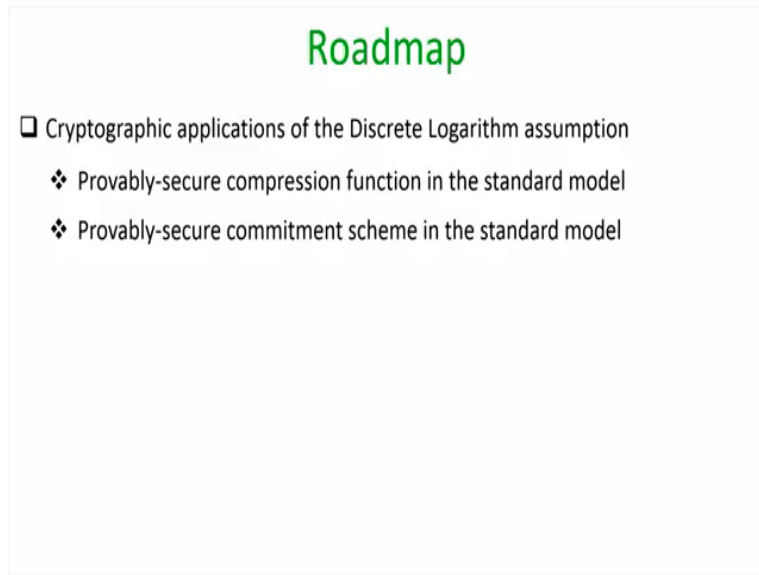


Foundations of Cryptography
Dr. Ashish Choudhury
Department of Computer Science
Indian Institute of Science – Bangalore

Lecture – 42
Cryptographic Applications of the Discrete Log Assumption

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:36)

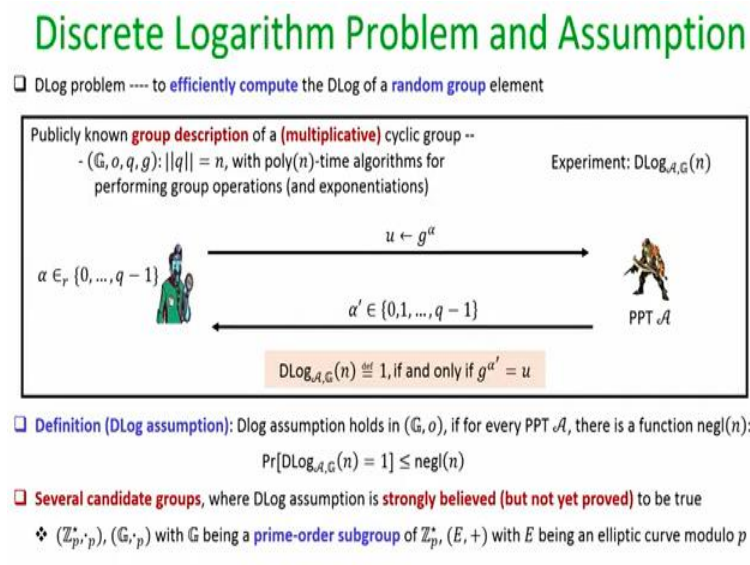


So just to recap we have seen till now, we have seen various cryptographic assumptions in the context of cyclic groups namely the DLog assumption, CDH assumption, DDH assumption and we have also seen candidate cyclic groups where we believe that those assumptions indeed hold namely those problems are indeed difficult to solve. In this lecture we will see some cryptographic applications of the discrete log assumption namely we will see provably-secure compression function in the standard model.

We will also see that how based on DLog assumption we can design provably-secure commitment scheme in the standard model. So, remember we had already seen instantiations of compression function and also instantiations of commitment scheme based on hash function. But their proof were not in the standard model, in the sense the proofs were given in the random oracle model which makes very strong assumption from the underlying hash function.

But in this lecture, we will see that how we can instantiate compression function, commitment schemes and give proofs without making any unconventional assumptions.

(Refer Slide Time: 01:35)

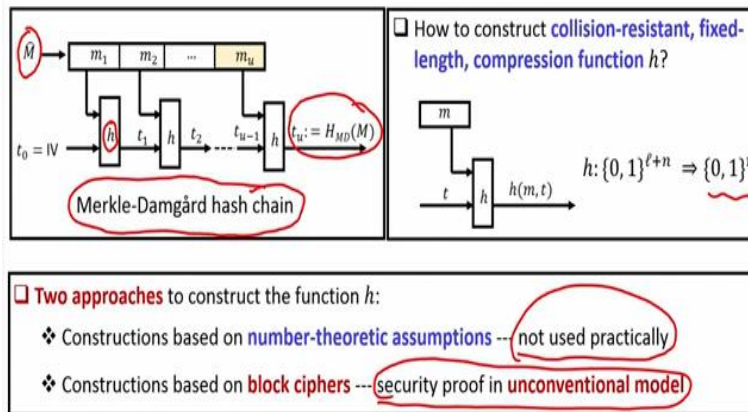


So just to recall the DLog problem and the DLog assumption. The DLog problem is you are given the description of a cyclic group, the group operation and the generator and you are given a random element from the group. Your goal is basically to come up with the discrete log of that random element in polynomial amount of time. The DLog assumption states that we say that the DLog assumption holds in that group if no poly time algorithm can solve the DLog challenge except with negligible probability.

