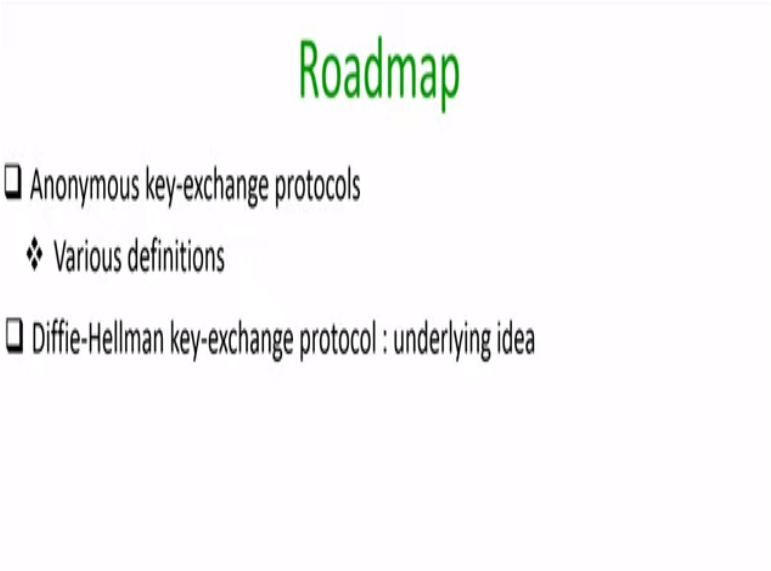
**Foundations of Cryptography**
**Dr. Ashish Choudhury**
**Department of Computer Science**
**Indian Institute of Science – Bangalore**

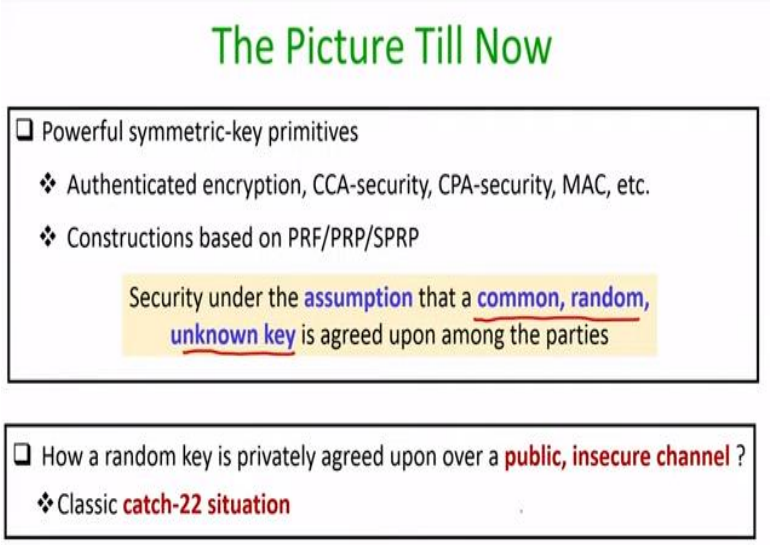
**Lecture – 36**
**Key Exchange Protocols Part 1**

**(Refer Slide Time: 00:35)**




Hello everyone, welcome to this lecture. the plan for this lecture is as follows, in this lecture, we will introduce the notion of anonymous key exchange protocols. We will see the various definitions and we will see the underlying ideas involved behind the seminal key exchange protocol due to the Diffie and Hellman.

**(Refer Slide Time: 00:47)**




So just to recap, the picture till now is as follows. Till now we had seen several powerful symmetric-key primitives, both in the passive world as well as in the active world. We have seen various notions of security, like authenticated encryption, CCA security, CPA security; we had seen other primitives for integrity and authentications like message authentication code. And we had seen constructions of all these notions of security, where constructions are based on pseudo random functions, pseudo random permutation, strong pseudo random permutations and so on. So a basic fact about all the constructions that we had seen until now is that the security for all these cryptographic primitives that we have seen, holds under the assumption that a common random unknown key is already agreed upon between the parties, specifically the sender and the receiver.

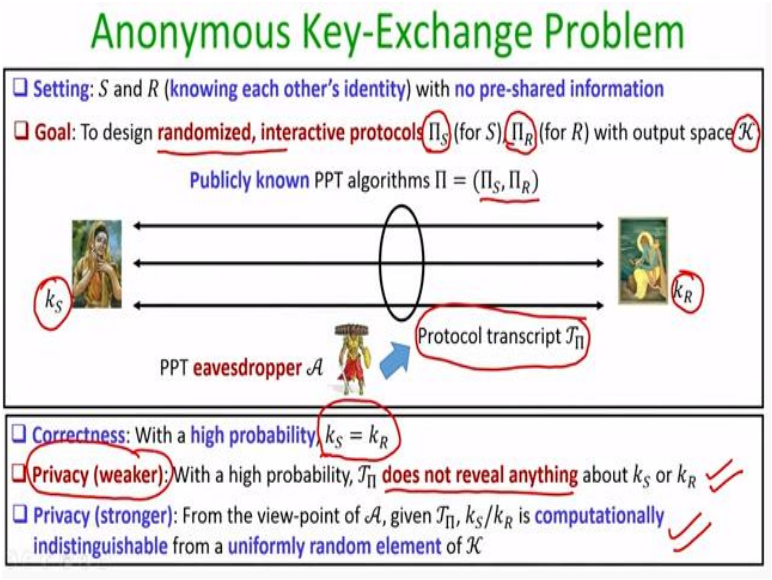
So now the question that we want to address here is that how a common random key is privately agreed upon over a public insecure channel? So till now we were assuming that somehow the key agreement has already happened and given that the key agreement has already happened, we saw how to design authenticated encryption schemes, CCA secure encryption schemes, CPA secure encryption schemes and so on.

