Secure Computation: Part II

Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 59
Cyclic Groups and Discrete Logarithm

(Refer Slide Time: 00:24)

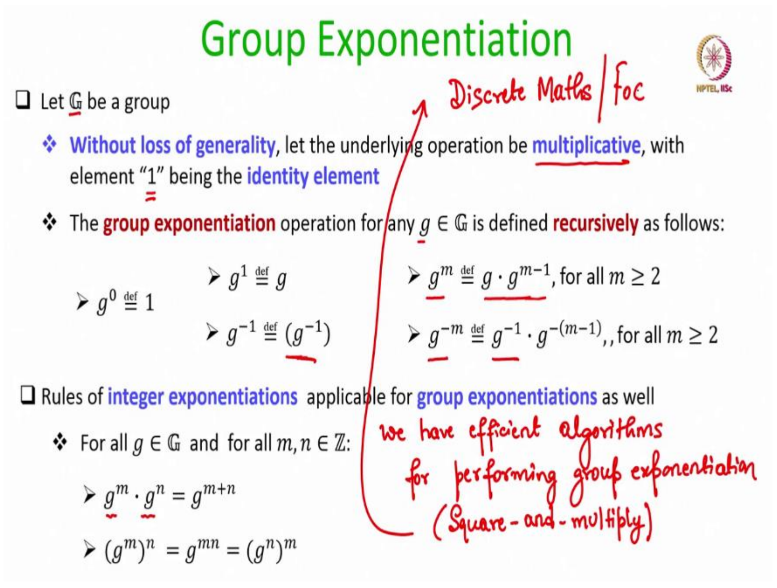
Lecture Overview

- Cyclic groups
- Discrete logarithm

Hello everyone, welcome to this lecture. So, now, for the next few lectures I will be discussing cryptographically secure multi party computation. So, there are several ways to design cryptographically secure multi party computation based on the underlying primitives. We can base the protocol on threshold encryption schemes we can base the protocol on fully homomorphic encryption schemes we can base the protocol on garbling schemes and so on.

But I will be discussing a simpler approach for designing cryptographically secure multi party computation based on commitment schemes. So, to understand the commitment schemes we have to quickly go through cyclic groups and the discrete logarithm problem.

(Refer Slide Time: 01:10)



Group Exponentiation

Discrete Maths | FoC

- Let $\mathbb{G}$ be a group
  - Without loss of generality, let the underlying operation be multiplicative, with element "1" being the identity element
  - The group exponentiation operation for any $g \in \mathbb{G}$ is defined recursively as follows:
    - $g^0 \overset{\text{def}}{=} 1$
    - $g^1 \overset{\text{def}}{=} g$
    - $g^{-1} \overset{\text{def}}{=} (g^{-1})$
    - $g^m \overset{\text{def}}{=} g \cdot g^{m-1}$, for all $m \geq 2$
    - $g^{-m} \overset{\text{def}}{=} g^{-1} \cdot g^{-(m-1)}$, , for all $m \geq 2$
- Rules of integer exponentiations applicable for group exponentiations as well
  - For all $g \in \mathbb{G}$ and for all $m, n \in \mathbb{Z}$:
    - $g^m \cdot g^n = g^{m+n}$
    - $(g^m)^n = g^{mn} = (g^n)^m$

we have efficient algorithms for performing group exponentiation (Square-and-multiply)

So, let us start with the group exponentiation operation. So, let $\mathbb{G}$ be a group and without loss of generality I assume that the underlying operation is multiplicative. But it need not be the case, but whatever I am discussing here holds even if the underlying group operation is additive. So, since I am considering multiplicative group, I will be denoting the identity element by the notation 1. So, this should not be confused with the integer 1.

