Foundations of Cryptography
Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
Indian Institute of Technology – Bangalore

Lecture –32
Random-Oracle Model - Part II

(Refer Slide Time: 00:34)

Hello everyone, welcome to this lecture. So just to recall in the last lecture, we had introduced the Random-Oracle model. In this lecture we will continue our discussion on Random-Oracle model. Specifically, we will see constructions of several cryptographic primitives based on hash functions, whose security we cannot prove just based on the standard properties of the hash functions and by standard properties I mean the collision-resistance property and it turns out that the proof for those cryptographic constructions, which we are going to see in this lecture can be completed only in the Random-Oracle model.

(Refer Slide Time: 01:08)

Is the Random-Oracle Methodology Sound?

- ❑ **Objections to the ROM**
  - ❖ Security of any ROM-based cryptographic primitive $\prod$ depends upon the random choices of the parties and the random choice of the oracle H
  - ❖ $\prod$ need not be secure, when H is instantiated by a **specific real-world hash function**
    - ➢ No concrete hash function can act as a random oracle

- ❑ **Support for ROM**
  - ❖ Several real-world ROM based crypto primitives which are highly efficient
  - ❖ A security proof in ROM implies that the **design principle of the primitive is sound**
    - ➢ Only weakness in the primitive could be due to the underlying instantiation of H
  - ❖ **No successful real-world attacks reported** for any ROM-based cryptographic primitive
    - A proof of security in ROM is always better than no proof at all

So, before we proceed, let us first argue whether the Random-Oracle methodology is sound or not? Because as you remember in the Random-Oracle model, we are making very strong assumptions about the underlying hash functions. Namely, we assumed that the underlying hash function behaves like a truly random function, with no entity in the system, which uses that cryptographic primitive, have access to that oracle. If someone wants to evaluate that underlying oracle on some input of its choice, then it has to make oracle call.

It is not allowed to see the internal details of the underlying oracle, which is far away from the reality because in reality when we instantiate a cryptographic construction based on hash function, then we have to select some candidate hash function and everyone in the system who is using that primitive will have access to the code of that hash function. So, that brings us to the debate whether the Random-Oracle methodology is sound or not? Now, it turns out that in cryptography we have 2 school of thoughts here.

We have one group which says that the Random-Oracle methodology is indeed useful, whereas the another group is against it. So let us see first objections to the Random-Oracle model. So, as I said earlier that the security of any cryptographic primitive Π which is based on Random-Oracle model, not only depends upon the random choice of the parties who are using that primitive, but also on the random choice of the underlying oracle H.

That means, that the primitive Π may not be secure when actually we go for the deployment of the primitive Π by instantiating that oracle H by some specific real-world hash function because it so turns out that no concrete hash function, say the SHA functions and MD5 functions and

so on can behave like a Random-Oracle. First of all, none of these hash functions behave like a truly random function and even if we so assume that they behave like a truly random function, we cannot prevent the adversary to see into the code of the underlying hash function. It will know the full details of the underlying hash function. Whereas the security proof that we gave in the Random-Oracle model, it depends upon what specific underlying Random-Oracle we are using during the instantiation of the primitive Π.

Whereas the other school of thought which supports Random-Oracle model is as follows. It turns out that there are several cryptographic primitives which are highly efficient and we will see some of the examples of such primitives in this lecture. Later on, we will see that most of the practical public-key cryptographic primitives or public key cryptosystems, they use hash functions and their security can be proved only if we go in the Random-Oracle model. We cannot prove the security of those public key cryptographic primitives based on hash function just using the collision-resistance or the standard properties of the hash functions. So, the support for the Random-Oracle model based proof is that a security proof that we give in the Random-Oracle model, it gives you the confidence that as far as the design principle of the cryptographic primitive is concerned, it sound. The only weakness in the primitive when we actually deploy that primitive could be due to the underlying instantiation of the hash function. As a design principle of that primitive Π, which uses the hash function as a random article, there is absolutely no flaw. So that is the support that we could come up for a proof that is there in the Random-Oracle model. Also, it turns out that for almost all the cryptographic primitives whose proof are there in the Random-Oracle model, we have obtained no successful real-world attacks when we actually instantiate hash functions by some real-world hash functions.

So that is another argument in the support of the Random-Oracle model based proof. The final argument which is there in the support of the Random-Oracle model based proof is that a proof of security of some cryptographic primitive in the Random-Oracle model is far better than no proof at all. That means, imagine you design a primitive Π based on some hash function and for that you have absolutely no proof at all, whereas you have the proof for the same primitive but in a very idealized setting namely say Random-Oracle model. Then you will be feeling confidence that indeed if I instantiate this primitive Π by instantiating the underlying hash function, then the only flaw that could be there might be due to the weaknesses in the

underlying instantiation of the hash function and then what you can think of, you can think of coming up with a better replacement or a better instantiation of that hash function instead of changing the design of the whole primitive Π. So that is a very strong argument in the support for a proof that is given in the Random-Oracle model.