But now we would like to address the main problem. We would like to address the question that how at the first place itself, a random key is privately agreed upon over a public and insecure channel. And this sounds like a classic catch-22 situation right? Because if you have a mechanism where sender and receiver can securely exchange a unknown key which is known only to the

sender and the receiver over a public insecure channel, then using that mechanism, they at the first place itself can do the secure communication also.

So what exactly catch 22 situation means is, where you have a scenario, where say for example you are a fresh graduate, comes out of the college and applies for a job and when the candidate faces the job interview, he faces the question that do you have any experience or not? But at the first place the candidate will get experience only if the job is given to the candidate. So that is what we mean by catch-22 situation. And we are exactly facing the same scenario here as well. We know how to do secure communication, assuming that we know how to securely exchange the key over an insecure channel.

**(Refer Slide Time: 03:16)**




So that brings us to the problem of anonymous key exchange. So let us define what exactly is the anonymous key exchange problem. So the setting is as follows: we have a sender and the receiver, and we assume that somehow, they know each other's identity, but did not have any pre-shared information. So you might be wondering that how it is possible for a sender and the receiver who are meeting for the first time that they know each other's entity. Later on we will remove this assumption as well. But for the moment, to make the description of the anonymous key exchange problem simpler, let us assume that both sender and the receiver know each other's entity. So the

goal here is to design a pair of protocols, which I denote as $\Pi_S$ for the sender, and $\Pi_R$ for the receiver, which are now interactive protocols.

And the output space of this protocol will be some $\mathcal{K}$ and the goal is to design such a pair of protocols one for the sender and one for the receiver such that according to the individual protocols sender and receiver communicate over an insecure channel and at the end of their respective protocols, sender and receiver output some respective keys. So the output of the sender I denote as $k_S$ and output of the receiver I denote as $k_R$.

And we assume in this problem definition that we have an eavesdropper, a computationally bounded eavesdropper, who is getting access to the entire communication that is happening between the sender and receiver. So the adversary is aware of the pair of protocols using which the sender and a receiver are interacting. Namely it knows the steps of $\Pi_S$ and $\Pi_R$ and it also gets access to the information that is exchanged between the sender and the receiver, which we call as protocol transcript, which I denote as $\mathcal{T}_\Pi$. And notice that this protocol transcript is going to be a random variable, because the information which sender and receiver are going to exchange, they will depend upon the internal randomness with which the sender and receiver are going to invoke in the respective protocols $\Pi_S$ and $\Pi_R$. So it is not the case that sender and receiver will exchange the same set of values every time they invoke this protocol $\Pi$, because they are invoking a randomized protocol.

Now the properties that I require from this pair of protocols $\Pi_S$ and $\Pi_R$ are as follows. The first property is the correctness property, which says that with a very high probability, the individual outputs of the sender and the receiver should be the same. Namely output $k_S$ and output $k_R$ should be same. And now we can have 2 variants of security, which we can demand from these protocols $\Pi_S$ and $\Pi_R$.

The first definition is a weaker form of privacy, which I call as weak privacy, which requires that with very high probability, a computationally bounded adversary, even after knowing the protocol transcript does not learn anything about the output of the sender and the receiver. So here I am not demanding security in the indistinguishability sense; here the demand is in the sense that either the

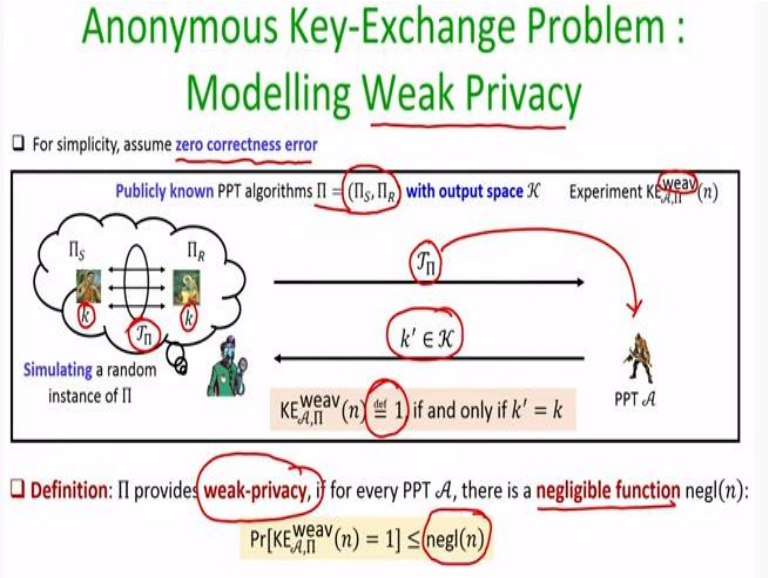
adversary should not learn the entire $k_S$ and the entire $k_R$. It is fine if the adversary learns "something" about $k_S$ and $k_R$. But the requirement here is that as an entirety, the outputs of the sender and the receiver should not be learnt by the attacker.

