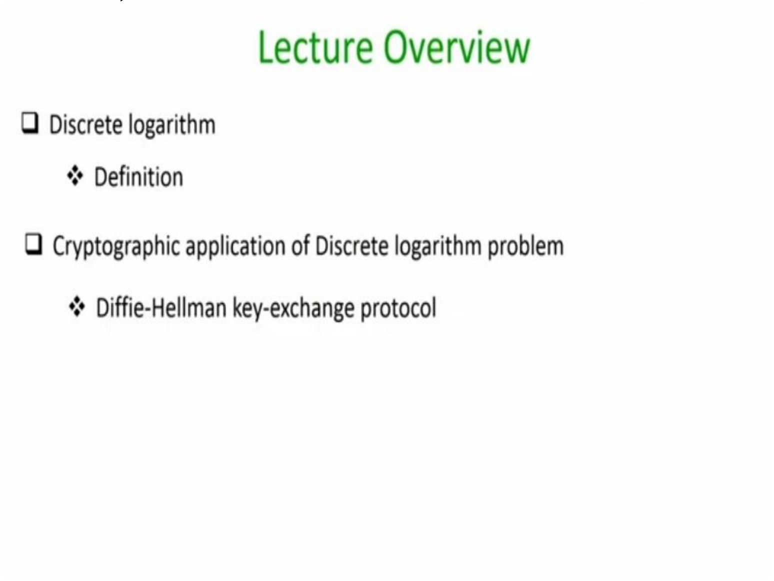
Discrete Mathematics
Prof. Ashish Choudhury
International Institute of Information Technology, Bangalore

Lecture - 65
More Applications of Groups

Hello, everyone, welcome to this lecture. So, till now we have discussed a lot of theory regard in the context of number theory and abstract algebra. In the next couple of lectures we will see how to tie whatever we have learned till now in number theory and abstract algebra and see some concrete applications in the context of cryptography.

(Refer Slide Time: 00:42)

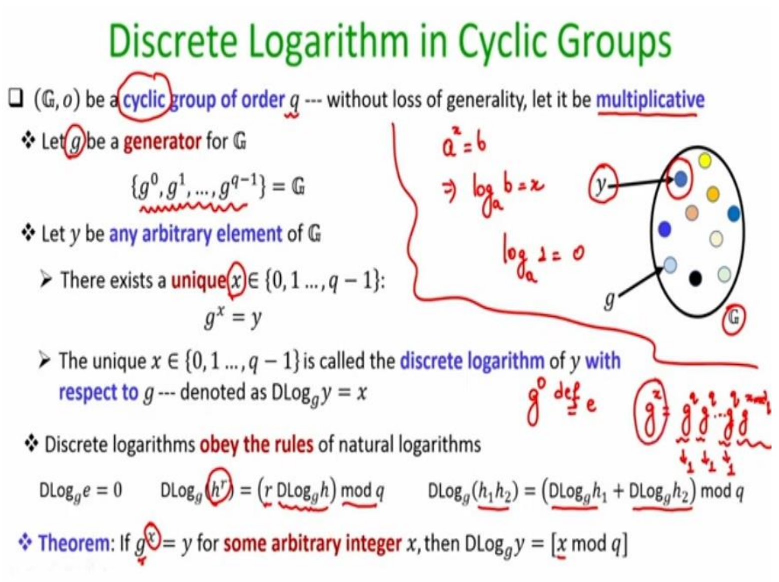


Lecture Overview

- Discrete logarithm
  - Definition
- Cryptographic application of Discrete logarithm problem
  - Diffie-Hellman key-exchange protocol

So, the plan for this lecture is as follows. In this lecture, we will introduce this concept of discrete logarithm and the discrete logarithm problem. And we will see some cryptographic applications of the discrete logarithm problem namely the seminal key exchange protocol due to Diffie and Hellman.

(Refer Slide Time: 01:01)



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 $\mathbb{G}$

$$\underbrace{\{g^0, g^1, \ldots, g^{q-1}\}} = \mathbb{G}$$

  - ❖ Let y be any arbitrary element of $\mathbb{G}$
    - ➢ There exists a unique $x \in \{0, 1 \ldots, q-1\}$:

$$g^x = y$$

    - ➢ The unique $x \in \{0, 1 \ldots, q-1\}$ is called the discrete logarithm of y with respect to g --- denoted as $\mathrm{DLog}_g y = x$
  - ❖ Discrete logarithms obey the rules of natural logarithms

$\mathrm{DLog}_g e = 0$ $\quad$ $\mathrm{DLog}_g (h^r) = (r\,\mathrm{DLog}_g h) \bmod q$ $\quad$ $\mathrm{DLog}_g (h_1 h_2) = (\mathrm{DLog}_g h_1 + \mathrm{DLog}_g h_2) \bmod q$

  - ❖ Theorem: If $g^x = y$ for some arbitrary integer x, then $\mathrm{DLog}_g y = [x \bmod q]$

$a^x = b$

$\Rightarrow \log_a b = x$

$\log_a 1 = 0$

$g^0 \overset{\text{def}}{=} e$

So, let us start with the discrete logarithm definition in the context of cyclic groups. So, let G be a cyclic group and that abstract operation is o and suppose the order of the cyclic group is q that means, you have q number of elements and for simplicity and without loss of generality, I will follow that multiplicative interpretation, while giving the definition of discrete logarithm, but the definition can be easily generalized even when the underlying operation is interpreted in the additive sense.

So, since my group G is a cyclic group it must be having a generator, so, let the g be generator and since the order of the group is q, has q number of elements that means, by raising or by computing q different powers of the generator, I can obtain all the elements of my group. Now, consider an arbitrary element y from the group, since the element y is a member of the group, it can be generated by some power of your generator.