And now we know at least 2 candidate groups where the DLog problem is believed to be hard and only we can use the groups \mathbb{Z}_p^* with underlying operation being multiplication modulo p or we can take a prime order subgroup of \mathbb{Z}_p^* or we can take the group to be the group based on the points on elliptic curves modulo p and underlying operation being the plus operation. So we will assume a multiplicative group but whatever we are going to discuss in this lecture, the steps can be simply modified or the steps of those cryptography primitives can be simply modified for a cyclic group where the underlying operation is addition.

(Refer Slide Time: 02:43)

Constructing Fixed-length Collision-Resistant Compression Functions



So let us see the first application namely constructing fixed-length collision-resistant compression functions. And for that let me recall the way we have constructed arbitrary length compression functions or hash functions using the Merkle-Damgård transformation. So, what the Merkle-Damgård transformation does for you is it takes an arbitrary input and gives you a fixed size hash for that.

For that we transform the input into a padded input to ensure that every block of the message is a multiple of n bits and then we do a chaining here. We compute a hash chain where in each iteration we take a current block of the message and the output of the previous hash or the output of the previous instance of the compression function. And then again apply an instance of the fixed size compression function and overall output of the hash function is taken as the output of the last instance of the compression function.

We have proved that if the underlying compression function h is collision resistant that means in poly time it is difficult to come up with a collision for the fixed size compression function, then the resultant hash function that we obtained by applying this Merkle-Damgård transformation is also collision resistant.

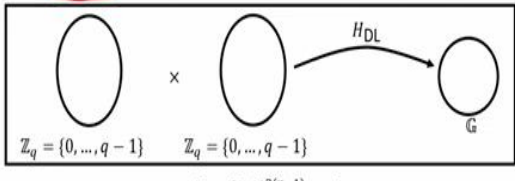
Now the question is how we construct this fixed-length collision resistant compression function which takes 2 inputs namely an input of size $l + n$ bits which can be parsed as 2 inputs one of size

ℓ bits and another of size n bits and gives you an output of n bits. We know 2 approaches for this: one approach is based on number-theoretic assumptions which we are going to see in this lecture. However, they are not used practically. The construction that we are aware of and which we have proved to be secure is based on block ciphers namely Davies-Meyer construction. However, the security proof of that construction is namely in the random oracle model.

(Refer Slide Time: 04:54)

Provably-secure Compression Function Based on the DLog-Assumption

- Given: a publicly known cyclic group (\mathbb{G}, o, q, g) , where $||q|| = n$ and a uniformly random element $h \in_r \mathbb{G}$
- Goal: to design a collision-resistant function $H_{DL}: \mathbb{Z}_q \times \mathbb{Z}_q \Rightarrow \mathbb{G}$, assuming DLog assumption holds for \mathbb{G}
- ❖ H_{DL} can be interpreted as $H_{DL}: \{0, 1\}^{n-1} \times \{0, 1\}^{n-1} \Rightarrow \mathbb{G}$ by encoding $(n-1)$ -length bit-strings as elements of \mathbb{Z}_q
- ❖ Let the elements of \mathbb{G} be encoded as ℓ -bit string
- If $2n - 2 > \ell$, then H_{DL} will be a collision-resistant, fixed-length compression function



Ex: \mathbb{G} being a prime-order subgroup of \mathbb{Z}_p^* consisting of quadratic residues

❖ $\ell = n + 1$

$H_{DL}: \{0, 1\}^{2(n-1)} \Rightarrow \mathbb{G}$

So, in this lecture we are going to see the construction based on the number theoretic assumptions whose security can be given just based on that discrete log assumption. What we are given here is we are given the description of a publicly known cyclic group where the group operation is a multiplicative operation. This is without loss of generality and we are given a publicly known generator. And as a setup we are also given a uniformly random element from the group whose discrete log is not known to anyone. So this is a one-time setup which we assume is done by a trusted entity, not by an adversary.

But once this setup is done, we can use this setup for polynomial amount of time or polynomial number of instances. Now using this setup, our goal is to design a collision resistant function which I call as HDL or H_{DL} based on the discrete log function. And it takes a pair of input where the first input is from the set \mathbb{Z}_q and another input is from the set \mathbb{Z}_q and output is going to be an element from the group.

And a security namely the collision resistance of the construction that we are going to design should be based on the DLog assumption. So, you can interpret function H_{DL} that we are trying to construct as a function which takes 2 inputs of size $n-1$ bits, by encoding $n-1$ length bit strings to elements of Z_q . This is because I am assuming here that the size of q is n namely the number of bits that I need to represent q is n .