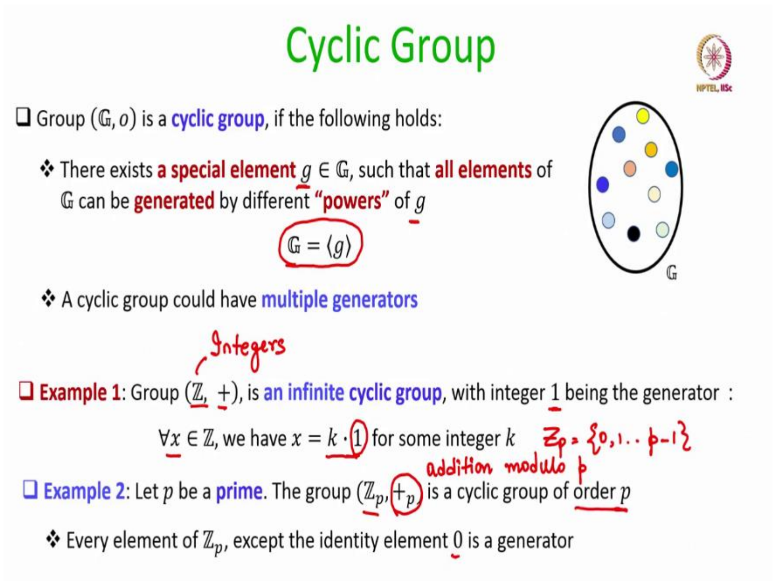
Now the group exponentiation operation for an element with respect to an element g is defined recursively as follows. We define g^0 to be the identity element of the group g^1 is defined to be the group element g itself. And now g^m for any $m \geq 2$ is defined to be the result of the group operation being performed over the elements g and g^{m-1}.

With respect to the negative powers g^{-1} is defined to be the inverse of the element g and g^{-m} for any $m \geq 2$ is defined to be the result of the group operation being performed over the group elements g^{-1} and $g^{-(m-1)}$. So, it is easy to see that the rules the way we have defined the group exponentiation the rules of integer exponentiations are applicable even for a group exponentiation.

Namely if I multi if I apply the group operation on the elements g^m and g^n then I will obtain the group element g^{m+n} and so on. I also would like to stress here that we have efficient algorithms.

We have efficient algorithms for performing group exponentiations. So, for instance we can use the square and multiply approach which is a very nice algorithm square and multiply algorithm. Its description you can find in any standard text on number theory, or you can refer to my course on discrete mathematics the NPTEL course on discrete mathematics or foundations of cryptography.

(Refer Slide Time: 04:18)



Cyclic Group

- Group $(\mathbb{G}, o)$ is a **cyclic group**, if the following holds:
  - There exists **a special element** $g \in \mathbb{G}$, such that **all elements** of $\mathbb{G}$ can be **generated** by different **"powers"** of g

    $$\boxed{\mathbb{G} = \langle g \rangle}$$

  - A cyclic group could have **multiple generators**
- **Example 1**: Group $(\mathbb{Z}, +)$, is **an infinite cyclic group**, with integer 1 being the generator :

  $\forall x \in \mathbb{Z}$, we have $x = k \cdot \boxed{1}$ for some integer k $\mathbb{Z}_p = \{0, 1 .. p-1\}$
- **Example 2**: Let p be a **prime**. The group $(\mathbb{Z}_p, \boxed{+_p})$ is a cyclic group of order p
  - Every element of $\mathbb{Z}_p$, except the identity element 0 is a generator

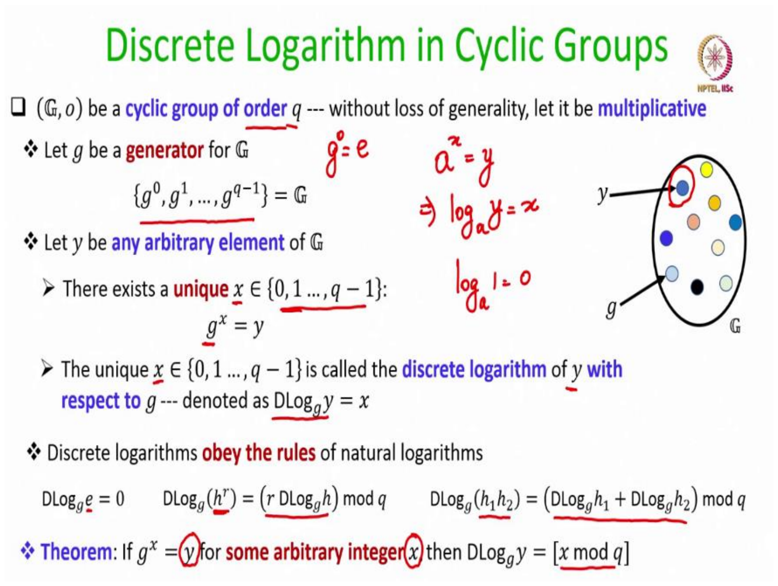
Now, let us define cyclic groups. So, a group is called a cyclic group if you have a special element g in the group which is called as the generator. And the specialty of this element is that you can generate all the elements of the group by computing different powers of the generator. So, in terms of notation we use this represent notation here within the angular bracket we write the generator which denotes that if I take the different powers of g starting from 0 and 0 onwards I will be able to generate all the elements of the group.