Whereas we can go for a higher notion of security, which I call that stronger privacy, which demands that from the viewpoint of that adversary, even if the adversary has access to the protocol transcript, from the viewpoint of the adversary the respective outputs of the sender and the respective output of the receiver, should be computationally distinguishable from a uniformly random element of the output key-space $\mathcal{K}$.

So that is a more stronger notion of secrecy. Because if you go for the stronger notion of secrecy, then it is not allowed that adversary learn something about the respective bits of $k_S$ and $k_R$; from the viewpoint of the adversary, the respective outputs of the sender and the receiver could be any candidate element from the underlying key-space $\mathcal{K}$.

So we will see construction satisfying both the weaker form of privacy and construction satisfying the stronger form of privacy. You might be wondering that why exactly we care to achieve weaker form of privacy. So that will be clear later on. Looking ahead, we will see constructions achieving weaker form of privacy, based on cryptograph consumptions which are mild. Whereas if you want to achieve key-exchange protocols which achieve stronger notion of privacy, then we have to go for cryptographic assumptions which are slightly stronger.

**(Refer Slide Time: 08:05)**




So we had seen intuitively what exactly weak-privacy and strong-privacy means. So now let us go ahead and model these requirements formally by an indistinguishability based game. And while giving the indistinguishability definitions, for simplicity we assume that we are considering key exchange protocols which has 0-correctness error. That means with probability 1, output of the sender and output of the receiver will be the same, so we are considering that kind of key exchange protocols.

So let us see first how to model the weak-privacy and remember the requirement of the weak-privacy states even if the adversary sees the whole transcript exchange between the sender and the receiver, the resultant output key which is obtained by the sender and the receiver should not be learned or should not be known to that adversary, in its entirety. And this requirement is modeled by this game.

So this game, which we call $\mathrm{KE}^{\mathrm{weav}}_{\mathcal{A},\Pi}(n)$, here KE stands for key-exchange and a superscript weav denotes weak eavesdropping or weak privacy, here the game is played between a challenger and a poly time adversary. And what the challenger or the experiment does is, basically it simulates a random instance of the key-exchange protocol Π. So the protocol Π is basically the pair of protocols $\Pi_S$ and $\Pi_R$, where the $\Pi_S$ protocol is going to be invoked by the sender and the $\Pi_R$ protocol is going to be invoked by the receiver.

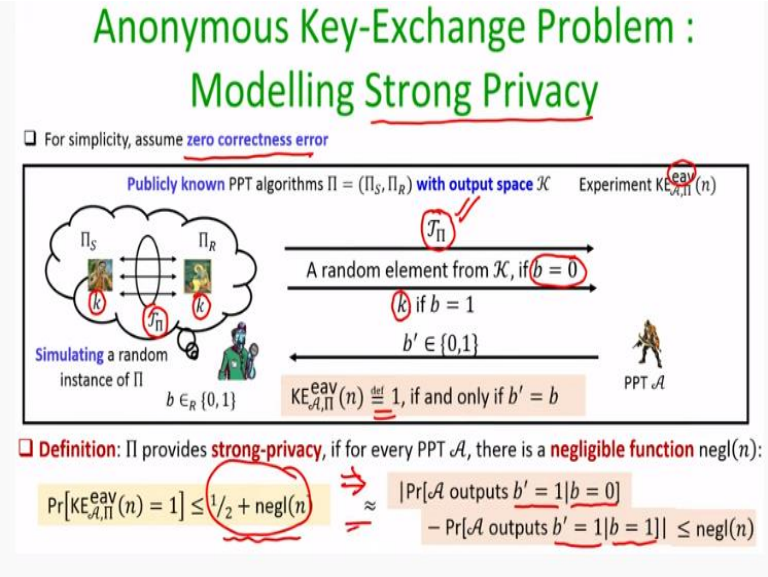
But as a collection, we call it as protocol $\Pi$. So the challenger basically simulates a random instance of the protocol $\Pi$, by playing the role of the sender and a receiver in its mind with their respective coins as per the protocol $\Pi$ and it generates a transcript $\tau_{\Pi}$. Now what it does is, it throws a challenge to the adversary, namely the transcript and this models the fact that in the real world, an adversary sitting between the sender and the receiver, would have observed a transcript which is generated by the sender and receiver by running the protocol $\Pi$.

Now the challenge for the adversary is that by seeing this transcript $\tau_{\Pi}$, it has to figure out what exactly is the value of key $k$, which sender and receiver would have obtained by running by the respect to this protocol transcript $\tau_{\Pi}$. So basically the adversary has to respond with a key from the key space, which I denote as $k'$. And the way we define the output of the experiment is as follows. We say that adversary has won the experiment, which is also denoted by saying that output of the experiment is 1, if and only if the guess of the adversary $k'$ is exactly equal to $k$. That means, the adversary A, without knowing the internal randomness of the sender and the internal randomness of the receiver and by just observing the protocol transcript $\tau_{\Pi}$, is able to come up with the exact key $k$, which sender and receiver are going to obtain by running with respect to this protocol transcript $\tau_{\Pi}$.