So there exists certain groups where I do not need to do encoding, naturally every $n-1$ length bit string can be map to an element of Z_q . But if that is not the case I can do some kind of encoding and I can encode every $n-1$ length bit string as an element of Z_q . So we can imagine that this function H_{DL} which we are interested to compute takes an input of overall size of 2 times $n-1$ bits which can be passed as 2 chunks of $n-1$ bits each and again which are mapped to 2 respective elements of Z_q . Suppose the elements of the group are encoded as l -bit strings then it turns out that if $2n - 2 > l$ then we can view this function H_{DL} which we are going to construct as a collision resistant function and indeed there are several cases where indeed $2n - 2 > l$.

For example, if we take the group \mathbb{G} to be a prime order subgroup of Z_p^* where p is say $2q + 1$ namely it consists of all quadratic residues. Then your l is nothing but $n + 1$ because the size of p will be more than the size of q . And in that case clearly $2n - 2 > l$ and hence if instantiate H_{DL} on this particular group of prime order subgroups of Z_p^* based on quadratic residues, then the resultant H_{DL} function can be viewed as a compression function.

(Refer Slide Time: 07:52)

Provably-secure Compression Function Based on the DLog-Assumption

Given: a **prime-order** cyclic group (\mathbb{G}, o, q, g) , where $||q|| = n$ and a **uniformly random element** $h \in_r \mathbb{G}$

Definition (representation of a group element relative to g and h): For any $u \in \mathbb{G}$, a pair $(\alpha, \beta) \in \mathbb{Z}_q^2$ is called a representation of u , relative to g and h , if the following holds

$u = g^\alpha h^\beta$

Fact I: For any $u \in \mathbb{G}$, there exist q distinct representations of u , relative to g and h

For every $\beta \in \mathbb{Z}_q$, there exists a **unique** $\alpha \in \mathbb{Z}_q$, such that $g^\alpha = u(h^\beta)^{-1}$ holds

Fact II: Given **two distinct representations** $(\alpha, \beta) \neq (\alpha', \beta')$ of any $u \in \mathbb{G}$ relative to g and h , one can **efficiently compute** $\text{DLog}_g h$

$$\left. \begin{array}{l} u = g^\alpha h^\beta \\ u = g^{\alpha'} h^{\beta'} \end{array} \right\} \begin{array}{l} \diamond g^{[\alpha - \alpha']} = h^{[\beta' - \beta]} \\ \diamond [\beta' - \beta] \neq 0 \\ \triangleright \text{Else, } [\alpha - \alpha'] = 0, \text{ as } g \text{ is a generator of } \mathbb{G} \end{array}$$

$$\diamond \Delta \equiv [\beta' - \beta]^{-1} \bmod q \text{ exists, which can be computed in } \text{poly}(n) \text{ time}$$

$$\diamond (g^{[\alpha - \alpha']})^\Delta = h \Rightarrow \text{DLog}_g h = ([\alpha - \alpha'] \Delta) \bmod q$$

So before going into the construction of the compression function, let us see a definition here. Remember as part of setup you are given a uniformly random element h from the group, apart from the generator. Now we define what we call as the representation of a group element relative to the base g and h . So for any group element u belonging to the group \mathbb{G} , we say any pair (α, β) from the set \mathbb{Z}_q is called as a representation of the element u relative to the base g and h if the relationship $u = g^\alpha h^\beta$ holds. Again, I stress this whole discussion is with respect to the assumption that my underlying operation is a multiplicative operation. But again, all these definitions and discussion can be carried over for the cyclic group where the underlying operation is addition. So, we have the definition of a representation of an element relative to g and h .

Now with respect to these definitions we have certain facts here. The first fact is that if you take any element u from the group \mathbb{G} then there exist exactly q number of distinct representations of the same element u relative to g and h . What that means is that, if you fix any arbitrary β from the set \mathbb{Z}_q then corresponding to that fixed β there exist a unique α from the set \mathbb{Z}_q such that the relationship $g^\alpha = u * (h^\beta)^{-1}$ holds.

And that is why for β_1 , you will obtain a corresponding α_1 which will be a representation such that (α_1, β_1) will be a representation of u .

In the same way if you fix another β say β_2 that will imply another α_2 such that (α_2, β_2) is the representation of same element u and so on. So, like that how many β s you can fix? You can have q number of β s because the range of β that is the set Z_q . And that is why the range of corresponding α s is also Z_q and that is why you have exactly q number of distinct representations of any element u . So that is the first fact.

The second fact is that if you are given 2 distinct representations for the same element u , then you can extract out the discrete log of the random element h to the base g . Why that? So, imagine you are given 2 distinct representations of the same element u say (α, β) is one of the representations and (α', β') is another representation both representing the same element u relative to g and h . That means these 2 equations hold and says (α, β) and (α', β') are distinct. That means pair vice (α, β) is different from (α', β') .