As I said, you have 2 types of groups here, one group hates proofs given in the Random-Oracle model whereas other group supports proofs given in the Random-Oracle model. So, as a design principle, we should always try to design schemes whose proofs are not in the Random-Oracle model. Namely, we should try to design schemes which uses hash function, whose security can be proved just on the standard properties of the hash function, namely the collision-resistance property. But if at all, it turns out that we cannot give a proof just based on the standard properties, but we can give proofs in the Random-Oracle model, then we should go for proofs in the Random-Oracle model provided that the construction that we are giving is highly efficient. So basically you have tradeoff. You have a tradeoff in the sense that if you have a construction based on hash function and they are highly efficient, but with no proof. Whereas you have another construction for the same primitive with a proof in the standard model but highly inefficient, then for the sake of efficiency, we can prefer to go for the scheme based on the hash function whose proof is available in the Random-Oracle model.
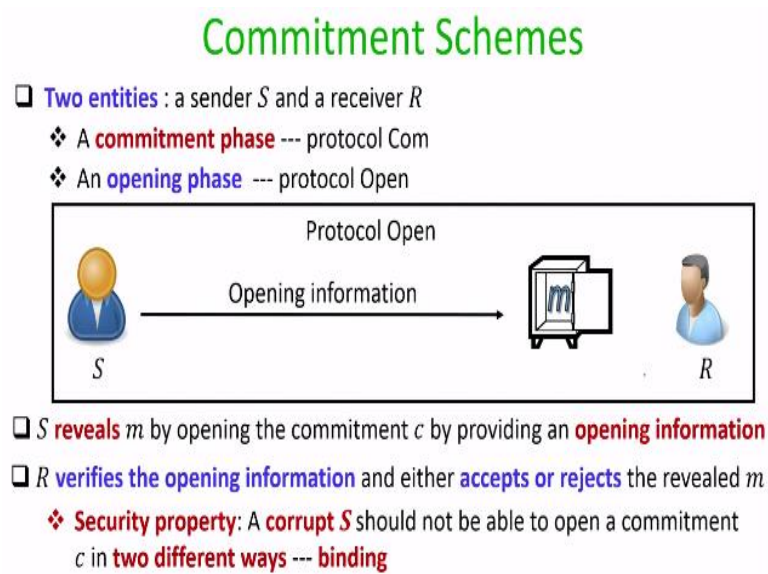
(Refer Slide Time: 07:21)
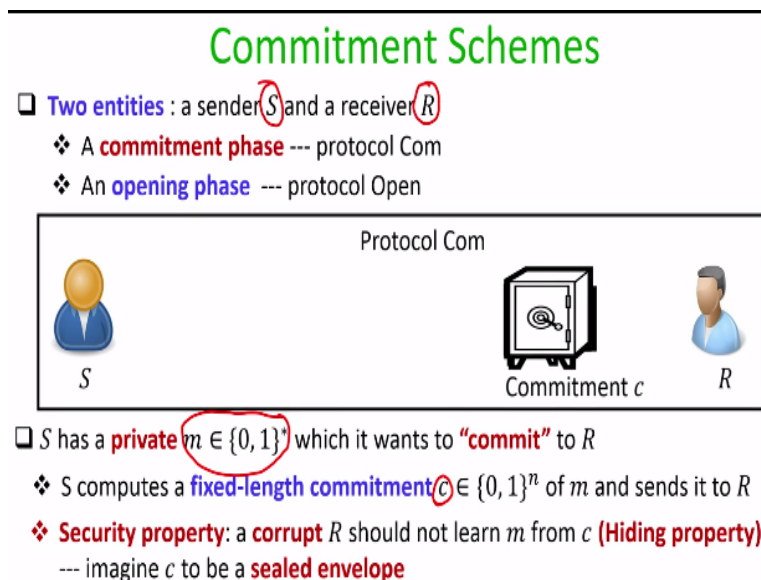
So, let us see some cryptographic primitives which we can design using hash functions for which the security proof can be given only in the Random-Oracle model. And the primitive that we are going to see here is what we call as commitment schemes. It is basically a two party

primitive involving a sender and a receiver. Basically when we say that we have a commitment scheme, it consists of 2 protocols or 2 phases, a commitment phase and an opening phase.

So, what happens in the commitment phase? In the commitment phase, sender has a message which is private and known only to the sender which we denote by m, which can be a bit string of some arbitrary length and in the commitment phase basically sender wants to commit the message to the receiver. So, you can imagine that in the commitment phase the sender computes a fixed length commitment, which we denote by c, for the message m and this commitment c is given to the receiver R.

So, you can imagine that this commitment c is like a sealed envelope or a sealed box inside which the sender has put the message m and given it to the receiver R. And the security property that we require from this commitment phase is that if the sender is honest and the receiver is corrupt, then by looking into the commitment c or by looking into the envelope c or in the box c, the receiver cannot find out what exactly is the message m which has been committed in the commitment c. So, that is a commitment phase of a commitment scheme.