If that happens then we say that output of the experiment is 1. And the security definition is, we say that the key-generation protocol $\Pi$ has weak-privacy, if for any poly time adversary participating in this experiment, there exists some negligible function, such that the probability that adversary wins the experiment is upper bounded by some negligible function, where the probability is taken over the random coins of the challenger. Namely the random coins, with which it is simulating the role of the sender and the receiver. So notice that here the adversary does not have to distinguish between something. The goal of the adversary is to come up with the key which the sender and the receiver are going to obtain with respect to the protocol transcript $\tau_{\Pi}$. That is why the security definition will not have an expression of the form that adversary chances of winning the experiment is upper bounded by half plus some negligible function. The goal of the adversary is to come up with the exact key with which sender and receiver are going to obtain by running the protocol $\Pi$ and which is consistent with the protocol transcript. Also in this definition we allow the adversary to win the game with some negligible probability because there is always

a guessing adversary, who can just guess some candidate $k'$ from the key space and with non-zero probability it may turn out that $k'$ is exactly equal to $k$.

**(Refer Slide Time: 12:31)**




So now let us see how we can model strong-privacy. And remember the goal of strong-privacy is that an adversary who monitors the transcript exchanged between the sender and the receiver, should not be able to distinguish the resultant key which sender and receiver are going to obtain, from any uniformly random element from the key space. And again for modeling the strong-privacy, we assume key-exchange protocols where there is 0-correctness error.

This is again without loss of generality; its very straight forward to incorporate this condition as well in the security definition. So now the experiment is called $KE^{eav}_{\mathcal{A},\Pi}(n)$ because now we are not modeling weak-privacy. We are now modeling strong-privacy here and the rules of the game are as follows. The adversary is waiting for a challenge here and the challenge is again generated more or less in the same way as it was generated in the experiment for weak-privacy.

Namely the challenger here plays the role of the sender and receiver in its mind and invokes an instance of the protocol $\Pi_S$ for the sender and invoke in the instance of the protocol $\Pi_R$ for the receiver, with their respective randomness and it generates a protocol transcript $\tau_\Pi$. That protocol transcript is now given as a challenge to the adversary. So this models the fact that the eavesdropper

between the sender and the receiver would have eavesdropped and obtained the protocol transcript $\tau_{\Pi}$.

And now since we are trying to model the indistinguishability based notion of security in the context of key-exchange protocol, the challenger additionally throws a challenge as follows. It tosses a fair coin, with probability 1/2 it could be 0 or with probability 1/2 it could be 1. And now apart from the transcript, depending upon whether the coin toss is 0 or whether the coin toss is 1, the challenger either submits a random element from the key space to the adversary or the key $k$, which the sender and receiver would have obtained by running the protocol instance in the challenger's mind.

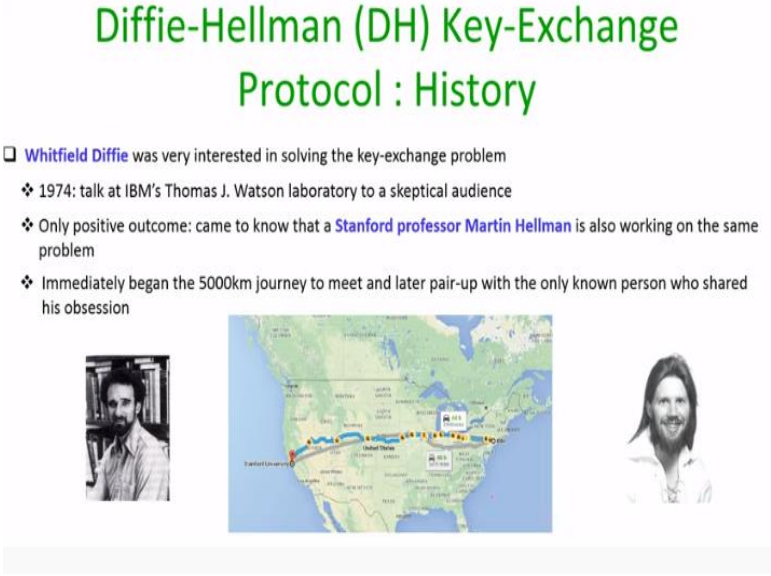
Now adversary does not know, whether the value from the key-space that it is seeing is a random element from the key-space or whether it is a key which the sender and the receiver would have obtained by running the protocol Π and obtaining the transcript. So the challenge for the adversary is to find whether he is in the method $b = 0$ or whether it is in the method $b = 1$. And it submits its response, namely a bit.

And the definition here is, we say that the adversary wins the experiment, which is also denoted as the output of the experiment is 1, if and only if adversary A could correctly identify whether he is seeing an element as per the method $b = 0$ or whether he is seeing an element from the key space as for the method $b = 1$. And the definition of strong-privacy here is, we say that a key-exchange protocol Π gives you strong-privacy, if for any poly time adversary participating in this experiment, there is some negligible function, such that the probability adversary wins the experiment is upper bounded by half plus some negligible function.