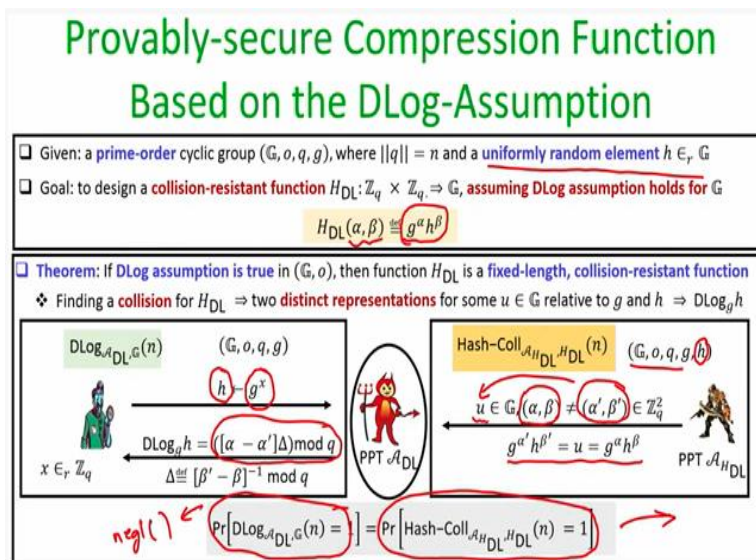
Now if both this relations hold, then I can do the substitution and obtain that $g^{\alpha - \alpha'}$ is same as $h^{\beta - \beta'}$. And now I claimed that $(\beta' - \beta)$ is non 0 because if $(\beta' - \beta)$ is 0, then h^0 is 1. And h^0 is 1 that means the identity element which is possible only if $(\alpha - \alpha')$ is also 0 which automatically implies that $\alpha = \alpha'$ and $\beta = \beta'$ which is contradiction to the assumption that (α, β) is different from (α', β') . That means $(\beta' - \beta)$ is non-zero. Since it is non-zero, then I can always compute the multiplicative inverse of $(\beta' - \beta)$ which I denote by Δ modulo q . Given β' and given β , the inverse namely the value Δ modulo q can be computed in $\text{poly}(n)$ time. There exist well known algorithms for computing this value Δ which I am not discussing here.

But you have to believe me that if $(\beta' - \beta)$ is non-zero, then the multiplicative inverse of $(\beta' - \beta)$ namely Δ modulo q exist which can be computed in poly time. So if we can compute Δ in poly time and if we are given g and if we are given α and α' , then we can clearly see that $g^{(\alpha - \alpha')}$ and in the exponent multiplied by Δ will give you the value h . Which means that the Dlog_g^h is nothing but $(\alpha - \alpha')$ multiplied by Δ modulo q .

So that means if someone can give you 2 distinct representations for any element u from the group \mathbb{G} , then using that element algorithm or using the pair (α, β) and (α', β') we can compute the Dlog_g^h

as well. And that forms the basis of computing our compression function or constructing our compression function that we are interested in.

(Refer Slide Time: 13:02)



The construction of the collision resistant compression function is as follows: as a setup we are given a uniformly random element and to compress the input pair (α, β) as per this function H_{DL} what we have to do? Basically, we have to compute the value (g^{α}, h^{β}) that is the overall output. And I claim here that if the DLog assumption is true in the underlying group \mathbb{G} that means that the DLog problem is indeed difficult to solve in poly time in the underlying group \mathbb{G} , then indeed this function H_{DL} is a fixed-length collision resistant function Why? So this is because finding a collision for this function H_{DL} means coming up with 2 distinct representations for some element u from the group related to g and h . As we have seen in the last slide if we have 2 distinct representations for some group element then it automatically means we know how to compute the Dlog_g^h which goes against the assumption that the discrete log problem is difficult to solve in the group.

So let us formally establish this whole implication by a reduction here. So imagine we have a polytime algorithm who knows how to find collision in the function H_{DL} . Using that algorithm, we construct another algorithm which can solve an instance of discrete log problem in the group \mathbb{G} . So this adversary A_{DL} , it participates in an invocation of the discrete log experiment with respect to the group \mathbb{G} .

Then as part of the challenge for that experiment the challenger of the discrete log experiment picks a random x and gives the challenge h which is g^x and the goal of this adversary A_{DL} is to extract out this x . What it does is it invokes the adversary $A_{H_{DL}}$ and participates in an instance of the collision resistance experiment and as part of that experiment what it does is it creates a setup where the group description is the same group \mathbb{G} where the adversary is A_{DL} wants to solve the discrete log problem and as a part of setup h is given as the public setup where h is uniformly random.