 (Refer Slide Time: 08:15)

Commitment Schemes

❑ **Two entities** : a sender S and a receiver R
  ❖ A **commitment phase** --- protocol Com
  ❖ An **opening phase** --- protocol Open

Protocol Open

Opening information

S R

❑ 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**

Now, let us see what happens in the opening phase. So, imagine receiver has received a commitment of some unknown message, available with the sender, in the opening phase. The sender provides or reveals the opening information to the receiver using which the receiver opens the commitment and the receiver verifies whether the opening information is correct or not and based on that, it either accepts or rejects the revealed message, that is there in the commitment. And the security property that we require from the opening phase is as follows.

Now, we consider a corrupt sender and an honest receiver. So, remember, for the hiding property, we considered an honest sender and a corrupt receiver, but when we are considering the binding property with respect to the opening phase, we consider the case when the sender could be corrupt and the receiver is honest. So, what exactly the binding property means is that a corrupt sender should not be able to open a commitment c in two different ways.

What it means is that it should not be possible for a corrupt sender to commit a message say m during commitment phase and later on provider revealing information which ends up opening that commitment c to another message m'. That means, once sender has fixed his commitment and he has decided what exactly to commit, later during the opening phase he should not be able to change his mind.

(Refer Slide Time: 10:18)
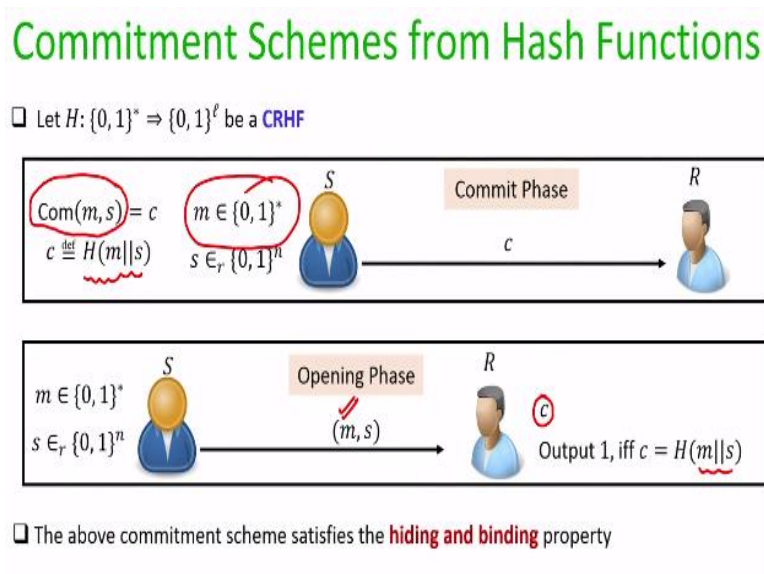
So, now let us see how exactly we can design commitment schemes using hash functions. I stress that that this is not the only way by which we can design a commitment scheme. Later on, when we will be discussing the number theoretic hardness assumptions, we will see how we can design commitment schemes based on cryptographic hardness assumptions based on number theoretic problems, but right now we are trying to give a scheme based on a hash function.

So imagine you are given a collision--resistant hash function H and the commit phase or the commitment phase for the commitment scheme that we are going to design is as follows. So the message that the sender wants to commit is some message m. To commit this message,

what sender does is, it picks randomness of some suitable size, denoted by s. So, basically s is a uniformly random string of size n bits, where n is some security parameter and the commitment of the message m is computed as follows. Basically, sender evaluates or computes the hash of the input message concatenated with the randomness which we denote as c and we call this protocol as protocol Com, with respect to the input m and randomness s and the commitment information c is given to the receiver. That is the commit phase. During the opening phase, the receiver has a commitment already available to it and it wants to now see what exactly it corresponds to, based on the opening information which the sender is going to provide.

So, what sender provides in the opening phases? The message, randomness which it might have used in the commit phase to commit message m in the commitment c and once the message, randomness comes to the receiver, what the receiver basically does is, it recomputes the commitment by hashing the input message concatenated with randomness and checks whether the recomputed commitment matches the commitment c, which the receiver already has received during the commit phase.
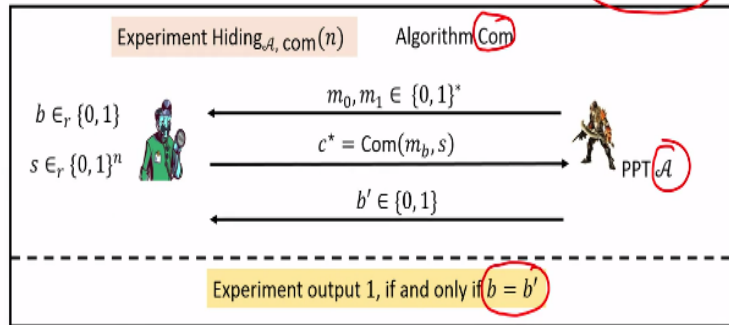
If it matches, then output is 1, that means accept the message m. Whereas if the recomputed commitment does not match the existing commitment c, then the receiver rejects the message m and outputs 0. So, that is a very simple construction of a commitment scheme based on a collision-resistant hash function. To commit a message, concatenate the message with some appropriate randomness and hash it. To open the message, you basically open the message or you reveal the message along with the randomness which you have used to compute the commitment.