So now this is different from the definition of the weak-privacy. Because in the weak-privacy the goal of that adversary was to come up with the exact $k$ which is consistent, or which would have been obtained by the sender and receiver as per the transcript $\tau_{\Pi}$. But now here the goal of the adversary is to distinguish the $k$, which sender and receiver are going to obtain as per the protocol transcript $\tau_{\Pi}$, from a uniformly random element from the key-space.

So that is why now we are having a condition which is similar to the indistinguishability based definition and again we are putting this condition half plus negligible, because there is always a guessing strategy by the adversary and the guessing strategy of the adversary will be just to guess whether he is in the method $b = 0$ or whether it is in the method $b = 1$. And the success probability of that guessing strategy is 1/2. Apart from that we are willing to let the adversary win this experiment with some negligible function because we are in the computational world. An equivalent formulation of this notion of strong-privacy is that we say that the protocol $\Pi$ gives you strong-privacy, if for any poly time adversary participating in this experiment, the distinguishing advantage of the adversary is upper bounded by some negligible function in the security parameter. That means it does not matter whether the challenger has generated the challenge by the method $b = 0$ or whether he has generated the challenge by method $b = 1$. In both the cases the response of the adversary should be almost the same $b' = 1$, except with some negligible probability. And we can prove that both these conditions are equivalent to each other. Namely if you have a key exchange protocol satisfying the first condition then it also implies that it satisfies the second condition and vice versa. So depending upon our convenience we can use any of these two conditions.

**(Refer Slide Time: 17:41)**




So now we have the security definitions of key-exchange protocols and now you might be wondering that whether indeed it is possible to design key-exchange protocols, where sender and

receiver, without having any pre-shared information can do some communication over a publicly known insecure channel and ends up agreeing upon a key which is not known to any third party?

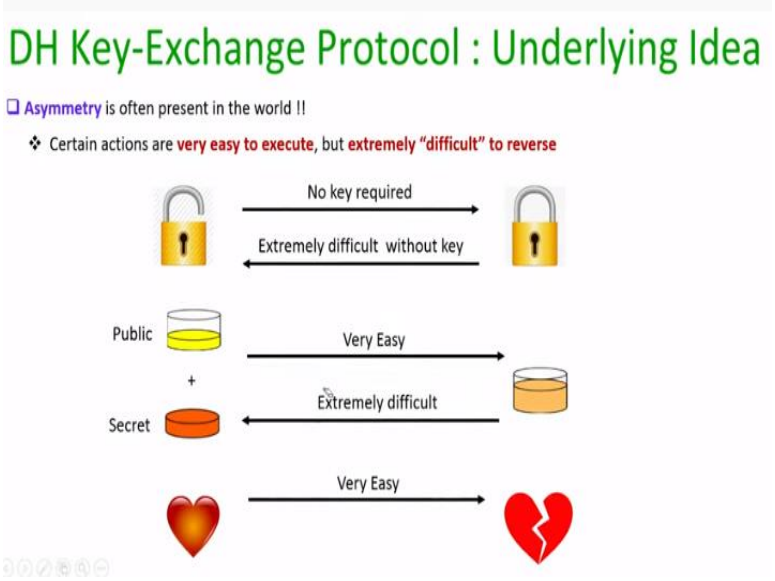
So on a very high level it might look like an impossible task because how it is possible? That is, an unknown sender and receiver who just know their identity and have no pre-shared information whatsoever, can do public interaction and agree upon something which is known only to them. But it turns out that we indeed have such key-exchange protocols. And the base of this key-exchange protocol is due to pioneer work by the Diffie and Hellman who are the first pair of cryptographers who came up with their seminal key-exchange protocol.

So in this lecture we are going to see the underlying ideas based on which their key-exchange protocol was developed. So before that let me tell you some fascinating history about how exactly their key-exchange protocol came into existence. So Whitfield Diffie was very interested in solving the key-exchange problem. And he strongly believed that indeed it is possible for a sender and a receiver to do public communication and agree upon a common secret key.

And in 1974 he gave a seminar at IBMs Watson laboratory to a skeptical audience, who were not buying his idea that indeed key-exchange is possible over a public channel. The only positive outcome which came out of that seminar is that he came to know that it is not only him, there is another person, a professor at Stanford, called Martin Hellman, who is also working on the same problem and trying to come up with public key-exchange protocols.

And as soon as Diffie came to know about this, he started some 5000 kilometer road journey to meet and pair up with Martin Hellman and that is how they started their work on coming up with a protocol for solving the key-exchange problem. And they work for two years rigorously and finally they came up with their groundbreaking Diffie-Hellman key-exchange protocol.

**(Refer Slide Time: 20:07)**




So let us try to understand the underlying idea which is there in the Diffe-Hellman key-exchange protocol. So the basic idea behind their key-exchange protocol is that there are several tasks in this world, which have asymmetry or they are asymmetric in nature. And asymmetry in the sense that they are very easy to execute. That means those actions are very easy to execute but extremely difficult to reverse.

So what I mean by that is imagine I give you a padlock in an open state. And if I ask you to lock it then you do not need any key. You just have to press the head of the padlock and from the open state you can easily take it to the locked state. But now if I give you the padlock in the locked state and I ask you to take it back to the open state then it will be extremely difficult for you if you do not possess the key for the padlock.