So, it is not necessary that a cyclic group has a unique generator, it could have multiple generators. The definition for cyclic group is that it should have at least 1 generator. So, let us see some examples of cyclic groups. If I take the set of integers with respect to the integer addition operation, then it constitutes a cyclic group where the element 1 will be the generator. Because if you take any integer x which is negative or positive, you can express that integer x as some k times the generator 1.

That means in terms of the generator you can represent each and every integer whereas, if you take a prime number p then the set $\mathbb{Z}_p$ which is the set of elements $\{0, 1 \dots, p-1\}$

then this set along with the operation addition modulo p. So, $+_p$ is the operation of addition modulo p, it also constitutes a cyclic group of prime order. Prime order means the number of elements here will be a prime quantity namely p and then if we have a cyclic group of prime order then a nice feature of such groups is that every element except the identity element 0 will be a generator of this cyclic group.

(Refer Slide Time: 06:45)



Discrete Logarithm in Cyclic Groups

- ❑ $(\mathbb{G}, o)$ be a cyclic group of order q --- without loss of generality, let it be multiplicative
  - ❖ Let g be a generator for $\mathbb{G}$
    $\{g^0, g^1, \dots, g^{q-1}\} = \mathbb{G}$
  - ❖ Let y be any arbitrary element of $\mathbb{G}$
    - ➢ There exists a unique $x \in \{0, 1 \dots, q-1\}$:
      $g^x = y$
    - ➢ The unique $x \in \{0, 1 \dots, q-1\}$ is called the discrete logarithm of y with respect to g --- denoted as $\text{DLog}_g y = x$
  - ❖ Discrete logarithms obey the rules of natural logarithms
    $\text{DLog}_g e = 0$ $\quad$ $\text{DLog}_g(h^r) = (r\,\text{DLog}_g h) \bmod q$ $\quad$ $\text{DLog}_g(h_1 h_2) = (\text{DLog}_g h_1 + \text{DLog}_g h_2) \bmod q$
  - ❖ Theorem: If $g^x = y$ for some arbitrary integer x then $\text{DLog}_g y = [x \bmod q]$

$g^0 = e$

$a^x = y$

$\Rightarrow \log_a y = x$

$\log_a 1 = 0$

Next let us define the discrete logarithms in the context of cyclic groups. So, imagine you have a cyclic group whose order is q; that means, there are q number of elements in the group and again without loss of generality imagine that the group is multiplicative. Since the group is a cyclic group; that means, by computing the different powers of 0 starting from 0 up to $q - 1$ I can generate all the elements of the group.

Now, suppose I take any arbitrary element of the group. Since it is a group element it can be expressed as a power of the generator. So, there will be some unique power x in the range 0 to $q - 1$ such that g^x is going to be that arbitrary element y then this unique power x in the range 0 to $q - 1$ is considered as the discrete logarithm of y with respect to the generator and we denote it by $DLog_g y$.

So, it is very similar to our natural logarithm in the for the natural logarithms. We know that if $a^x = y$ then we say that $\log_a y$ is x. You can imagine that discrete logarithm is similar to our natural logarithm in the discrete world in the context of a cyclic world. Like natural logarithms discrete logarithms also obey some nice properties.