Now, what we are going to prove is that this commitment scheme that we have designed here satisfies the hiding property and binding property and before we go into the proof, let us formally define what exactly we mean by the hiding property and the binding property in the context of a commitment scheme.

(Refer Slide Time: 12:56)

Hiding and Binding : Formal Definitions

❑ **Hiding property :** c should not **reveal anything** about m --- experiment $\text{Hiding}_{\mathcal{A}, \text{com}}(n)$

Experiment $\text{Hiding}_{\mathcal{A}, \text{com}}(n)$ Algorithm Com

$b \in_r \{0,1\}$

$s \in_r \{0,1\}^n$

$\xleftarrow{\quad m_0, m_1 \in \{0,1\}^* \quad}$

$\xrightarrow{\quad c^\star = \text{Com}(m_b, s) \quad}$ PPT $\mathcal{A}$

$\xleftarrow{\quad b' \in \{0,1\} \quad}$

Experiment output 1, if and only if $b = b'$

❑ Algorithm Com has **hiding property**, if for every PPT $\mathcal{A}$, there is a **negligible function** $\text{negl}(n)$:

$$\Pr[\text{Hiding}_{\mathcal{A}, \text{com}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n) \quad \approx \quad \begin{array}{l} | \Pr[\mathcal{A} \text{ outputs } b' = 1 | b = 1] \\ - \Pr[\mathcal{A} \text{ outputs } b' = 1 | b = 0] | \leq \text{negl}(n) \end{array}$$

So, remember, the hiding property informally requires that if the sender is honest and the receiver is corrupt, then by looking into the commitment c, the receiver should not be able to identify or learn what exactly is the message m, which is has been committed in the commitment c. And this is modeled by an experiment which we call as the hiding experiment. So, the rules of the hiding experiment are as follows. So, we have the description of a publicly known commitment algorithm, which is going to be used by the sender to commit some message.

We have an adversary A who wants to basically learn the value of the message which has been committed in a commitment, even by knowing the description of the algorithm com. So, the game is played between the adversary and the experiment or a challenger and the rules of the game are as follows. So, basically the adversary submits a pair of messages m_0, m_1 and the challenger basically does the following. It randomly decides one of these two messages by tossing a fair coin. Once it has decided what message to commit, it picks a randomness of appropriate size as per the commitment algorithm com and it commits that selected message m_b under the randomness s. Now, the challenge for the adversary is to identify whether $c^\star$ is a commitment of the message m_0 or whether $c^\star$ is a commitment of the message m_1. So, what this basically experiment models is that if receiver is corrupt, then based on the commitment that it sees from an honest sender, it should not be able to distinguish apart whether it is seeing a commitment of m_0 or whether it is seeing a commitment of m_1, where the receiver is given the power that it already knows that the commitment could be either the commitment of m_0 or m_1. So, the output of the adversary is a bit, namely whether it feels the commitment $c^\star$ is the commitment of m_0 or m_1 and our security definition is as follows. We say that output of the

experiment is 1 or the adversary has won the game, if it could correctly identify the message which has been committed in $c^\star$. Namely, it outputs $b' = b$. And we say that formally a commitment's algorithm com satisfies the hiding property if for every poly-time adversary participating in this experiment, there is some negligible function such that the probability that adversity can win the hiding experiment is upper bounded by half plus the negligible quantity. Another way to put the same condition is that the distinguishing advantage of the adversary should be upper bounded by some negligible function.

That means, it does not matter whether the commitment is for message m_1 or whether the commitment $c^\star$ is for the message m_0. In both the cases, the response of the adversary is almost identical except with some negligible function. And in both the definitions, the probability is taken over the random choice of the bit b and the randomness s and the random choice of the adversary with which it could decide the message pair m_0, m_1. So, on a very high level, this indistinguishability based definition of commitment might look like the indistinguishability based definition of encryption schemes that we have seen. The only difference here is that we do not deal with any keys here. So that is why we do not have any training phase and post-training phase and so on. The goal of the adversary is basically to just distinguish apart a commitment of m_0 from a commitment of m_1.

(Refer Slide Time: 16:44)
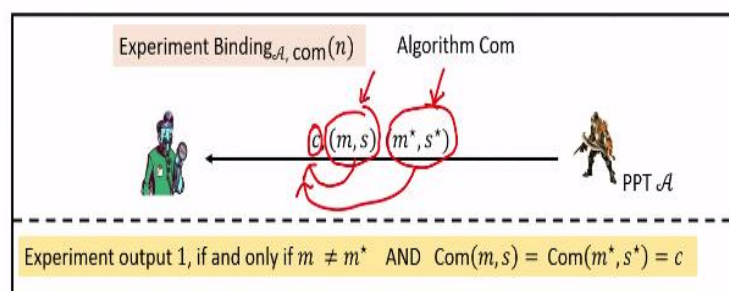
Now let us formally define the binding property. And remember informally the binding property demands that if the sender is corrupt and the receiver is honest, then it should not be possible for a corrupt sender to open an existing commitment c via 2 different messages. And