So, you see that h that is thrown as a challenge to the discrete log solver is used now as a setup for this H_{DL} sub function. Now as per the property of this collision finder algorithm it can come up with a collision namely it comes up with a hash value u which could be the hash value of (α, β) as well as the hash value of (α', β') such that (α, β) and (α', β') are different but their hash values are u .

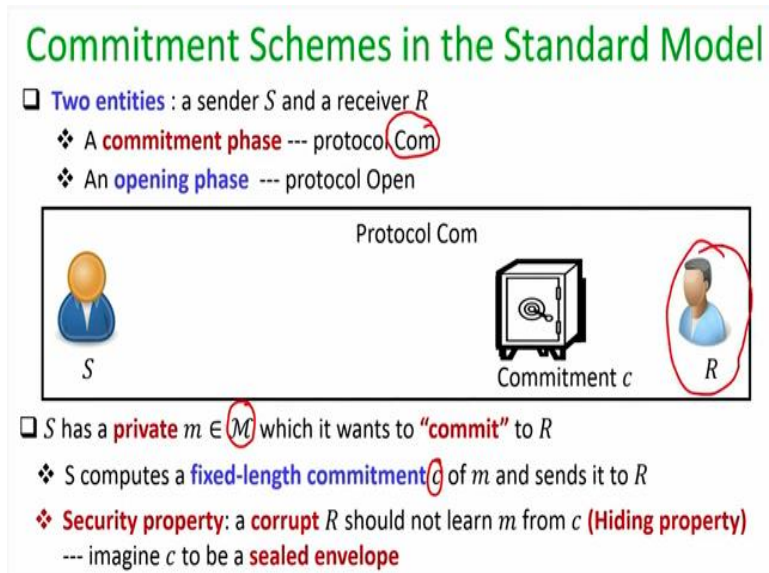
And as per the description of H_{DL} function since the hash of (α, β) and a hash of (α', β') is u that means this relationship holds. So now what this adversary A_{DL} does is as soon as it sees that adversary $A_{H_{DL}}$ is giving a collision by using the mathematics that we had seen in the last slide, it ends up computing $Dlog_g^h$ namely it computes $(\alpha - \alpha')$ multiplied by the multiplicative inverse of $(\beta' - \beta)$ modulo q . And you can see that the running time of the adversary A_{DL} is exactly the same as the running time of adversary $A_{H_{DL}}$.

And relationship wise, we can say that the probability that the D log solver wins its experiment namely it wins the instance of the discrete log experiment is exactly the same as the probability with which the collision finder algorithm wins the collision finding algorithm against the HDL function.

But since we are assuming that the DLog assumption holds in our group \mathbb{G} , then we know that for any poly time algorithm, the probability that it can win an instance of discrete log in the group \mathbb{G} is negligible. That means the left hand side probability is always upper bounded by a negligible function that automatically implies that probability in the right hand side of the expression is also

bounded by a negligible probability. So that establishes the fact that the function H_{DL} that we have constructed indeed constitutes a collision resistant compression function.

(Refer Slide Time: 17:33)



Now let us see the second application of applying the discrete log assumption. So here we are now going to instantiate a commitment scheme where the proof will be in the standard model. Just to recall a commitment scheme is a scheme or a cryptographic primitive involving two entities are namely a sender and a receiver and it is 2-phase protocol. In the commit phase sender invokes a protocol commit or com and the opening phase an open protocol is executed.

So in the commit phase the sender has a message m from some message space which it wants to commit to the receiver. To do that sender basically computes the fixed length commitment of the message which I denote by c and it sends the commitment c to the receiver R . And the security property that we require from this com protocol is that if the receiver is a bad guy and computationally bounded then by seeing the commitment it should not learn what exactly is committed inside c . So from its viewpoint the c should view like a sealed envelope. It cannot see what exactly is the message which has been kept inside the commitment.

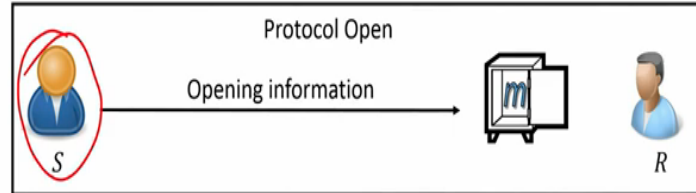
(Refer Slide Time: 18:41)

Commitment Schemes in the Standard Model

❑ **Two entities** : a sender S and a receiver R

❖ A **commitment phase** --- protocol Com

❖ An **opening phase** --- protocol Open



❑ S **reveals** m by opening the commitment c by providing an **opening information**