That unique power in the range 0 to q - 1 which when raised to the generator gives you the element y, will be called as the discrete logarithm of the y to the base g. That is the definition of my discrete logarithm. So, in some sense, it is equivalent to our definition of natural logarithms. So, we know that if a to the power x = b then we say that log of b to the base a is x, we are trying to come up with an equivalent definition in the context of a cyclic group.

So, g is a special element the generator because, if I keep the generator to the base and compute different powers of the generator, I can obtain all the elements of my group that means, I can say that you give me any element of the group there must be some power of the generator such that that power x of the generator gives me the element y, that unique power x in the range 0 to q - 1 is called as the discrete logarithm.

And interestingly, like the natural logarithms, your discrete logarithm also obeys the rules that are there in the context of natural logarithms. For instance, we know that log of 1 to the base of any a is 0 because a to the power 0 is defined to be 1 in for natural logarithms. In the context of discrete logarithms, we say that the discrete log of the identity element of the group to the base of g will be 0.

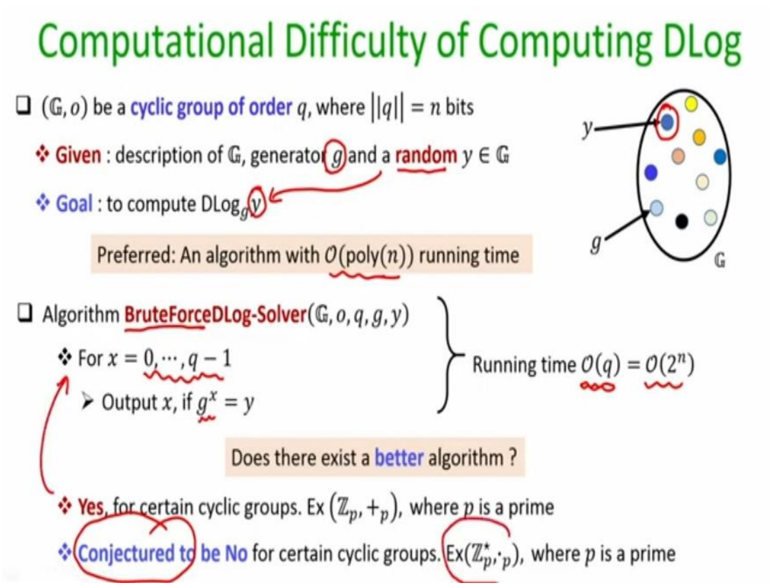
Because, remember, as per the rules of group exponentiation, we have defined g^0 to be the identity element. In the same way, if I take an element h from the group and compute the element h to the power r which will be also a group element and now if I try to find out the discrete logarithm of the group element h to the power r to the base g then it will be same as r multiplied with the discrete logarithm of h to the base g.

And if this value is in the range 0 to q - 1 well and good, otherwise, I take a mod and bring down the value to the range 0 to q - 1. In the same way, if I have 2 group elements h_1 and h_2 then the discrete log of the product of h_1 and h_2 will be same as the summation of the discrete logs of h_1 and h_2 individually modulo q. And the general theorem statement that we can have is the following.

If you are given that $g^x = y$ where g is the generator then, either x will be the discrete log of y if x is within the range 0 to q - 1, else if x is greater than q then, if I take x modulo q then the resultant value will be in the range 0 to q - 1 and that will be the discrete logarithm of y to the base g. This is because if x is indeed greater than q then I can rewrite g^x as several blocks of g^q, g^q, g^q and the last block consisting of $g^{x \; mod \; q}$.

And each of these blocks of g^q will give me the identity element because g is the generator and its order will be q, hence g^q will be identity element and the last block will be $g^{x \; mod \; q}$.so, I get $g^{x \; mod \; q}$. So, if $g^x = y$, so, is $g^{x \; mod \; q}$ and $x \; mod \; q$ will be a value in the range 0 to q - 1 and hence it will be the discrete logarithm.

(Refer Slide Time: 06:17)



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 $O(\mathrm{poly}(n))$ 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 $O(q) = O(2^n)$

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$(\mathbb{Z}_p^*, \cdot_p)$, where p is a prime

So, now an interesting problem is that how easy or how difficult it is to compute the discrete logarithm. So, imagine we are given an abstract cyclic group of order q and this notation means that, the number of bits that I need to represent my q is n bits. So, this notation is nothing but the number of bits needed to represent q. That means magnitude wise q is as large as 2^n. Now, let us see how difficult or how easy it is to compute a discrete logarithm.

So, you are given the description of the cyclic group. By the description I mean, you know, the characteristic of the elements of the group, your group might be exponentially large. It might have exponentially large number of elements and you may not have sufficient space and resources to store down all the elements of your group. But you may know the characteristic or the properties of the elements of your group.

And you are given a generator. So, you know that, by computing different powers of that generator, you can generate any element of your group. And what is given to you, a random element y from the group. That is important, a random element. It is not a predetermined or specific element of the group, it is randomly chosen. And a discrete log problem is to compute the discrete log of this randomly chosen y, given that, you only have the y and the generator, you do not have anything else.

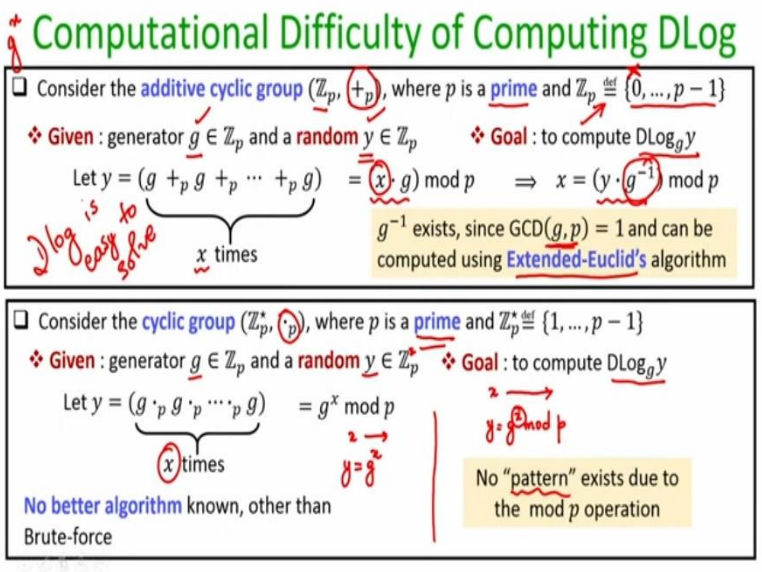
So that means your goal is to come up with a unique power x in the range 0 to q - 1 such that, g^x would have given you y. And what we want to do is, we want to come up with an algorithm whose running time should be polynomial in the number of bits that I need to represent my q, namely, n, I do not need an exponentially large algorithm. So, here is a naive algorithm which will always be successful to give you the discrete log of the randomly chosen y.