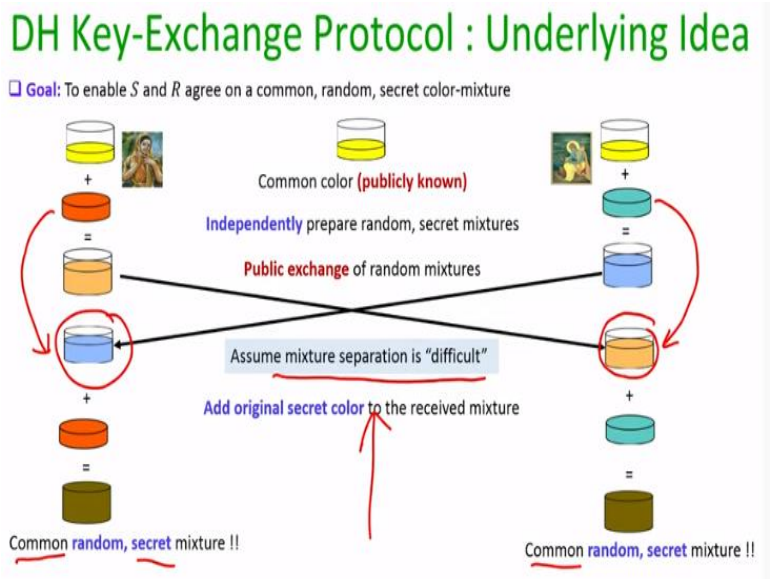
So in that sense this you have an action here where going from one state to another is extremely easy. But going back from the obtained state back to the original state is extremely difficult. It is not impossible. Remember that I am saying its "extremely difficult" to go back or reverse the action if you do not know the key. There might be other methods to reverse that action without the key, but those alternatives will be extremely difficult.

In the same way consider this task that you are given a publicly known color and say if you want to prepare up a secret mixture, then it is very easy for you to do that by taking that publicly-known

color and adding to that existing color, a secret color and then obtaining the mixture. So preparing the mixture is very easy for you. But if someone gives me this mixture and does not tell me what exactly was the secret color which was added to the public color to obtain this mixture, then it will be very difficult for me to find out or separate out this mixture into its constituents, namely the publicly known color and the secret color which I have added here.

So in that sense, preparing a mixture is easy but separating out the mixture to its individual components is an extremely difficult task. Again I stress I am not saying it is an impossible task. I am saying it is an extremely difficult task, Its very time consuming. And finally it is very easy to break anyone's heart by just saying some bad words. But once someone's heart is broken, then it is very difficult to to win back the love and the confidence of that person. In that sense going from this state to this state is very easy, but going back is extremely difficult.

**(Refer Slide Time: 22:50)**




So Diffie-Hellman thought of several ideas regarding how exactly they can solve the key-exchange problem and when they came across this idea that there are certain tasks which have asymmetry involved with them. Then based on that concept, they thought of this idea of solving the key-exchange problem. So we are not going to see the exact mathematical details of the Diffie-Hellman key exchange protocol. Those details we will discuss in subsequent lecture. I am just trying to give you the intuition that what exactly was the underlying idea here.

So the goal here is for the sender and the receiver to agree upon a common random secret mixture, which will be known only to the sender and the receiver, who have no pre-shared information to begin with. So here is how the protocol will proceed.

Both sender and receiver will start with some common color, publicly known. And what they do is, they individually prepare some random secret mixtures, by adding a secret color independently picked and adding it to the publicly known color. So both of them are independently taking the secret colors and preparing a mixture. And once they obtain their respective mixtures, what they do is, they publicly exchange their mixtures with each other or some public channel.

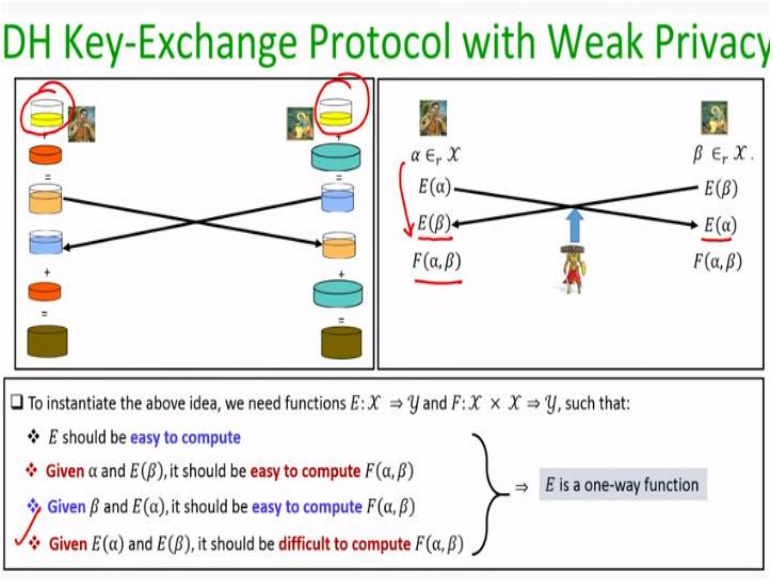
And when these mixtures are exchanged, we assume here that the mixture separation is extremely time consuming, extremely difficult, it is a computationally heavy task, that is an assumption I am making here. Now once both sender and receiver obtains their respective mixtures, what they do is, to the mixtures that they have obtained from the other party, they add their respective secret colors with which they started the protocol.

