Scribe: Cryptography and Network Security (Week 10: Class.5)

Rohit - 20CS60R71

12-Nov-2020

1 The Discrete Logarithm

Consider a finite mathematical group (G,.) under the operation of multiplication is called as discrete logarithm if we start with an element $\alpha \epsilon G$ and suppose that its order is n, then that means i can form subgroup α raise to the any power i from 0 to n-1.

$$< \alpha >= (\alpha^i : 0 \leq i \leq n-1)$$

So, a Discrete Logarithm Problem also known as DLP is to find the unique integer i, that is in range from 0 to n-1, such that for given alpha and beta, the below relation satisfies

$$\alpha^i = \beta$$

So, this integer is referred as Discrete log β with respect to α

$$i = log_a \beta$$

Discrete Logarithm is the inverse of exponentiation operation. Exponentiation is easy to compute For exponentiation we just have to do square and multiply operations. We have many available efficient algorithm of finding exponentiation like power trees, fibonnaci method etc. However, if the group is properly chosen, computation of discrete log is believed to be difficult. Thus the exponentiation is a potential one-way problem. Exponentiation is having profound application in public key cryptography.

2 The ElGamal Cryptosystem

Let p be a prime number and let's assume that DLP in $(Z_p^*, .)$ is hard. And let $\alpha \epsilon Z_p^*$ be a primitive element means that if alpha raise to any power that this will lead to non-zero elements in Z_p^*. Let's take plaintext P that belongs to Z_p^* and ciphertext C that belongs to $Z_p^* X Z_p^*$ and let's take key K which takes pramaeters p, α, a and β where

$$\alpha^a \equiv \beta (modp)$$

For a given key K, the parameters p, α and β are publicly known and a is the only secret. So, for decryption we should secret a. So, in encryption we take secret random number r that belongs to $_(p-1)$. The encryption of plaintext x is done according to random number r, and generates encrypted text (y1,y2) where both y1,y1 belongs to Z_p^*.

$$e_k(x,r) = (y_1, y_2)$$

where,

$$y_1 = \alpha^r modp$$

$$y_2 = x\beta^r modp$$

Here, we can see that in y2, beta raise to the power r is used as blinding x by mutliplying it with x. So for decryption, if we get know the value of beta raise to the power then by multiplying its inverse with y2 we get x. The value of beta raise to the power r can be obtained from $y_1 = \alpha^r$ by using the secret value a, as

$$\beta^r = (\alpha^a)^r$$

$$\beta^r = (\alpha^r)^a$$

$$\beta^r = y_1^a$$

So, for y1, y2 ϵZ_p^*, we define decryption as

$$d_k(y_1, y_2) = y_2(y_1^a)^(-1)$$

In semantic security, cyphertext should not reveal any information about the plaintext. It is like a game. To understand this game, let's take two entities challenger and adversarY. The adversary send two messages m0, m1 to challenger and ask the challenger to ancrypt any one of them. Then challenger choose a random binary number b $\epsilon 0, 1$ and generates cyphertext $C = E_k(m_b)$ and send this to adversary. Then the adversary guess guess b' and sends to challenger and if this b' matches to b and adversary wins. So, the probability of matching is

$$Pr(b = b') \leq 1/2 + negligible(n)$$

where negligible(n) is a semantic security parameter. From this game we can understand that if challenger not did the encryption by randomly choosing b then it would become very easy for adversary to guess b by observing previous values of C because then encryption becomes deterministic. So, to make adversary difficult to guess b, we need to make encryption randomized. That is why randomized encryption is better than deterministic.
ElGamal cryptosystem is also randomized because we did encryption based on random number r. The ciphertext depends on both the plaintext x and the random value r chosen by Alice (that is encryptor). The same plaintext can be mapped into p-1 ciphertext depending on the choice of r. For example

Let's take p = 2579, $\alpha = 2$(primitive element of Z_p^* and secret value, a = 765, then

$$\beta = 2^7 65 mod 2579 = 949$$

Suppose, alice wishes to send x = 1299 to Bob. She randomly chooses r = 853.

$$y_1 = 2^8 53 mod 2579 = 435$$

$$y_2 = 1299(949^8 53) mod 2579 = 2396$$

Alice sends y = (435, 2396). Bob computes

$$x = 2396(435^7 65)^(- 1) mod 2579 = 1299$$

3 Algorithm for the DLP Problem

If α^i was monotonically non decreasing with i, we could have done a binary search to find i. But the problem with modular exponentiation is that there is no ordering of the powers. Thus one have to do an exhaustive search in the worst case. Thus it can be solved in O(n) time and O(1) space. However pre-computation helps. // Suppose we store all possible values of α^i (mod p) as ordered pairs $(i, \alpha^i modp)$ and sort the elements with respect to the second parameter. Now search for the given challenge by employing binary search. Complexity of pre-computation is O(n), of memory is O(n), of time to sort is O(nlogn) using a good sorting algorithm and of time to search is O(log n). Often we neglect the log n terms in these algorithms, as n is much larger than log n. Thus time to search O(1) and pre-computation or memory both are O(n).

4 Shank's Baby Step Giant Step Algorithm

Shanks(G,n,α, β)

1. $m = leastintegerfunction(\sqrt{n})$
2. for j = 0 to m-1, compute $\alpha^(mj)$
3. Sort the m ordered pairs (j, $\alpha^(mj)$) with respect to their second coordinates, obtaining the list L1.
4. for i = 0 to m-1, compute $\beta\alpha^(- 1)$
5. Sort the m ordered pairs (i, $\beta\alpha^(- 1)$) with respect to their second coordinates, obtaining the list L2.
6. Find a pair (j,y) ϵL_1 and $(i, y)\epsilon L_2$ (i.e find two pairs having identical second coordinates)
7. Compute, $log_\alpha \beta = (mj + i)$

In above algorithm, Steps 2 and 3 can be pre-computed. If Step 6 is successful then

$$\alpha^(mj) = y = \beta\alpha^(- 1) => \alpha^(mj + 1) = \beta$$

If $\beta\epsilon < a >, log_\alpha\beta = (mj + i)$, where both i, j are from 0 to m-1. The search is successful as we can ensure that $log_\alpha\beta \leq m(m-1) + (m-) = n - 1$ as expected. The algorithm runs in O(m) time with O(m) menory (neglecting logarithmic factors).
For application of the DH key agreement scheme, we can g and p public. Taking exponent a as Alice's secret and exponent b as Bob's secret. Then alice sends $g^a modp$ to bod and bob sends $g^b modp$ to alice. Alice computes $(g^b)^a = g^(ba) = g^(ab) modp$. Bob computes $(g^a)^b = g^(ab) modp$. He could use K = $g^(ab) modp$ as symetric Key.
It is subject to man in the middle (MIM) attack. When Alice send $g^a modp$ to Trudy then Trudy whose secret exponent is t, sends $g^t modp$ to Bob. Then bob sends $g^b modp$ to Trudy and then Trudy sends $g^t modp$ to Alice. IN this, Trudy shares secret $g^(at) modp$ Alice. And Trudy shares secret $g^(bt) modp$ with bob. Alice and bob don't know Trudy exists.MIM attacks can be prevent by encrypting DH exchange with symmetric key. Encrypt DH exchange with public key. Sign DH values with private key.