I call this algorithm as brute force discrete log solver because it basically does what a naive algorithm will do, you basically try all powers of x in the range 0 to q - 1. And check whether computing g power x gives you the element y or not, if it is then, you output that x and stop the algorithm. And definitely you will hit upon the exact value of x which is the discrete logarithm of y somewhere when you are iterating over all values of x.

So, you will always get the answer. But let us focus on the running time of this algorithm. In the worst case, you may end up performing iteration over all candidate values of x. So, I can say in the worst case, the running time is order of q. But q is not a polynomial quantity in n, it is actually an exponentially large value in the number of bits that I need to represent my value q. So, this algorithm is not a polynomial time algorithm but rather it is an exponential time algorithm.

So now, the next question is does there exist a better algorithm? And the answer is both yes, as well as no. Yes because as we will see soon, there are indeed certain cyclic groups where I can efficiently find out the discrete logarithm of any randomly chosen y, without doing the brute force. But at the same time, we do have some candidate cyclic groups where it is for which it is conjectured that we do not have any better algorithm other than brute forcing over all candidate values of x. So, this is a instance of one such cyclic group.

(Refer Slide Time: 10:39)



Computational Difficulty of Computing DLog

- Consider the additive cyclic group $(\mathbb{Z}_p, +_p)$, where p is a prime and $\mathbb{Z}_p \overset{\text{def}}{=} \{0, \dots, p-1\}$
  - Given : generator $g \in \mathbb{Z}_p$ and a random $y \in \mathbb{Z}_p$
  - Goal : to compute $\text{DLog}_g y$

  Let $y = \underbrace{(g +_p g +_p \cdots +_p g)}_{x \text{ times}} = (x \cdot g) \bmod p \Rightarrow x = (y \cdot g^{-1}) \bmod p$

  g^{-1} exists, since $\text{GCD}(g, p) = 1$ and can be computed using Extended-Euclid's algorithm

- Consider the cyclic group $(\mathbb{Z}_p^*, \cdot_p)$, where p is a prime and $\mathbb{Z}_p^* \overset{\text{def}}{=} \{1, \dots, p-1\}$
  - Given : generator $g \in \mathbb{Z}_p$ and a random $y \in \mathbb{Z}_p^*$
  - Goal : to compute $\text{DLog}_g y$

  Let $y = \underbrace{(g \cdot_p g \cdot_p \cdots \cdot_p g)}_{x \text{ times}} = g^x \bmod p$

  No better algorithm known, other than Brute-force

  No "pattern" exists due to the mod p operation

So now, let us see the case where the discrete log can be easily computed. So, I consider the cyclic group $\mathbb{Z}_p$ where $\mathbb{Z}_p$ is the set of all integers modulo p, namely, it has integers 0 to p - 1. And my operation here is addition modulo p. You are given a generator, by the way, since the order of this cyclic group is a prime quantity because it has prime number of elements then, from the results that we know till now, every element of this set $\mathbb{Z}_p$, except the identity element 0, will be a generator.

So, 1 is a generator, 2 is a generator, 3 is a generator, p - 1 is also a generator. So, now let us see an instance of discrete log problem and how efficiently we can solve it. So, you are given the generator, you are given the random y and your goal is to come up with a unique exponent x, such that, g^x would have given you y, namely, you want to compute a discrete log of y.

By the way, the interpretation of g^x here, will be in the additive sense, we are not going to multiply g because our underlying operation here is addition. So, g^x should be interpreted as $x \times g$. So, imagine x is the discrete logarithm of y and that is the case then, y is nothing, but g added to itself modulo p, x number of times and our goal is to find out what exactly is x.

So, I know that y is nothing but x times g modulo p because g added to itself modulo p is equivalent to saying that, I multiply x with g and then take mod p. And my goal is to find out this unknown x. Now, it is easy to see that this unknown x is nothing but the product of y with the multiplicative inverse of g modulo p. So, this g inverse is now not the additive inverse. This is now the multiplicative inverse of g modulo p.

And now you might be wondering whether the multiplicative inverse of g modulo p exists or not. Indeed, it exists because the generator g is an element in the range 0 to p - 1 and it is co-prime to p. In fact, all the elements 0, 1, 2 up to p - 1 are co-prime to p because your p is a prime. So, the generator g is also co-prime to p. And then, since the generator is co-prime to p, we know that its multiplicative inverse modulo p exist which we can easily find out using Euclids algorithm.

So, now you can see that here, I do not need to do a brute force, I do not need to check whether x = 0 satisfies relation y = x times d modulo p or not, I do not need to check for x = 1, x = 2 and all the way x = p – 1. I do not need to do that because I know that, just by multiplying y with the multiplicative inverse of g, I can hit upon that right x. So, this is an instance of a cyclic group where D log is easy, very easy to solve.

Now, let us see another cyclic group where it is conjectured that discrete log problem is really difficult to solve for a random instance. So, consider the cyclic group $\mathbb{Z}_{p^*}$ where $\mathbb{Z}_{p^*}$, will have all the integers which are relatively prime to p and p itself is a prime number. So, if that is the case then $\mathbb{Z}_{p^*}$ will have all the numbers except 0 from $\mathbb{Z}_p$. And my operation is now multiplication modulo p in that group.

