponent of a values and we will successfully complete the brute force process. The outputs of all values are below. For the question at the end, as we see in the output. If we have a modular root finding function, this can break the basic encryption structure of RSA.

```
C:\Users\emrecan\Desktop\BGK_516_HW4\question4\cmake-build-debug\question4.exe
 A^2 = 121
 A^3 = 97875
 A^5 = 121074

Process finished with exit code 0
```