this is simply modelled by an experiment, where the adversary's goal is to come up with a commitment c and a pair of message, randomness where the first message randomness pair is (m, s) and the second message randomness pair is $(m^\star, s^\star)$, such that the commitment of the message m with the randomness s, and the commitment of the message $m^\star$ and the randomness $s^\star$ are the same, namely c, even though the message m and message $m^\star$ are different. If that is the case, then we say that the adversary has won the game. So, when we say that the sender wants to break the binding property, its goal is basically to come up with some message m and the corresponding randomness s. Another message, say $m^\star$, different from m and an appropriate randomness $s^\star$, such that when you commit the message m with the randomness s, you obtain the commitment c, as well as when you commit the message $m^\star$ with the randomness $s^\star$, you again obtain the commitment c. If that is the case, if that is possible, then what the corrupt sender can do is during the commit phase, it could commit a message m with some randomness s, but it when it comes to reveal the message which has committed in c, it can change its mind and say I have committed $m^\star$ by showing the appropriate randomness $s^\star$.

What we want from a commitment scheme satisfying the binding property is that the probability that a computationally bounded or computationally corrupted S is able to come up with such (m, s) and $(m^\star, s^\star)$ is upper bounded by some negligible function. So formally, we say that a commitment scheme has the binding property, if for every poly-time adversary A participating in this experiment, there is a negligible function, such that the adversary can win the game or it can break the binding property is upper bounded by some negligible probability.

Notice that we are not putting in this definition that a success probability of the adversary should be upper bounded by half plus negligible because here the goal of the adversary is not to distinguish a commitment of m_0 from a commitment of m_1. Its goal is to basically come up with a pair of colliding message, randomness with both of which could give you the same commitment and there is always a guessing strategy by the adversary where the adversary could guess (m, s) and $(m^\star, s^\star)$, which indeed gives you the same commitment c, as per the commitment algorithm.

The success probability of this guessing adversary is nonzero. So that is fine as per the definition. The best we can hope for is that the success probability of any adversary coming up

with a bad (m, s) and a bad $(m^\star, s^\star)$, giving the same commitment c, is upper bounded by some negligible quantity.

(Refer Slide Time: 19:52)
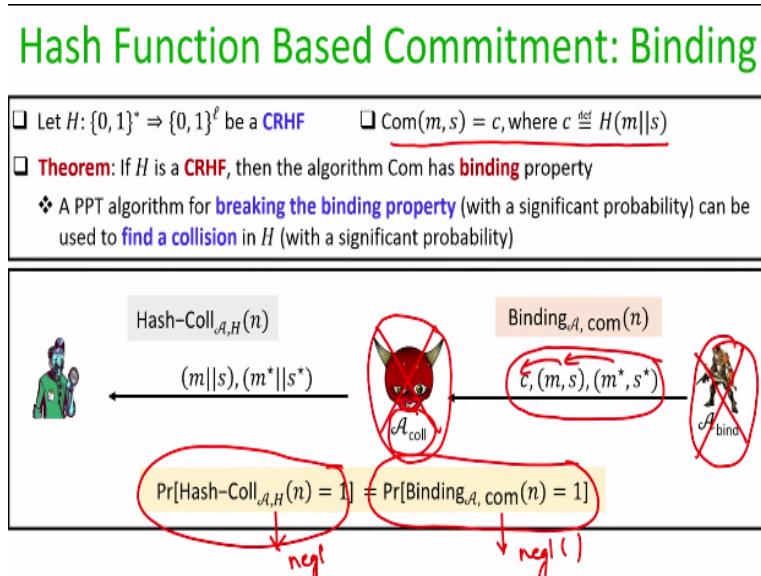
So now let us come back to the hash function based commitment scheme that we have designed and what we are now going to formally prove is that that commitment schemes satisfies the binding property and the hiding property. So the binding property will simply be based on the collision-resistance property of the underlying hash function. So assume we are given a collision-resistant hash function H, here is how we have constructed our commitment scheme. So, to commit a message m with some randomness s, basically you have to compute the hash value on the input m concatenated with s.

We can formally prove that if the underlying hash function is collision resistant, then indeed the algorithm com that we have defined here has the binding property. Namely, what we can formally prove is that any poly-time adversary who can break the binding property with a significant probability, with almost the same probability, it can be used to find out a collision in the underlying hash function. But that is a contradiction to the assumption that our hash function is collision resistant.

More formally, assume we have an adversary, a poly-time adversary who can break the binding property against the commitment scheme that we have designed. Using that adversary, we can show another adversary, which can find the collision in the underlying hash function. So what does adversary $\mathcal{A}_{coll}$ does is it invokes the existing algorithm $\mathcal{A}_{bind}$, which can break the binding

property. And basically the adversary who can break the binding property, he comes up with this set of triplet. Namely it comes up with a commitment c, a message, randomness pair (m, s) and another message randomness pair $(m^\star, s^\star)$. And what this adversary $\mathcal{A}_{coll}$ does is, it outputs a pair of collision namely m concatenated with s and $m^\star$ concatenated with $s^\star$. And it turns out that if indeed the binding adversary $\mathcal{A}_{bind}$ has broken the binding property, namely the triplet that he has produced breaks the binding property, that means, (m, s) as well as $(m^\star, s^\star)$ gives you the same commitment c as per the algorithm com. Then indeed, the inputs m concatenated with s and the inputs $m^\star$ concatenated with $s^\star$, constitutes a collision for the underlying hash function. That means, what we can say is that the probability that our collision finder wins the collision-resistant experiment is exactly the same with which our adversary who can break the binding property can break the binding experiment. But since we are assuming that our hash function is collision-resistant, then we know that quantity on your left hand side is some negligible function. No poly-time adversary could come up with a collision in polynomial amount of time. That means, this adversary $\mathcal{A}_{coll}$ does not exist and that automatically means that the adversary $\mathcal{A}_{bind}$ also does not exist. That means, only with negligible probability, the so called adversary $\mathcal{A}_{bind}$ that we have defined exist. It cannot break the binding property with significant probability.