So, we know that this is a cyclic group. And, in fact, later we will prove it concretely that this group is indeed a cyclic group, but for the moment, you have to believe me that this group is indeed a cyclic group. So, now let us see a random instance of the D log problem in this group. So, you are given a generator, you are given a random value y and your goal is to compute a discrete log of y.

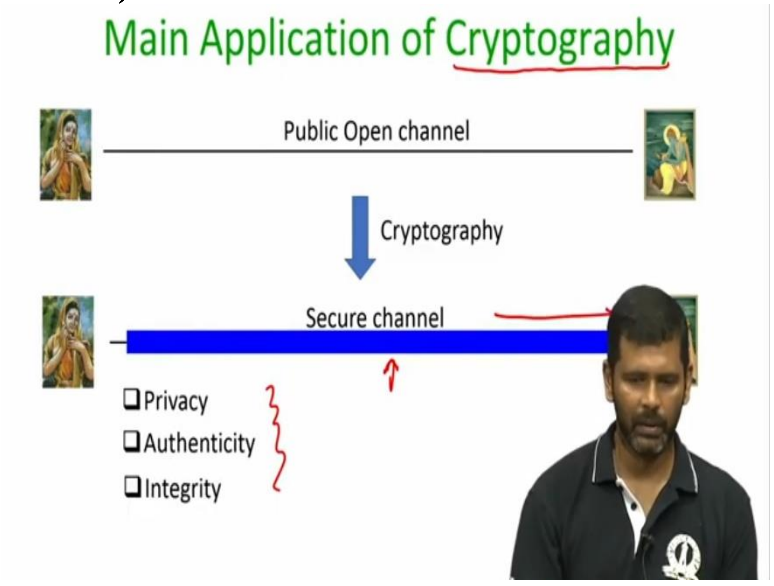
So, this should be $\mathbb{Z}_{p^*}$ because y is an element of your group and the group that we are right now considering is $\mathbb{Z}_{p^*}$. So, imagine $y = g^x mod\, p$. And it turns out that we do not have any better algorithm to compute x, other than naive brute force algorithm that we have discussed in the last slide. And this is because there is no pattern available because of the mod operation which I am performing. What do I mean by no pattern available?

So, by that, I mean that, if I keep on increasing the value of x and check the value of y where $y = g^x$ to the power x modulo p then it is not necessary, it is not the case that as your value of x increases, the value of y also increases. You compare this with the corresponding function where I do not do a mod p operation. Suppose my function $y = g^x$ then for such a function, I can confidently say that as the value of x increases, the value of y also increases.

But as soon as I redefine my y to be $g^x mod\, p$ then because of the mod p operation, it is no longer the case that, as the value of x increases, the value of y also increases, it will increase, decrease, increase, decrease, increase, decrease and there will be absolutely no pattern in which the values of y increases and then suddenly dips and then suddenly increases and so on.

So, there will be a complete chaos if you plot a chart or a graph between x and ys with respect to the fixed g. And it turns out that we cannot find out any pattern and hence, if my x is not known to you, if I do not give you the value of x and just give you the value of y and it will be very difficult in general to compute the value of x, if my group is a sufficiently large group.

(Refer Slide Time: 17:24)



Main Application of Cryptography

- ❑Privacy
- ❑Authenticity
- ❑Integrity

So, now, let us see some applications of the discrete log problem in the context of cryptography. So, let me tell you something about cryptography. So, it is a mathematical science, and the main goal of the cryptography is to establish a secure communication channel between 2 entities say, Sita and Ram, who do not know anything about each other, they are meeting for the first time over the internet, and they want to talk over the internet by exchanging public messages.

And at the same time, they would like to ensure that, no one else should be able to find out what exactly they are communicating. So, main application of cryptography is that we would like to run some algorithm. And using those algorithms we would like Sita and Ram to exchange messages, so that, it should give them the effect of a secure channel, secure channel in the sense, it would look like as if Sita and Ram are doing conversation over very secure channel which provides 3 properties.

It should provide the privacy of the communication, namely it means that, even if a third party observes whatever communication is happening between Sita and Ram and even if the third party knows the protocol description, according to which Sita and Ram are doing the conversation, still that third party should not be able to figure out what exactly Sita and Ram actually are talking about that means, the actual contents of their messages.

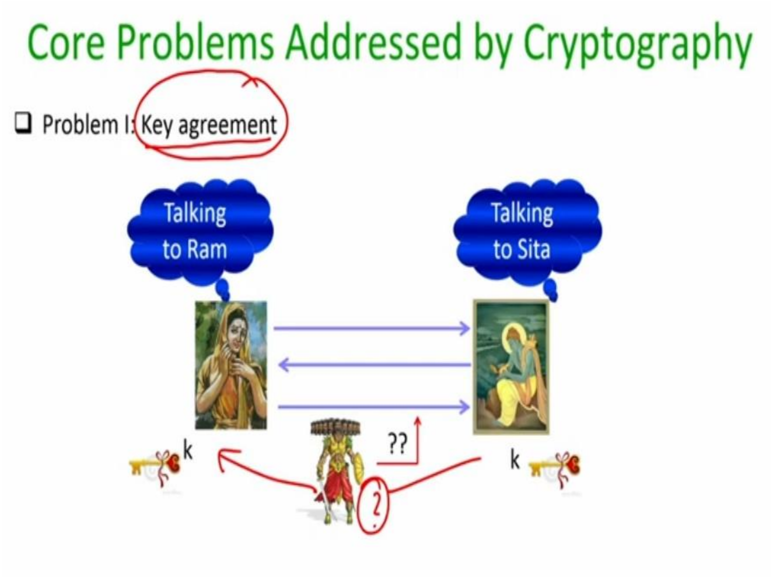
So that is a rough definition of privacy, we also need the authenticity property namely any message or any packet which is coming to Sita, it should have a proof that indeed it came from the person called Ram and in the same way any packet which goes to Ram, there should be a proof that or there should be a mechanism to verify that indeed that packet came from the person called Sita. So that is a rough definition of authenticity property.