So for instance sender on receiving this mixture it takes a secret color and adds to it. And in the same way, the mixture that the receiver has obtained from the sender, which is the senders mixture, the receiver adds its own secret color. And now you can see what both sender and receiver are going to obtain. Both sender and receiver are going to obtain a final common mixture because this final common mixture is a mixture of 3 colors, the publicly known color, the senders secret component and the receivers secret component.

It does not matter in what order you mix them. Irrespective of the order in which you mix them, the resulting mixture is going to be the same. And that ensures that both sender and receiver are going to obtain a common mixture. And now why it is secret? It is secret, because we are assuming here that when sender and receiver are exchanging their respective mixtures, mixture separation is extremely time consuming.

That means if there is an eavesdropper who is actually monitoring the communication happening between the sender and the receiver and it knows the entire description of the protocol steps. And what the adversary does not know, it would not know the exact secret components which the sender and the receiver have individually added. And even after seeing the mixture which sender and receiver are exchanging, it will be difficult for the adversary to separate out or find out that respective things, which the sender and receiver have added. And that ensures that the final mixture which sender and receiver are agreeing upon is already a secret color or a secret mixture which will not be known to anyone. So that is the fundamental idea behind that Diffie-Hellman key exchange protocol. So of course this is an idea behind the key-exchange protocol. Now we have to convert this idea into an algorithm.

**(Refer Slide Time: 26:20)**




Specifically, a mathematical algorithm. So what we are going to discuss now is, how we can convert this idea of exchanging colors and coming up with a common secret mixture known only to the sender and the receiver, into a key-exchange protocol with weak-privacy. So I am retaining the blueprint of the key-exchange protocol based on exchanging color mixtures and now what we have to do is, we have to come up with an instantiation or replacement of each of the steps, where colors are used, mixtures are prepared and exchange and so on, by some mathematical steps.

So to instantiate the idea, what we need here is, we need some special functions which I denote as $E$ and $F$. So the function $E$ is from the set $\mathcal{X}$ to $\mathcal{Y}$, so it is a one input function. Whereas the

function $F$ is a two input function, taking two inputs from the set $\mathcal{X}$ and giving you an output from the fancy set $\mathcal{Y}$. The requirements from the functions are as follows.

So the function $E$ should be easy to compute and the second requirement from the function $E$ and function $F$ is that if you are given any $\alpha$ from the domain of the function $E$, and the value of the function $E$ on any input $\beta$, then even without knowing the value $\beta$, just based on the knowledge of $\alpha$ and the function output of $E$ on the value $\beta$, it should be easy to compute the value of the function $F$, on the input pair $(\alpha, \beta)$ and this should hold for any $(\alpha, \beta)$.

That means if you know $\beta$ and if you know the value of the function $E$ on an unknown input $\alpha$, then even without knowing $\alpha$, it should be easy for you to compute the value of the function $F$ for the input pair $(\alpha, \beta)$. And finally since we are aiming for weak-privacy, we require the following property; we require that if someone knows the value of the function $F$ on an unknown $\alpha$ and an unknown $\beta$, then it should be very difficult to find out the value of the function $F$ on the input pair $(\alpha, \beta)$.

So these are the properties which I require from the function $E$ and $F$. And it is easy to see that all together, all these properties automatically imply that my function $E$ should be a one-way function. So remember we had already discussed what exactly is a one-way function long time back. A one-way function informally is a function which is easy to compute on any value from its domain, but computationally very difficult to invert, for random values from the range of the function.

So it is easy to see that all these implications imply that the function $E$ should be a one-way function. Because if the function $E$ is not a one-way function, then given $E(\alpha)$, it will be easy to compute $\alpha$. And given $E(\beta)$, it will be easy to compute $\beta$. And automatically if you know $(\alpha, \beta)$ and a description of the function $F$, you will be able to compute the value of $F$ on $(\alpha, \beta)$, which goes against the requirement that it should be difficult to compute $F(\alpha, \beta)$, if you are just given the value of $E(\alpha)$ and $E(\beta)$ only.

But not only we require the function $E$ to be one way, it turns out that we need some additional properties as well. Namely the property that we require here is that if you know $\alpha$ and the value of

$E$ on the input $\beta$, then even without knowing $\beta$, it should be easy for you to compute $F(\alpha, \beta)$. And in the same way, without knowing $\alpha$, given just $E(\alpha)$ and $\beta$, it should be easy for you to compute $F(\alpha, \beta)$.