(Refer Slide Time: 22:56)
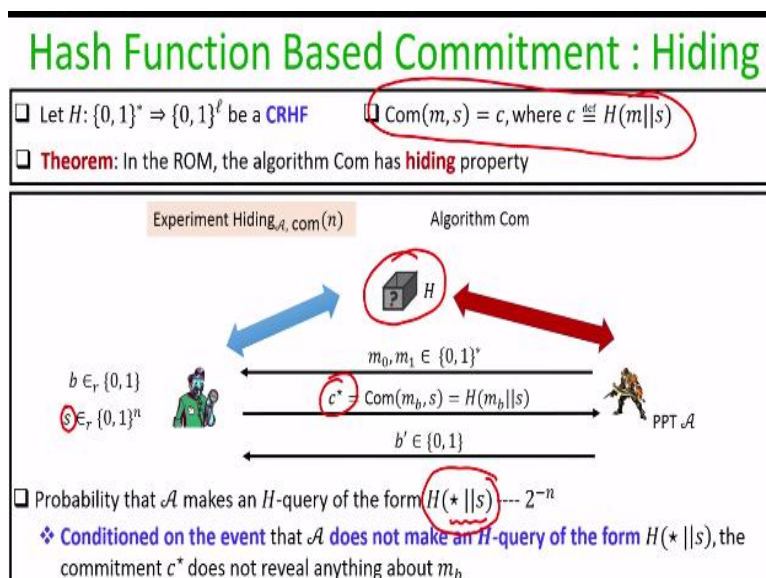
Now let us prove the hiding property of the commitment scheme that we have designed and it turns out that just by using the standard properties of the hash function, we cannot prove the hiding property. But interestingly we can prove that if we instantiate this commitment scheme in the Random-Oracle model, then in the Random-Oracle model, we can prove that this

commitment scheme satisfies the hiding property. So, let us go into the details. Now, since we are going into the Random-Oracle model, we have to modify the hiding experiment.

So remember in the hiding experiment, the rules of the game were as follows. The adversary simply throws a pair of messages (m_0, m_1) and one of them is randomly committed and the challenge for the adversary was to identify whether it is seeing a commitment of m_0 or whether seeing a commitment of m_1. But now since we are going to instantiate the commitment using a hash function which is taken in the Random-Oracle model and in the Random-Oracle model, each entity in the system is going to have oracle access to the function H, we have to incorporate the oracle access to this oracle H in the experiment.

So, the modified experiment is as follows. So, the adversary basically submits a pair of messages and the challenger or the experiment decides one of the messages to commit randomly. So, it decides a suitable randomness. Once it has decided what message to commit and now it throws the commitment of that message as per the com algorithm, and to commit that message m_b, basically the challenger has to evaluate $H(m_b||s)$.

But since we are assuming that we are in the Random-Oracle model, the output $H(m_b||s)$, is obtained by the experiment making an oracle call to the oracle H, on the input m_b, concatenated with s. And as per the rules of the Random-Oracle model, adversary A is not allowed to see the oracle calls that experiment has made to the Oracle H. Now, since we are in the Random-Oracle model, this adversary A is also allowed to make polynomial number of oracle queries to this oracle H on any input of its choice.

Then finally, it outputs whether it has seen a commitment of the message m_0 or whether it has seen a commitment of message m_1. I stress here that in this whole experiment, that function H is a random oracle with neither the experiment nor the adversary having access to the code of this oracle H. So, since we want to now prove that what is the probability that adversary A could significantly output $b' = b$, since we are in the Random-Oracle model, we have to basically argue that what is the probability that adversary has made oracle query with inputs of the form "something" concatenated with the randomness s.

So, remember that the commitment $c^\star$ is the output of the oracle H on some message m_b, where m_b could be either m_0 or m_1, which is already known to the adversary A, because adversary itself has thrown that pair of messages. What the adversary does not know is the other part of the input, which has been used to compute the commitment $c^\star$. Namely it does not know the value of the randomness s, and since the randomness s is a uniformly random value, the probability that adversary A makes an H query of this form, namely H query where the input of the query is "something" concatenated with the randomness s which has been used by the challenger, is maximum $\frac{1}{2^n}$. Conditioned on the event that adversary has not made any query of this form, namely H query of the form "something" concatenated with s, the value of $c^\star$ is going to be a uniformly random value from the viewpoint of the adversary and it does not reveal anything whether it corresponds to the H output or H value for the message m_0 or for the message m_1.