And the third requirement of this secure channel is that of integrity. That means, if there is a third party which messes some of the bits or contents which are exchanged over the secure channel then it should be detected at the receiving end. So, through cryptography, we achieve all these 3 properties. So, basically cryptography gives you a set of algorithms, a set of protocols, according to which Sita can convert her messages in some format and communicate to Ram and vice versa.

So, there are lots of applications of cryptography. So, for example, if you are a user and if you are doing a net banking transaction then you are supposed to give your net banking password, at that time, you do not want your net banking password to be revealed to a third party, it should be securely communicated to the bank. So that is an application of cryptography.

In the same way, whenever you are buying something on the internet on Amazon, you are asked to enter your credit card details; again, you would like to exchange or send your credit card information in a secure way without any third party knowing about the exact details of your credit card information. So, again, cryptography is coming into picture. So, it turns out that cryptography is now used left and right in each and every application because slowly and slowly each and everything is becoming digital. So now, you might be wondering what sort of algorithms we use in cryptography?

(Refer Slide Time: 21:01)



Core Problems Addressed by Cryptography

- Problem I: Key agreement

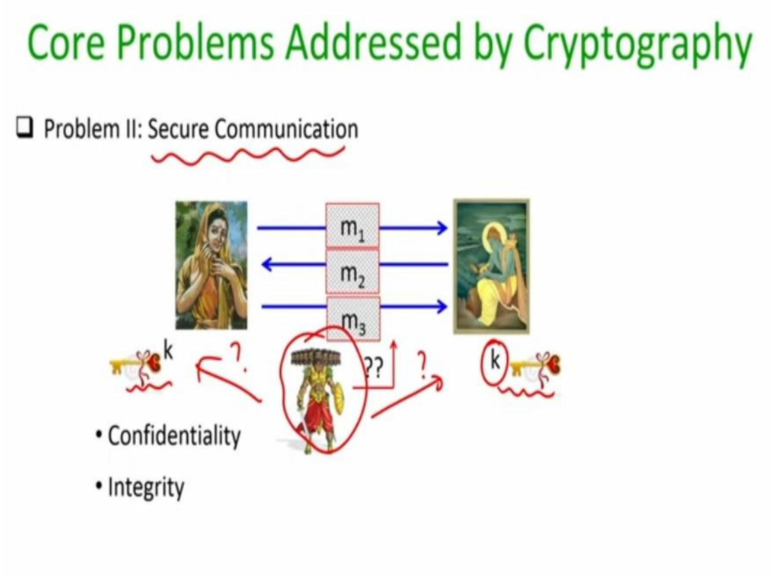
So, now I will give you very simple algorithms based on number theory and abstract group algebra, abstract algebra that we have seen till now. So, the 2 core problems that are addressed by cryptography are the following. The first problem is that of key agreement. So, what exactly is the requirement in the key agreement problem? So, the setting is the following.

We have 2 entities Sita and Ram, who do not have any pre shared information that means, no secret question, secret date of birth, nothing. They are meeting for the first time and they are going to talk publicly over the internet. So, we need a protocol here according to which Sita and Ram should talk to each other, and the protocol description also will be publicly on that is also important.

It is not the case that, process by which Sita is going to decide her message is known to Ram beforehand and vice versa. Because I am assuming that they do not know anything beforehand. So that means, if at all they are going to use a protocol that will be publicly available. So, we need a publicly available protocol according to which Sita and Ram should talk to each other.

And at the end of the protocol, magically, both Sita and Ram should arrive at a common key k which is a binary string of some length. And the interesting property, the security property that I need from this key agreement protocol is that, if there is any third party who has monitored the communication between Sita and Ram and who knows the protocol description should not be able to figure out what exactly is the key k which Sita and Ram has output. It might look like an impossible task, but we will see soon, how exactly key agreement can be achieved?

(Refer Slide Time: 22:49)



Core Problems Addressed by Cryptography

- Problem II: Secure Communication

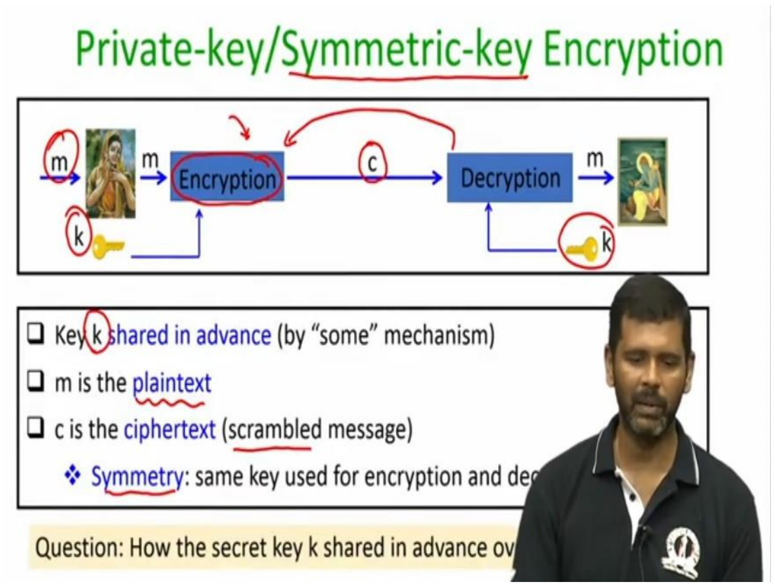
- Confidentiality
- Integrity

And assuming that the key agreement has been achieved, the second problem that is addressed by the cryptography, the second core problem, I should stress here, it is not the case that secure communication is the only problem, the second core problem addressed by cryptography startup secure communication. So, the setting here is the following, we will assume that Sita and Ram has already executed the key agreement protocol over the internet, and they have agreed upon a common key.