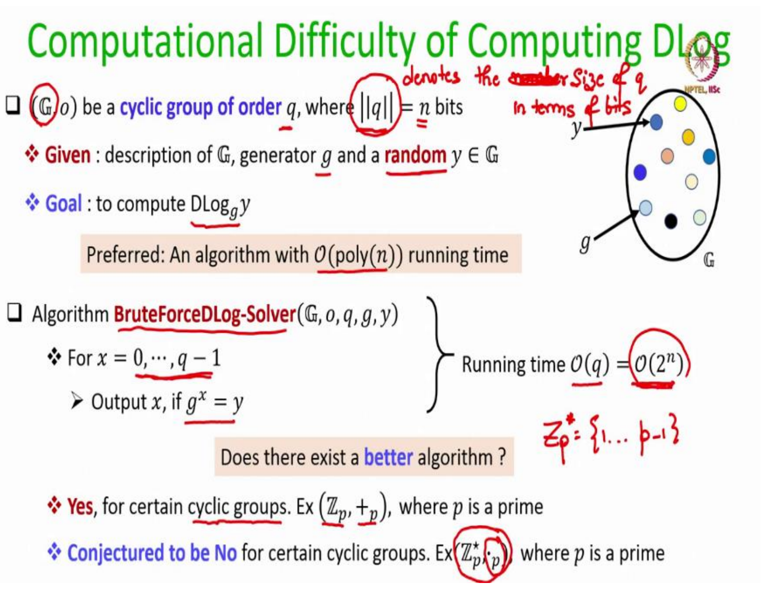
So, we know that $\log_a 1$ is 0 because a^0 is defined to be 1. In the same way the discrete logarithm of the identity element of the group with respect to the generator is 0 because remember that we have defined g^0 to be the identity element of the group. In the same way if we take an element h and raise it to the power r it will be a group element because the group satisfies the closure property.

Now, if we want to take if you want to compute the discrete logarithm of this new element h to the power r that will be same as r times the discrete logarithm of h. Now r times discrete logarithm of h might be a value which crosses $q - 1$. So, to ensure that the resultant value is in the range 0 to $q - 1$ we must take a mod q. Because as per the definition of discrete log it is the unique power x in the range 0 to $q - 1$.

In the same way if you want to compute the discrete logarithm of the product of two elements. Then it will be same as the summation of the discrete logarithms of the individual elements. In general if you are given an arbitrary element y such that $y = g^x$ for any integer x may not be in the range 0 to $q - 1$, then a discrete logarithm of y in the range 0 to $q - 1$ can be obtained by computing x modulo q.

So, these are some standard properties of discrete logarithms. I am not going into the proof for these properties I will be telling you the references which you can follow in case you want to go through the proof of these properties.

(Refer Slide Time: 10:23)



Computational Difficulty of Computing DLog

- $(\mathbb{G}, o)$ be a cyclic group of order q, where $||q|| = n$ bits
  - **Given** : description of $\mathbb{G}$, generator g and a random $y \in \mathbb{G}$
  - **Goal** : to compute $\mathrm{DLog}_g y$

Preferred: An algorithm with $\mathcal{O}(\mathrm{poly}(n))$ running time

- Algorithm **BruteForceDLog-Solver**$(\mathbb{G}, o, q, g, y)$
  - For $x = 0, \cdots, q - 1$
    - Output x, if $g^x = y$

Running time $\mathcal{O}(q) = \mathcal{O}(2^n)$

Does there exist a better algorithm ?

- **Yes**, for certain cyclic groups. Ex $(\mathbb{Z}_p, +_p)$, where p is a prime
- **Conjectured to be No** for certain cyclic groups. Ex $(\mathbb{Z}_p^\star, \cdot_p)$ where p is a prime

denotes the ~~number~~ size of q in terms of bits

$\mathbb{Z}_p^* = \{1 \ldots p-1\}$

Now, how difficult or easy is it to compute the discrete log of a arbitrary given number? So, imagine you are given a cyclic group whose order is q such that the number of bits required to represent the value q is n. So, this notation denotes the size of q in terms of bits. That means, q is an n bit number that does not mean there are n elements in the group there are q elements in a group g. Where q is represented by n bits so; that means, the magnitude of q is roughly 2^n; that means, my group size is exponentially large here.

Now, imagine I give you a generator and a random element from the group I stress a random element it is not a fixed element of the group. Now since my group g is a cyclic group and g is a generator there will be a discrete logarithm of y. So, I would like to compute a discrete logarithm of y with respect to the generator and I prefer an algorithm where the number of operations is polynomial in n namely the size of group elements.

So, there is of course, a brute force discrete log solver and that brute force algorithm will do the following. It will go over every possible value for the discrete logarithm starting from 0 all the way to $q - 1$ and checks whether $g^x = y$ or not. Of course, the discrete logarithm of y will be a value in the range 0 to $q - 1$ and that is precisely this algorithm is exploiting.

