[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2022-2023
Scribed by: Akshay (202051018) — Lecture (Week 13)

1 ElGamal Public Key Cryptosystem

1. Select a prime p.
2. Consider the group ($\mathbb{Z}_p^*$, . mod p) (* means excluding 0 from $\mathbb{Z}_p$)
3. Select a primitive element $\alpha \in \mathbb{Z}_p^*$ (generator)

   $\mathbb{Z}_p^* \rightarrow$ Cyclic group

   $\mathbb{Z}_p^* = <\alpha>$
4. Plaintext space = $\mathbb{Z}_p^*$
   Keyspace = $\{(p, \alpha, a, \beta) \mid \beta = \alpha^a \ mod \ p\}$
5. Public Key: p, α, β
   Secret Key: a
   $\beta = \alpha^a$, Given β, α finding a will be hard (Discrete log problem) .
6. Select a random number $x \in \mathbb{Z}_{p-1}$, x is kept secret
7. <u>Encryption</u>
   $e_K(m, x) = (y_1, y_2)$ (m: message)
   $y_1 = \alpha^x \ mod \ p$
   $y_2 = m.\beta^x \ mod \ p$
8. <u>Decryption</u>
   $d_K(y_1, y_2) = y_2(y_1^a)^{-1} \ mod \ p$

   $$\begin{aligned} y_1^a &= (\alpha^x)^a \ mod \ p \\ &= (\alpha^a)^x \ mod \ p \\ &= \beta^x \ mod \ p \end{aligned}$$

   $$\begin{aligned} y_2.(y_1^a)^{-1} &= (m.\beta^x).(\beta^x)^{-1} \ mod \ p \\ &= m \ mod \ p \end{aligned}$$

Security of ElGamal Cryptosystem depends on two problems:

1) **Discrete log problem** is hard
$\beta = \alpha^a$, Given β, α finding a will be hard

2) **Diffie Hellman problem** is hard
Knowns: g, g^a, g^b

Not knowns: a, b
Finding g^{ab}
Because security can be broken if we are able to find α^{ax} from knowns i.e. $(\alpha, \alpha^a = \beta, \alpha^x = y_1)$ as we just need to multiply inverse of this with y_2 to reveal message (m).

2 Discrete Log Problem

Given: Finite cyclic group G of order n, generator α of G, element $\beta \in$ G.

Find: Integer x , $0 \leq x \leq n-1$ such that $\alpha^x = \beta$

Exhaustive search (O(n)) is inefficient.

Baby-Step Giant-Step Algorithm:
Time complexity: $O(\sqrt{n})$

$m = \lceil \sqrt{n} \rceil \qquad \alpha^n = 1$
If $\beta = \alpha^x$ then we can write:
$x = i.m + j$ (through Division algo)
m: divisor, i: quotient, j: remainder
$0 \leq i, j \leq m$

$\alpha^x = \alpha^{im}.\alpha^j$
$\beta = \alpha^{im}.\alpha^j$
$\implies \alpha^j = \beta.(\alpha^{im})^{-1}$
$\alpha^j = \beta.(\alpha^{-m})^i$

For x, we need to find unique i,j which will satisfy above equation.
Now, instead of x, target is to find i and j. Size of space of i,j is strictly less than m ($=\lceil \sqrt{n} \rceil$)
Now, aim is to find i and j in such a way that complexities don't get multiplied.

- Compute each value of j and corresponding α^j. Store it in a table in a sorted manner.
- For each i :
  - Compute $\beta.(\alpha^{-m})^i$
  - get corresponding j by subtracting $i.m$ from x
  - get corresponding α^j from the look-up table
  - Compare α^j with $\beta.(\alpha^{-m})^i$ for solution.

Formal Presentation of this algorithm:

Input: generator α of a cyclic group G, ord(G) = n, $\beta \; \in$ G.

Output: the discrete log, $x = log_x \beta$

1. Set $m \leftarrow \lceil \sqrt{n} \rceil$
2. Prepare a table T with entries $(j, \, \alpha^j) \qquad 0 \leq j < m$

(a) Sort T by second component

3. Compute α^{-m} and set $\gamma \leftarrow \beta$
4. For i=0 to m-1 do:
   (a) check if γ is second of some entry in T.
   (b) If $\gamma = \alpha^j$ then we got the solution. Return
   (c) Set $\gamma \leftarrow \gamma.\alpha^{-m}$

Storage: $O(\sqrt{n})$
Number of multiplications: $O(\sqrt{n})$
Sort : $O(\sqrt{n}.log\sqrt{n}) = (\sqrt{n}.logn)$

3 Kerberos (Version 4) (User Authentication Protocol)

Didn't get clarity in this topic. I will come back to it later.