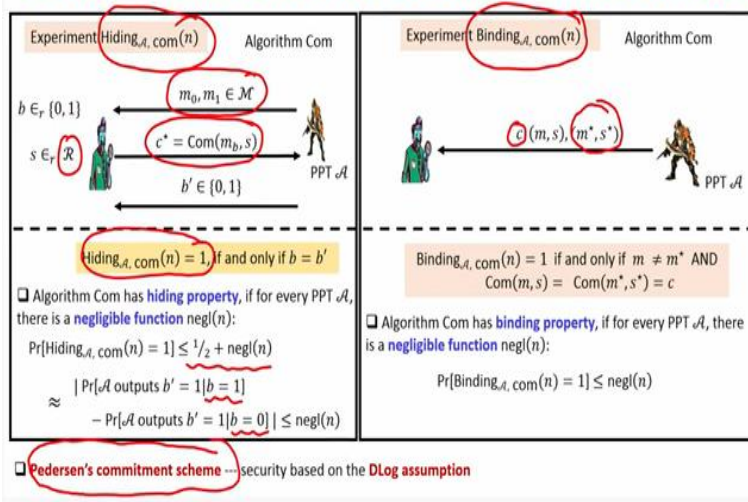
❑ R **verifies the opening information** and either **accepts or rejects** the revealed m

❖ **Security property**: A **corrupt** S should not be able to open a commitment c in **two different ways** --- **binding**

The second protocol is the opening protocol which implements the opening phase where sender released a message by providing some kind of opening information. Once the opening information is provided or verifies opening information and opens the sealed envelope, based on certain criteria it either accepts or rejects the value which is reveal now in the opening phase. The security property that we require here is that if the sender is bad and the receiver is honest then it should not be possible for a corrupt sender to open a commitment c in two different ways namely it should be binded to whatever it has committed in the commit phase.

(Refer Slide Time: 19:22)

Commitment Schemes in the Standard Model



So before going into the construction of the commitment scheme in the standard model let me also recall how we model the hiding property and the binding property. So the hiding property is

module by the hiding experiment where the adversary submits a pair of message from the message space and one of those messages is committed randomly by choosing some uniform randomness from the randomness phase by the challenger.

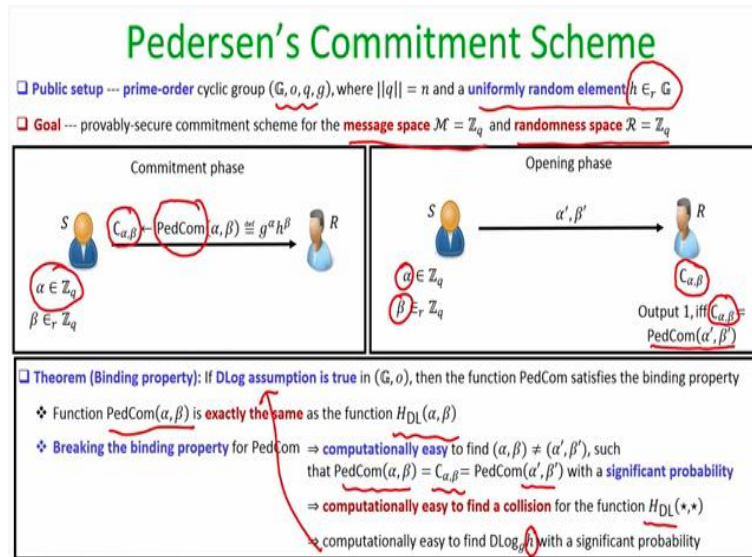
Challenge commitment is c^* which could be either a commitment of m_0 or a commitment of m_1 with equal probability and the challenge for the adversary is to identify whether it is seeing a commitment of m_0 or whether it is seeing a commitment of m_1 . And we say that output of the experiment is 1 or adversary wins to experiment if it can correctly identify whether it is seeing a commitment of m_0 or m_1 .

And a security definition is we say that a commitment scheme or commitment protocol com satisfies the hiding property, they have the probability of any polytime adversary within this experiment is upper bounded by $1/2 + \text{some negligible probability}$. Or stated otherwise, it does not matter whether the c^* is a commitment of m_0 or whether c^* is a commitment of m_1 . The output of this adversary should almost be the same say $b' = 1$ except with negligible probability.

And a way we have formalized the binding experiment: binding requirement is by the binding experiment where adversary simply submits a commitment c and a pair of message, randomness (m, s) and another pair of message, randomness the (m^*, s^*) such that m and m^* are different but still the commitment of m with respect to the randomness s and a commitment of m^* with respect to the randomness s^* turns out to be the same value c and we say that the algorithm com has binding property.

If the probability that any polytime adversary could come up with a pair of different messages committing to the same commitment is upper bounded by a negligible probability. So now what we are going to do is we are going to see a very nice commitment scheme which is called a Pedersen's commitment scheme which provides you the hiding property, binding property and its security is just based on the DLog assumption.

(Refer Slide Time: 21:40)



So the public setup that is given as part of the commitment scheme is the description of the cyclic group, the group order and uniformly random group element from the group whose discrete log is not known to anyone. This is like a trusted setup done by a trusted third party and it is done once for all. Once it is done, we can use it to instantiate or invoke polynomial number of invocations of the commitment scheme.