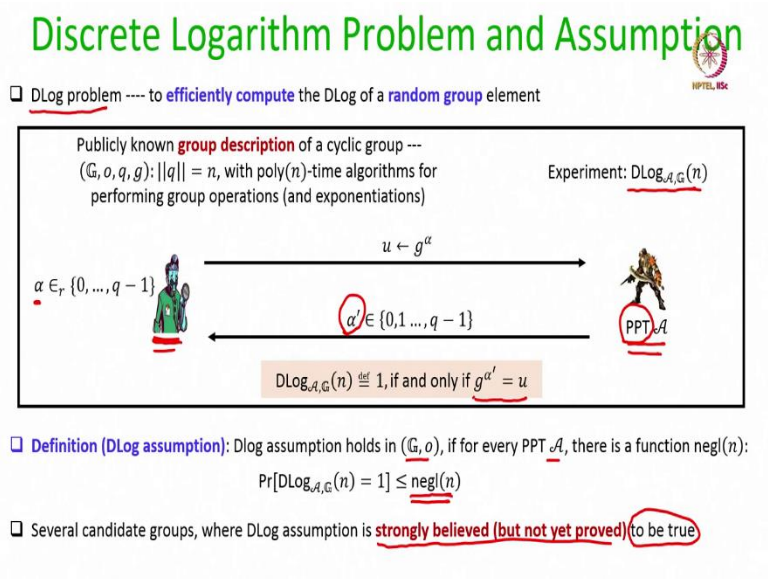
So, this algorithm will give you the output, but what is the running time of this algorithm? The running time of this algorithm is order of q. You should not say that this is a polynomial time algorithm because q in terms of magnitude is 2^n. So, the running time of this algorithm increases exponentially as the magnitude of q increases. So, this is definitely not a preferred algorithm.

The question is does there exist a better algorithm than this naive brute force discrete log solver the answer is yes, but not always. So, there are certain cyclic groups where we can get where we can use better algorithms in fact, efficient algorithms to compute the discrete logarithm for our random number without doing the brute force. So, for instance if I take the cyclic group $\mathbb{Z}_p$ where the operation is addition modulo p then we have better algorithms.

But at the same time there are certain candidate cyclic groups where, as of today, we do not have any algorithm more efficient than this brute force algorithm. Or even if we have alternate algorithms, their worst case running time ends up to be asymptotically order of 2^n or order of $|\mathbb{G}|$.

So, for instance one such candidate cyclic group is the group $\mathbb{Z}_p^\star$. So, $\mathbb{Z}_p^\star$ is the set 1 to $p-1$ and the operation here multiplication modulo p. Of course, there are some other better candidate cyclic groups for which it is conjectured that we do not have any better algorithm whose running time is better than this order of 2^n.

(Refer Slide Time: 14:51)



Discrete Logarithm Problem and Assumption

- DLog problem ---- to **efficiently compute** the DLog of a **random group** element

Publicly known **group description** of a cyclic group --- $(\mathbb{G}, o, q, g)$: $||q|| = n$, with poly(n)-time algorithms for performing group operations (and exponentiations)

Experiment: $\text{DLog}_{\mathcal{A},\mathbb{G}}(n)$

$\alpha \in_r \{0, \dots, q-1\}$

$u \leftarrow g^\alpha$

$\alpha' \in \{0,1 \dots, q-1\}$

PPT $\mathcal{A}$

$\text{DLog}_{\mathcal{A},\mathbb{G}}(n) \overset{\text{def}}{=} 1$, if and only if $g^{\alpha'} = u$

- **Definition (DLog assumption)**: Dlog assumption holds in $(\mathbb{G}, o)$, if for every PPT $\mathcal{A}$, there is a function negl(n):

$$\Pr[\text{DLog}_{\mathcal{A},\mathbb{G}}(n) = 1] \leq \text{negl}(n)$$

- Several candidate groups, where DLog assumption is **strongly believed (but not yet proved)** to be true

So, based on this discussion we now formulate the discrete logarithm problem and the $DLog$ assumption. So, an instance of $DLog$ problem is basically to efficiently compute the discrete logarithm of a given random group element and the difficulty of the $DLog$ problem the difficulty of solving the $DLog$ problem is formalized by an experiment. We call that experiment as the $DLog$ experiment which is parameterized with respect to an adversary and a group $\mathbb{G}$; where the size of the group is exponentially large in n. And the problem instance is created as follows.