And now using this common key, we would require Sita and Ram to come, we would require some algorithms which are publicly known, according to which Sita can convert or encrypt her message into some garbled text into some garbage and communicate to Ram and Ram should be able to convert back those garbage or scrambled text back to the original contents using the same key, k which Sita has. So, namely we want to come up with algorithms which should help me to do secure communication.

And by secure communication here I mean that, if there is a third party or Ravana, who knows the public description of your algorithm but does not know the value of key then even after observing the communication happening between Sita and Ram and even after knowing the full protocol description according to which these messages have been computed, the Ravana should not be able to come up with the values of m_1, m_2, m_3 and so on. So that is the second problem addressed by cryptography.

(Refer Slide Time: 24:23)


Private-key/Symmetric-key Encryption

- Key k shared in advance (by "some" mechanism)
- m is the plaintext
- c is the ciphertext (scrambled message)
  - Symmetry: same key used for encryption and de

Question: How the secret key k shared in advance ov

So, it turns out that there are two kinds of, two classes of cryptographic algorithms which we use. The first category is that of private key or symmetric key encryption. In the symmetric key encryption, the setting is the following. It will be ensured that a common key is already shared between Sita and Ram by some mechanism, say, by running a key agreement protocol and no one else apart from Sita and Ram knows the value of that key.

Now, if that is the case assuming this setup has been done, the way symmetric encryption works is as follows. So, imagine Sita has some message, it could be an email, it could be just a hi message, it could be anything, it could be her net banking password. So, she has some message which is abstracted as a binary string, we call her message as plain text. We want to design an algorithm which we call as an encryption algorithm which takes a message m and the key k both of which are binary strings.

And it should produce another binary string which we call a ciphertext. And this ciphertext will be the scrambled message because it will have absolutely no meaning, in the lose sense and Sita will compute this ciphertext and communicate it over the internet and send it to Ram. Now, once Ram obtains this scrambled message, he will have a decryption algorithm, he will have in a sense, he will know that Sita has used an encryption algorithm whose details are publicly known and the corresponding matching decryption algorithm also will be publicly known.

So, Ram will use the corresponding decryption algorithm. And the inputs for the decryption algorithm will be the ciphertext that he has received and the same key which has been used by

Sita to produce the scrambled text. And this decryption algorithm will magically produce back the same message m which Sita has used or wanted to communicate.

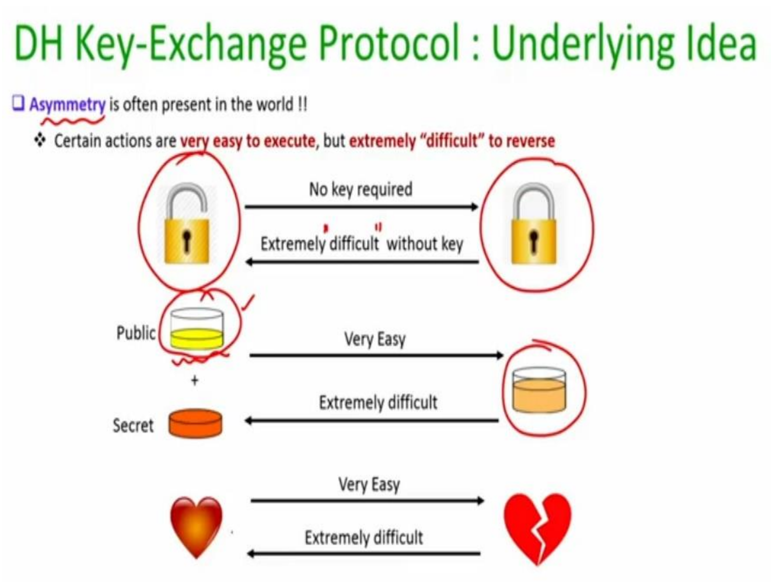
So, the reason it is called symmetric key encryption is because of the symmetry, namely, the same key is used both for encrypting the message as well as for decrypting the message. Now, the system might look very neat, very clean, just you encrypt your message and send a message, encrypted message, Ram receives encrypted message and decrypt and recover back the message.

So, the analogy could be that, assume Sita and Ram have already exchanged a key for a physical lock. If Sita has a message, what she can do is, she can take a box, keep her message written in a paper inside the box and close the box with a lock and using the key that she has. And now she can send this lock box by a courier or anything. So, if there is a third person who does not have the key for opening the lock of the box, he would not be able to do that.

Now, once the courier is delivered to Ram, since Ram also have the same key, he can use it, unlock the lock and see what exactly is the content kept inside that box. So, the same message which Sita wanted to send will be delivered to Ram. So that is the analogy. But the system will work if both Sita and Ram have already agreed upon this common key for the lock. How at the first place they can do that?

Because everything will be now happening over a public channel because it is not the case that Sita and Ram knew beforehand in advance. It is like saying the following, if I want to do a transaction over the internet; Amazon may not be knowing well in advance that a person called Ashish Chowdhury, would like to do a transaction with Amazon. So, I will be doing my transaction at a run time, how at the first place I establish a secure key with Amazon? And that too, by communicating over the internet, so that is a big question. How at the first-place key agreement has taken place?

(Refer Slide Time: 28:24)


DH Key-Exchange Protocol : Underlying Idea

- ❑ Asymmetry is often present in the world !!
  - ❖ Certain actions are very easy to execute, but extremely "difficult" to reverse

So, it was a folklore belief that it is not possible to agree upon a common key by interacting over a public channel. But the Turing Award winner, Diffie and Hellman, proved this belief to be incorrect, by coming up with their seminal key exchange protocol. So, I would not be going into the full details of security proof and other details of the key exchange protocol, I will just try to give you the underlying idea.