In the Pedersen's commitment scheme, the message space is basically \mathbb{Z}_q . That means the sender would like to commit any value in the range 0 to $q - 1$ and the randomness space is also going to be the set \mathbb{Z}_q . So the commitment phase or the com protocol is as follows: suppose sender wants to commit a value α which could be any value in the range 0 to $q - 1$. To commit that sender picks a uniformly random β from the randomness space namely \mathbb{Z}_q and it computes the commitment function which I denote as PedCom(α, β) which is nothing but $g^\alpha h^\beta$.

I denote that commitment by this notation. So C denotes the commitment and its subscripts α and β denotes that this is a string or a group element which is a commitment of some unknown value α with respect to the randomness β . So in the subscript the first component denotes a value which is committed and a second component denotes that randomness which is used to compute that commitment.

The opening phase is straight forward. So imagine receiver has an existing commitment available with it which it has obtained in a commitment phase and say sender has committed α with respect to the randomness β . Now it provides the opening information which I denote by (α', β') . If sender is honest and α' will be same as α and β' will be same as β . Whereas if the sender is corrupt and if it wants to reveal a different message in the existing commitment then it might submit (α', β') different from (α, β) .

To verify whether the revealing information or the opening information is correct or not. Receiver does the following: it recomputes the commitment with respect to the provided (α', β') namely it computes $g^{\alpha'} h^{\beta'}$ and compare it with the existing commitment that it has. If it matches, then it outputs α or outputs 1. Basically, it accepts α' otherwise it rejects α' .

So now what we are going to prove here is that if the DLog assumption is true in the underlying cyclic group, then the PedCom function that we have defined here satisfies the binding property. And this simply follows from the fact that if you see closely that the PedCom function that we have defined here is exactly the same as the compression function H_{DL} that we have defined just previously. That means breaking the binding property of this Pederson commitment function is equivalent to coming up with a pair of values (α, β) and (α', β') such that commitment function PedCom when it computed on (α, β) and when evaluated on computed on (α', β') gives you the same value as $C_{\alpha, \beta}$.

That means basically one knows how to find out a collision for the function H_{DL} and if one knows how to compute or find collisions in the function H_{DL} with significant probability in polynomial amount of time, then we know how to compute the $Dlog_g^h$ in polynomial amount of time with significant probability. But that goes against the assumption that discrete log assumption is true in the underlying group.

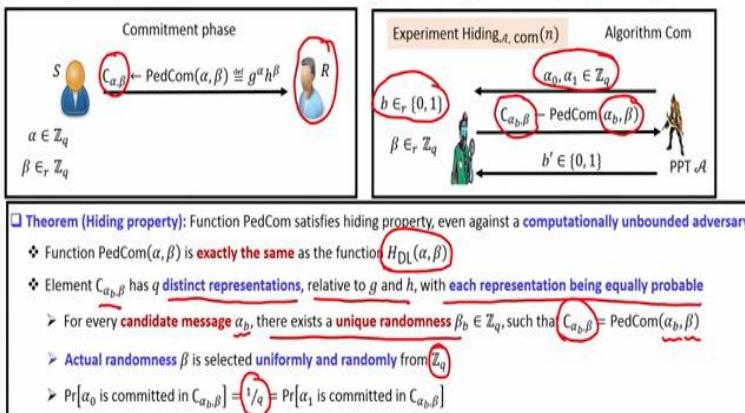
So that means if at all the binding property is broken then we know the discrete log problem is also easily solvable in the group which is against the assumption that the discrete log problem is difficult in my group. So that proves you binding property.

(Refer Slide Time: 25:51)

Pedersen's Commitment Scheme

Public setup --- prime-order cyclic group (G, o, q, g) , where $|q| = n$ and a uniformly random element $h \in_r G$

Goal --- provably-secure commitment scheme for the message space $\mathcal{M} = \mathbb{Z}_q$ and randomness space $\mathcal{R} = \mathbb{Z}_q$



So now let us try to prove the hiding property. So remember the goal of the hiding property is that if the sender is honest and if the receiver is malicious then by just looking into the commitment $C_{\alpha, \beta}$ it cannot find out whether it is seeing a commitment of m_0 or whether it is seeing a commitment of m_1 .