So, we have two entities here the experiment where which is modeled by some hypothetical verifier and an adversary. Now this experiment or the verifier it picks a random index in the range 0 to $q-1$. It can be done easily and then it gives the group element g^α to the adversary. So, the discrete logarithm of this element g^α is α which is known to this experiment. Now the challenge for this adversary is to compute a discrete logarithm for this value u.

Notice that the value u is going to be a random element in the group now the adversary is allowed only polynomial time here. So, PPT here denotes probabilistic polynomial time. So, adversary simply cannot run the brute force algorithm which we had discussed earlier. It is free to use any other algorithm as long as its running time is polynomial time and in polynomial time it has to submit a response to the experiment, we say that the adversary has won the experiment, or the output of the experiment is 1.

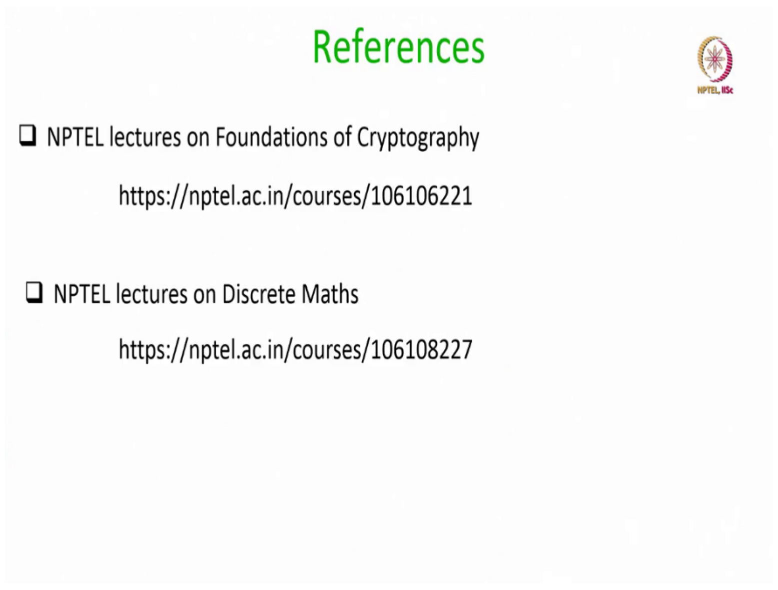
So, the output of the experiment being 1 denotes that adversary has won the experiment. And adversary has won the experiment if and only if it has computed the discrete logarithm correctly. That means, whatever is the response submitted by the adversary g to the power that response is equal to the random element which the experiment has chosen.

Now, what is the $DLog$ assumption? Informally $DLog$ assumption means that it is difficult for any adversary to win an instance of $DLog$ modeled by the above experiment in that group. Formally we say that the discrete log assumption holds in the group if for every polynomial time adversary who participates in the above experiment. The probability that it can when the experiment is upper bounded by some negligible quantity; that means, except with a negligible probability the adversary will fail to win the experiment.

We are giving here some negligible advantage to the adversary to win the experiment. Because a simple strategy for the adversary to win the experiment could be to just guess the value of the discrete logarithm. For that adversary does not have to do any sophisticated task it just has to guess a value α' and there is always a non 0 probability that the alpha prime guessed by the adversary turns out to be the discrete logarithm.

It turns out that there are several candidate cyclic groups where discrete logarithm is conjectured to be difficult. That means, $DLog$ assumption holds in those groups, but again it is just a conjecture we do not have any formal proof. As of now even after trying for several years as of today we do not have any efficient algorithm to solve an instance random instance of discrete log problems in those groups.

(Refer Slide Time: 19:10)


References

- NPTEL lectures on Foundations of Cryptography

  https://nptel.ac.in/courses/106106221

- NPTEL lectures on Discrete Maths

  https://nptel.ac.in/courses/106108227

So, with that I end this lecture. So, these are the references which you can follow to know more about discrete logarithm cyclic groups and so on. You can either refer to my NPTEL lectures on foundations of cryptography or my NPTEL lectures on discrete mathematics.

Thank you.