That means with almost same probability, from the viewpoint of the adversary, it could be a commitment of m_0 or it could be a commitment of m_1. So, now you can see that as soon as we take this whole construction, this simple construction based on the hash function, and take it to the Random-Oracle model, we can very conveniently give a proof that indeed this construction satisfies the hiding property. But if we do not take this construction in the Random-Oracle model, we cannot complete the security proof for the hiding property just based on the standard properties of the hash function.

(Refer Slide Time: 27:23)

Key-Derivation from Hash Functions

- m: pre-shared, highly un-predictable data
  - ❖ **Need not** be a uniformly-random bit-string
  - ❖ Ex: a common password, consisting of 28 uniformly-random upper case letters
- Goal: to locally derive a **random secret key** from m (for example, a 128 bit AES key)
  - ❖ Ad hoc mapping of m to bit-strings **need not be secure**
  - ❖ Ex: taking the ASCII bit-representation of the first 16 characters of m
    - ➤ ASCII representation of letters in the range A to Z has "010" as the **first three bits**
    - ➤ 37.5% bits of the resultant key are fixed and publicly known

So, now, let us see some other primitives which we can design based on hash functions, whose proof we can give only in the Random-Oracle model. And this primitive is what we call as key-derivation function. Later on, when we will be discussing public key cryptography, we will be extensively encountering this primitive. So what exactly is the goal of key-derivation function? So the scenario is as follows. Imagine we have two parties, say a sender and a receiver and they have some pre-shared data, which is highly unpredictable. That highly unpredictable pre-shared data, need not be in the binary form, it may not be a binary string and not only that, it need not be a uniformly-random bit string. For example, you can imagine that S and R might have pre-shared a common password consisting of 28 characters, where all the characters are uppercase letters, namely they belong to the set $\{A, \cdots, Z\}$ and each of the characters in this set occurs with equal probability and with uniform probability, so that is the scenario.

The goal here is basically sender and receiver locally want to apply some function on their pre-shared data m and derive a random secret key. For example, they might be interested to derive a 128-bit uniformly random AES key from the pre-shared data m, which is highly unpredictable and not known to anyone except the sender and receiver. I stress here the goal is to do that by locally applying some publicly known function. Sender and receiver do not want to interact, they just want to apply some function on the pre-share data m and want to derive some random secret key.

It turns out that if sender and receiver apply some ad hoc mapping or ad hoc function to the pre-shared data, then resultant output need not be uniformly random bit string. For example, let us take the case where both sender and receiver takes the ASCII bit representation of the first 16 characters of m. Why is it 16 characters because the ASCII representation of each character will be 8 bits and since the goal of the sender and the receiver is to derive a 128 bit key, it suffice for the sender and the receiver to just concentrate on the first 16 characters of their common password and convert the 16 characters into the ASCII representation and take that to be the resultant AES key. Intuitively, you might feel that since the password consists of uniformly random uppercase characters, when those uppercase characters are converted into their ASCII representation, the corresponding binary representation also will be uniformly random, but it turns out that this ad hoc mapping is simply not secure.

It is not secure in the sense that the resultant 128 bit strings that sender and receiver are going to derive is not a uniformly random 128 bit string. This is because if you see the ASCII representation of the letters for the uppercase letters A to Z, then all those ASCII representation starts with 010 as the prefix. So the ASCII representation of A starts with 010 and so is the ASCII representation of B and so is the ASCII representation of Z and so on. So, it does not matter what exactly are the first 16 characters of the password m that is available with the sender and the receiver, if they convert the first 16 characters into their ASCII representation, then an adversary who knows that sender and receiver are basically taking the ASCII representation of the first 16 characters of the common password, the adversary knows that 37.5% of the resultant key are fixed and known to any third party. So that means, out of the 128 bit keys that sender and receiver has derived, 37.5% bits of the key is already known to any third party in the world and that is why we cannot claim that the resultant 128 bit key is a uniformly random 128 bit key.

So now you can see that we cannot just apply ad hoc mapping to highly unpredictable pre-shared data and hope that the resultant mapping gives us uniformly random key.

(Refer Slide Time: 31:43)
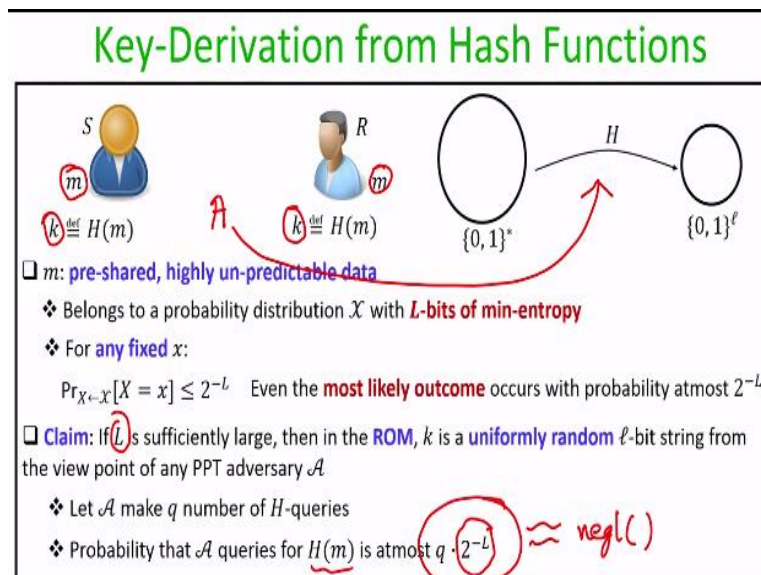
So that is the goal of key-derivation function. So key-derivation function, as the name suggests, it derives key from an already existing pre-shared highly unpredictable data. And we can design key-derivation function in varieties of ways, but what we are going to see is the highly practical instantiations of key-derivation functions based on hash functions. So the idea here is imagine that S and R have some pre-shared highly unpredictable data.