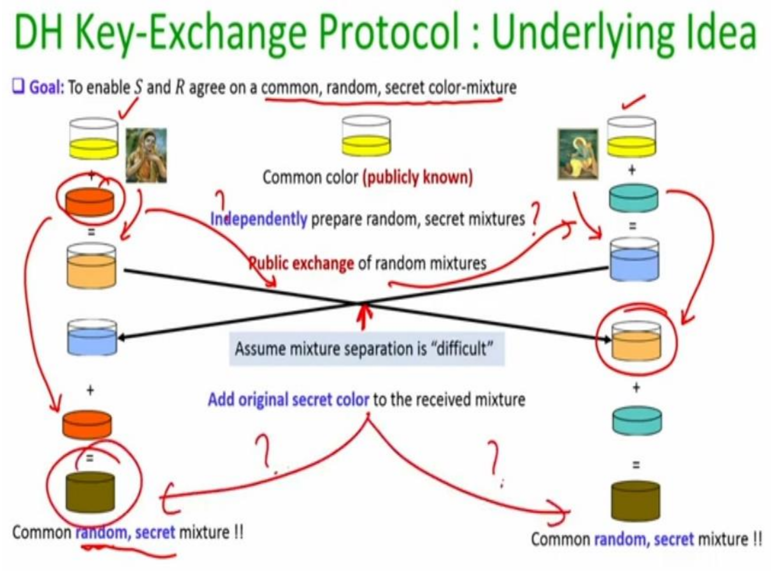
So, the main idea used in their key exchange protocol is the following. They observed that there are plenty of tasks in this universe which are asymmetric, they are asymmetric in the sense, they are very easy to compute in one direction. That means, it is very easy to go from one state to another state but extremely difficult to reverse back the effect of that action. So, for instance, if I take a padlock in an open state then it is very easy to lock the padlock, I just have to press it I do not need any key.

But now, once I go to closed state of this padlock and if I asked you that, can you open it, until and unless you do not have the key it will be extremely difficult for you. So, I am saying it is extremely difficult to open it without a key, there might be some other mechanisms to open it as well. You might have a Jugaad method but that will be extremely difficult, very time consuming. I am not saying it is impossible. So, there is a difference between extremely difficult and impossible.

In the same way, consider this task; I take a publicly known color. And then I prepare a secret mixture. In the sense I do the public color, I add a secret color. And then once the mixture is prepared, I give it to you. So, the mixture preparation is very easy. And now if I ask you that, okay, I give you this mixture, I also tell you the public color with which I started with, can you tell me what exactly was the secret color that I added?

Again, it is not an impossible task, you yourself can take a tumbler with a publicly known color and keep on adding various colors which you can add and see whether that gives you the same secret mixture that I have. But that might be a very time-consuming affair. And most importantly, it is very easy to break someone's heart by saying very bad words, but it is very extremely difficult to win the love and confidence of that person back.

(Refer Slide Time: 30:56)



DH Key-Exchange Protocol : Underlying Idea

❑ Goal: To enable S and R agree on a common, random, secret color-mixture

Common color (publicly known)
Independently prepare random, secret mixtures
Public exchange of random mixtures
Assume mixture separation is "difficult"
Add original secret color to the received mixture
Common random, secret mixture !!
Common random, secret mixture !!

So, based on this idea that asymmetry is there in lots of tasks. This is the underlying idea of Diffie Hellman key exchange protocol. So, I will be first explaining the protocol assuming that Sita and Ram want to agree upon a common secret mixture which should be random at least should be decided and no one else should be able to learn what exactly is the secret mixture. So, to begin with, both Sita and Ram will be starting with some common publicly known color.

And now, what they will be doing is the following. They will prepare independently some secret mixtures. So, Sita will prepare her secret mixture independently and Ram will be preparing his secret mixture independently, by adding a secret color, individually and then they will publicly exchange their mixtures. So, Sita will send her a copy of the mixture to Ram, Ram will send his copy of the mixture to Sita.

And here I am assuming that mixture separation is an extremely difficult task that means, if there is a third party who is observing the communication here, who knows the entire process according to which Sita and Ram are acting, so, he knows that both Sita and Ram started with a secret colour. He also knows that, Sita has added a secret component but the exact value of that secret component is not known to this third party.

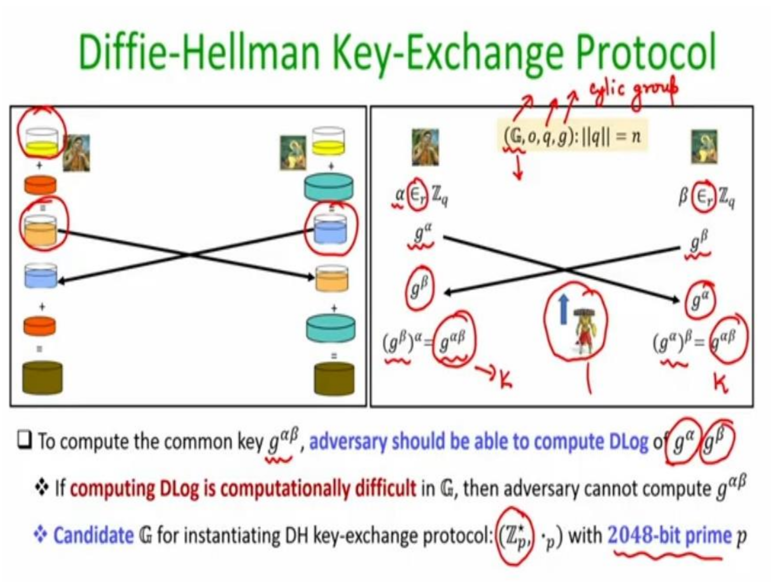
In the same way, he knows that Ram has added a secret component but he does not know what exactly is that secret component? And now of course, he is seeing the public mixtures being exchanged. Now, what is the goal? The goal for Sita and Ram is to come up or agree upon a common mixture which should be known only to them. So, what they can do is, they can individually add the secret component that they have added, to the copy of the mixture that they are receiving from the other party.