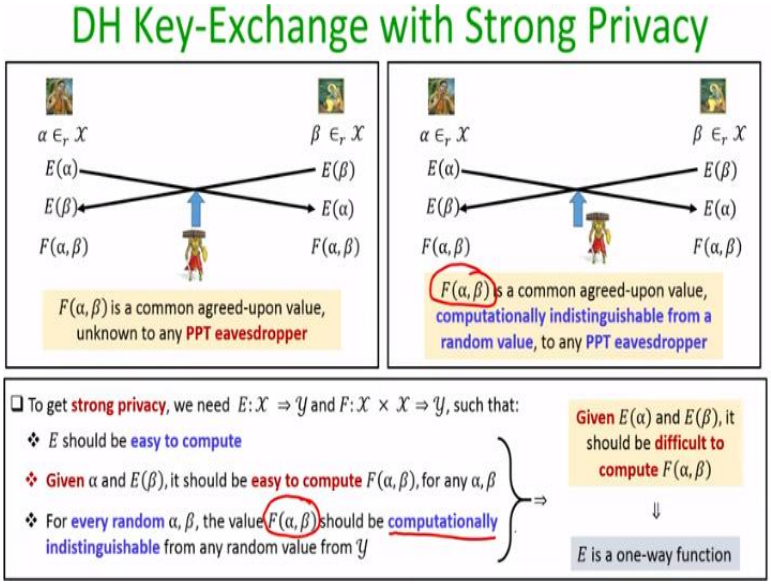
So these are now additional requirements on this one-way function. Later on we will see exact instantiations of these functions $E$ and $F$. But for the moment, assume we have such functions $E$ and $F$. Now let us see how exactly we can convert this color based protocol into a concrete key-exchange protocol. So assume the public setup, namely the public color here which is known both to the sender and the receiver is nothing, but a description of the function $E$ and a description of the function $F$.

Now what the sender and receiver has to do they have to prepare their respective secret mixtures. So what they do is, sender picks a random $\alpha$ from this $\mathcal{X}$ set and computes $E(\alpha)$. And independently, receiver picks a $\beta$ from the set $\mathcal{X}$ and compute $E(\beta)$. And they exchange the values of $E(\alpha)$ and $E(\beta)$ over a public channel. And now I am assuming that even if there is an adversary who sees the value of $E(\alpha)$ and $E(\beta)$, since the function $E$ is a one-way function, it will be difficult for an adversary to find out what exactly $\alpha$ and $\beta$ are; that is what is the property I am assuming from the function $E$.

Now once $E(\alpha)$ and $E(\beta)$ are known to the sender and receiver respectively, again by using the fourth property from the function $E$ and $F$, we can see that it is easy for both sender and receiver to compute $F(\alpha, \beta)$. Why? Because sender is now knowing $E(\beta)$ and it has $\alpha$. So using the knowledge of $E(\beta)$ and $\alpha$, it can obtain $F(\alpha, \beta)$. And in the same way, the receiver it is now learning $E(\alpha)$, it is fine if he does not know $\alpha$, but given $\beta$ and $E(\beta)$, it can compute $F(\alpha, \beta)$.

And now you can see that $F(\alpha, \beta)$ is a common value from the set $\mathcal{Y}$, which is known both to the sender and the receiver. But it will not be known in its entirety to any eavesdropper and that comes from the fact that the function $E$ is a one-way function.

**(Refer Slide Time: 31:53)**




Now if you want to obtain a key-exchange protocol with a stronger privacy notion namely the strong-privacy, where the resultant key should be indistinguishable from any potential key from the key space. Then we can retain the same blueprint of the Diffie-Hellman key-exchange protocol that we have seen just now. Namely sender and receiver picking respective $(\alpha, \beta)$ and exchanging the values of $E(\alpha)$ and $E(\beta)$.

What we have to just do now is to ensure that the resultant key or the resultant value namely $F(\alpha, \beta)$ is a uniform, computationally indistinguishable value from the viewpoint of an eavesdropper. We need now some more stronger properties from the function $E$ and from the function $F$. Namely the requirement of one-wayness is still there; the function $E$ should be easy to compute but difficult to invert on random values from the domain. And again given $\alpha$ and the value of $E(\beta)$, it should be easy to compute $F(\alpha, \beta)$, for any $(\alpha, \beta)$.

And now the stronger property that I require from the function $E$ and $F$ is that for every random $(\alpha, \beta)$, the value of $F(\alpha, \beta)$ should be computationally indistinguishable from any random value from the set $\mathcal{Y}$. So this new requirement was not there when we were aiming to obtain the Diffie-Hellman key exchange protocol with a weaker notion of privacy. Because there the requirement was that adversary should not learn $F(\alpha, \beta)$ even if it knows $E(\alpha)$, $E(\beta)$. But now since we are aiming for a stronger privacy notion, we are putting additional requirements on the function $E$ and $F$. Namely, we require that even if someone knows $E(\alpha)$, $E(\beta)$, the value of $F(\alpha, \beta)$ for that

adversary or for that person should be computationally indistinguishable from any random value from the set $\mathcal{Y}$.

And if we assume that indeed our function $E$ and $F$ satisfies this additional requirement, it is easy to see that the same protocol which we had just seen giving us the weaker privacy notion, ends up giving us a stronger privacy notion. We do not have to add additional step; we just have to make stronger assumptions about the functions $E$ and $F$. So that brings me to the end of this lecture.

Just to quickly summarize, in this lecture, we have introduced the key-exchange problem where the goal of the sender and the receiver is to interact publicly over an in-secure channel with no pre-shared information and end up with some common output, which is known only to the sender and the receiver but unknown to any third party. And we have introduced two notions of secrecy for such key-exchange protocols, namely the notion of weak-privacy and the notion of strong-privacy. Thank you.