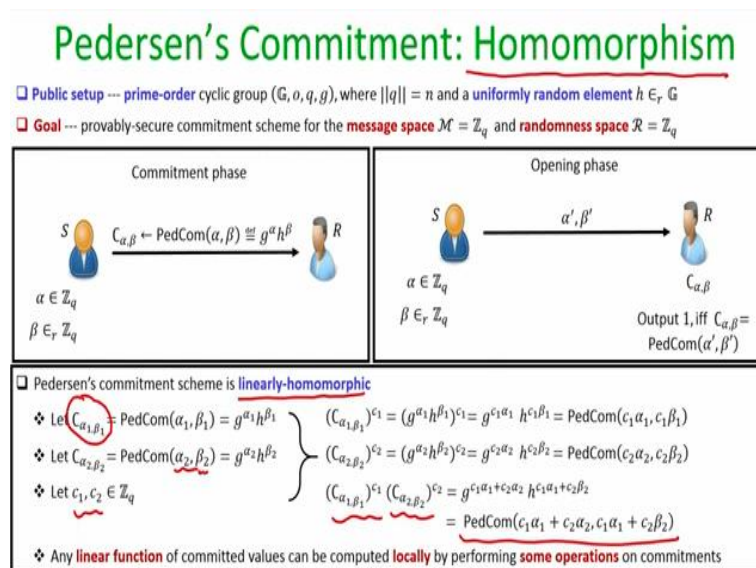
Namely, if we see the hiding experiment played with respect to the Pedersen's commitment scheme then here is how the hiding experiment would look: the adversary might submit a pair of messages say (α_0, α_1) . The challenger would have randomly picked any of those two messages for committing and to commit it picks a uniform randomness β and compute Pedersen's and commitment of the message α_β, α_b . And the goal of adversary is to find out whether it is seeing a commitment of α_0 or whether it is seeing a commitment of α_1 .

So what we can prove now is that this Pedersen commitment scheme satisfies the hiding property even against a computationally unbounded adversary. And this comes from the fact that this Pedersen commitment function is exactly the same as the compression function H_{DL} and what it means is that the commitment that the adversary is seeing it has exactly q number of distinct representations relative to g and h and each of these representations are equally probable.

That means for every candidate α_b whether it would be $\alpha_0, \alpha_1, \alpha_2$ or any α from the message space, there exist a unique randomness say β_b such that the commitments that adversary is seeing could

be the commitment of the message α_b under that randomness β . But actual randomness which is used by the sender or which is actually picked by the challenger in the hiding experiment is randomly chosen from the set \mathbb{Z}_q which means that the probability that α_0 is committed in this commitment C sub α_b, β and the probability that the commitment that adversary is seeing could be the commitment of message α_1 is exactly the same with probability $1/q$. The commitment that adversary is seeing could be the commitment of α_0 and with the same probability it could be the commitment of α_1 . And this holds even if my adversary is computationally unbounded that proves the hiding property.

(Refer Slide Time: 28:23)



So I would like to end up this lecture with another interesting feature of this Pedersen's commitment scheme which we call as the homomorphism property. So I am retaining the commit phase and opening phase of the commitment Pedersen's commitment scheme. And this Pedersen's commitment scheme is linearly homomorphic. What exactly I mean by that is, imagine you are given a commitment of some unknown α with respect to an unknown randomness β_1 .

So you just know the commitment but you do not know what exactly is the value which is committed and what is the corresponding randomness? And in the same way, imagine you are given a commitment of another unknown α_2 with respect to an unknown randomness β_2 and say you know public constant c_1 and c_2 belonging to the set \mathbb{Z}_q .

Now the way Pedersen commitment function is defined, if we take the commitment of α_1 and raise it to c_1 then it turns out that we basically end up getting a value which can be treated as the commitment of the value c_1 times α_1 under the randomness c_1 times β_1 . In the same way if you take the second commitment namely the commitment of the unknown α_2 and raise it to the known c_2 , basically we end up getting Pedersen commitment value which can be treated as a Pederson commitment of the value c_2 times α_2 with respect to the randomness c_2 times β_2 .

I stress here that you are just basically performing operations on the commitment. By performing those operations on the commitment, you are getting something which can be treated as commitment of some another value. And now if I take these two computed commitments basically, I end up getting a value which can be treated as a Pedersen commitment of the value c_1 times $\alpha_1 + c_2$ times α_2 with respect to the randomness c_1 times $\alpha_1 + c_2$ times β_2 . What it means that any linear function of the committed values can be computed locally by just performing some operations on the commitments.

That means even though you do not know what exactly is α_1 , what exactly is α_2 . If someone has committed α_1 and given you the sealed α_1 and a sealed α_2 in the form of commitment, and if you want to perform a sealed commitment of some linear function of α_1 and α_2 namely a function of the form c_1 times $\alpha_1 + c_2$ times α_2 where the linear combiners of the linear function namely c_1 and c_2 are known to you.

Then even without knowing α_1 and α_2 and a randomness β_1, β_2 you could perform operations on the commitment itself which will end up giving you the Pedersen commitment of some linear function of the unknown underlying values and that is what we mean by the linearly homomorphic property. So, it turns out that this linearly homophobic property of the Pedersen commitment scheme can be exploited in advanced cryptographic primitives like secure multi-party computation.

So that brings me to the end of this lecture. Just to summarize, in this lecture we have seen the applications of discrete log assumption namely we have seen an instantiation of fixed length compression function and we have seen an instantiation of commitment scheme namely Pedersen's

commitment scheme and a security of both these primitives are in the standard model and their security can be proved based on the discrete log hardness assumption. Thank you!