So till now, I was just giving you an informal description of what we mean by pre-shared highly unpredictable data. So, let us formally define it. So, I say that it is highly unpredictable in the sense that it belongs to a probability distribution $\mathcal{X}$ with L-bits of min-entropy. What exactly that means? Well, that means that if I pick a value as per the probability distribution $\mathcal{X}$, then the probability that the picked value is some given x, is upper bounded by 2^{-L}, that is what is the interpretation of this notation.

So this means that I am picking some value as per the probability distribution and the picked value is some random value, so that is why this X is the random value denoting the value that I am picking as per the probability distribution $\mathcal{X}$. Now, I am arguing that what is the probability that that random value takes the value equal to x, for a given fixed x. If that probability is upper bounded by 2^{-L}, then I say that this probability distribution $\mathcal{X}$ has L-bits of minimum entropy.

Informally, what it means is that even the most likely outcome as per this probability distribution $\mathcal{X}$, can occur with probability at most $\frac{1}{2^L}$. So what basically this L-bits of min-entropy captures, it captures the probability with which an adversary can guess the value picked by a sender or receiver as per the probability distribution $\mathcal{X}$. That means if adversary already knows that certain outcomes are more likely to occur and certain values are less likely to occur. And If the adversary's strategy is to always guess that it is the most likely value that a sender has picked, then even in that case, the probability that adversary's guess is correct is $\frac{1}{2^L}$. That is the idea of this notion of L-bits of min-entropy. So assume both sender and receiver have access to a publicly known hash function, and they have some pre-shared data m, which has L-bits of min-entropy, then to derive a key what basically sender and receiver can do is, they can just evaluate the hash function on the pre-share data m.

The resultant output which I denote as k, will be the pre-shared binary string of size ℓ-bits, which will be now available to the sender and the receiver. So that is a key-derivation function in its most simplest form and now, we can argue that if we model the function H as a Random-Oracle and if we take this construction and instantiate in the Random-Oracle model and if the value of L is sufficiently large, then the resultant output little k is indeed a uniformly random ℓ-bit string from the viewpoint of any poly-time adversary.

Imagine there is an adversary A sitting between the sender and the receiver, and since we are now in the Random-Oracle model, what the adversary is going to do is, adversary will be now making polynomial number of oracle queries to this H and adversary would not be knowing what exactly is the underlying oracle H. So, until and unless adversary does not query this oracle H on the exact value m, which is pre-shared between the sender and the receiver, the value of k, which sender and receiver are outputting here will be uniformly random from the viewpoint of the adversary.

Now, what is the probability that adversary indeed queried for H of m, given that it has made q number of queries to the oracle? Well, if my probability distribution or if the data m which has been pre-shared between the sender and receiver has big L-bits of min-entropy, then what the adversary can hope for is that every time it can ask the value of this H oracle on the most favorable data that it feels that the sender and the receiver might have picked as per this probability distribution

So, the probability that in each single query, it has actually queried for the exact m, is $\frac{1}{2^L}$, because that has come from the definition of L-bits of min-entropy and since the adversary has done q number of queries, with maximum probability $\frac{q}{2^L}$, adversary A might have queried for the input H of m from the oracle H. And if we assume that big L is sufficiently large and q is any how polynomially bounded in the security parameter, this is some negligible function in the security parameter.

So, since adversary has not queried for H of m, except with negligible probability, from the viewpoint of the adversary, the resultant key k is going to be a uniformly random ℓ-bit string and that ensures that now sender and receiver can safely use the derived key k for any cryptography primitive as the key. Say for instance if they want to use the key k as the key for AES, they can safely use it. So, now, you can see that how this highly efficient key-derivation function based on hash function can be proved secure if we go in the Random-Oracle model.

But if we just go into standard model and use the standard properties of the hash functions, namely collision-resistance, we cannot prove that the so called key derivation function is indeed a secure key derivation function. So that brings me to the end of this lecture. Just to summarize in this lecture, we have continued our discussion on the Random-Oracle model. We

have seen that we have certain arguments in the favor of Random-Oracle model, we have certain arguments against Random-Oracle model and so on.

We have seen instantiation of 2 important cryptographic constructions or primitives namely commitment schemes and key-derivation functions based on hash functions, whose proof we can give only in the Random-Oracle model. Thank you.