So, whatever Ram's mixture that Sita has received, she takes that and to that she adds whatever components she has added to prepare her secret mixture. And same task is done by Ram. He takes Sita's mixture and to that he had the secret color that he added to prepare his copy of the secret mixture. And what this will give? This will give both of them a common mixture because it does not matter in what order you add the 3 colors finally, it will give you the same mixture.

And it will be random, it will be random in the sense, next time Sita and Ram again runs the same protocol, they will start with the same copy of the public mixture. But now the secret components with Sita and Ram are going to add might be different because every time every execution of the protocol they will be preparing independent mixture. So that is why, the output at the end of each instance of this protocol will be a random mixture. And why it will be secret?

It will be secret because any third party who is monitoring the communication, he would not be able to separate out the secret contribution of Sita and Ram. So that is why he would not be knowing what exactly is the final mixture that Sita and Ram have obtained. Now, we have to convert this whole process, this whole color exchanging idea into a concrete algorithm, mathematical algorithm and protocol.

(Refer Slide Time: 34:13)


cylic group
$(\mathbb{G}, o, q, g): ||q|| = n$
$\alpha \in_r \mathbb{Z}_q$
$\beta \in_r \mathbb{Z}_q$
g^α
g^β
$(g^\beta)^\alpha = g^{\alpha\beta}$
$(g^\alpha)^\beta = g^{\alpha\beta}$
K
K

Diffie-Hellman Key-Exchange Protocol

- To compute the common key $g^{\alpha\beta}$, adversary should be able to compute DLog of g^α g^β
  - If computing DLog is computationally difficult in $\mathbb{G}$, then adversary cannot compute $g^{\alpha\beta}$
  - Candidate $\mathbb{G}$ for instantiating DH key-exchange protocol: $(\mathbb{Z}_p^*, \cdot_p)$ with 2048-bit prime p

So, on your left hand side, I have returned the blueprint of the color based key exchange protocol. And now, I will instantiate each and every step by concrete mathematical step. So, both Sita and Ram started with some public information. That public information is the description of a cyclic group, its order and the description of the generator. So that is a public information.

So, Sita and Ram both knows that, okay, they are going to use this cyclic group and every third party who wants to derive the key that Sita and Ram have agreed, are going to agree upon will also know the description of the cyclic group. So, it is like saying the following, if Sita is considered as a user and Ram is considered as a Amazon and if Sita wants to do a transaction with Ram and want to agree upon a common key then, a third party who wants to break the communication knows what exactly are the description of the cyclic group that Sita and Ram are going to use.

So now, the first step of the protocol was that, Sita and Ram prepare some random secret mixtures independently. The corresponding instantiation of that step is that Sita randomly picks a group element and Ram randomly picks a group element, how they can do that?

So, Sita picks a random $\alpha \in_r \mathbb{Z}_q$ in the range 0 to q - 1. So, this notation means that α is randomly chosen, this r denotes that it is randomly chosen. So, she picks the α randomly from the set 0 to q - 1 and computes g^α. And independently Ram picks a random β in the range 0 to q - 1 and computes g^β. So that is their independent secret mixtures which they now communicate.

And now, what was the final step? So, Sita upon receiving Ram's mixture, she adds her own secret component, her secret component was α. So, adding here will mean that, she will take Ram's mixture namely g^{β} and raise the whole thing to α which will give her $g^{\alpha \cdot \beta}$. And what Ram is going to do? He will take Sita's mixture, namely g^{α} and to that he will add his own contribution.

Again adding in this context means, raise it to the power β which will result in $g^{\alpha \cdot \beta}$. But what about a third person, an attacker, an eavesdropper, who has monitored the communication, will he be able to compute $g^{\alpha \cdot \beta}$ because that is a common key which Sita and Ram are going to agree upon. Well, for the adversary or for the third person to compute $g^{\alpha \cdot \beta}$, he should know α or β.

Because if any of these 2 values is learned by the attacker, he can easily compute $g^{\alpha \cdot \beta}$. But for learning α or β he has to basically solve, either this instance of discrete log or this instance of discrete log, namely, upon saying g^{α}, he should be able to compute α in the reasonable amount of time or given g^{β} he should be able to compute β in a reasonable amount of time.

That means, if I ensure that computing discrete log is extremely time consuming for this attacker and by time consuming means, at least it takes say 10 years or 15 years then, I can say that this protocol is safe because I do not care after 15 years if attacker comes to know, what exactly I communicated 15 years back. Because I will not be interested to keep the privacy of my communication for so long, so, as long I ensure that it is extremely difficult.

How extremely difficult it is? So, in loose tense, it is a order of several years, if it is extremely difficult for an attacker to come, solve an instance of discrete log, then this protocol gives me a mechanism according to by which Sita and Ram can agree upon a common key. So, now, you might be wondering that what should be the choice of the group, how big it should be and so on.

So, it turns out that, if we instantiate this protocol with my group being $\mathbb{Z}_{p^*}$ that means, if I ensure that my group G is the set $\mathbb{Z}_{p^*}$ where, p is some 2048 bit prime number then, using current best computing speed machines for solving a random instance of discrete log, it will take order of several years and hence, an adversary who tries to attack the scheme will fail to do that.

And that ensures that Sita and Ram now, can safely use the key $g^{\alpha \cdot \beta}$ as the common key and run an instance of symmetric key encryption scheme to do the secure communication of their messages. So, now you can see that how exactly the concept that we have seen in the context of cyclic groups are useful to come up with a very important practical solution for a practical problem namely that of key agreement.

So, with that, I conclude today’s lecture. Just to summarize, in this lecture, we introduced the problem of discrete log and we saw that in some groups, solving discrete log might be very easy, in some groups, it is conjecture that solving a random instance of discrete log is extremely difficult. And if we work with those groups then, we can design practical algorithms for real world problems like the key exchange problem. Thank you!