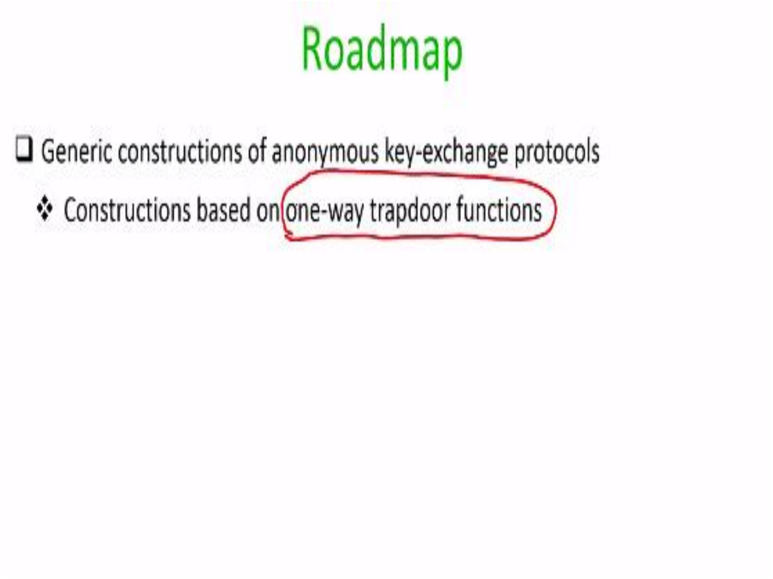
37Foundations of Cryptography
Dr. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology – Bangalore

Lecture -37
Key Exchange Protocols Part II

(Refer Slide Time: 00:28)



Roadmap

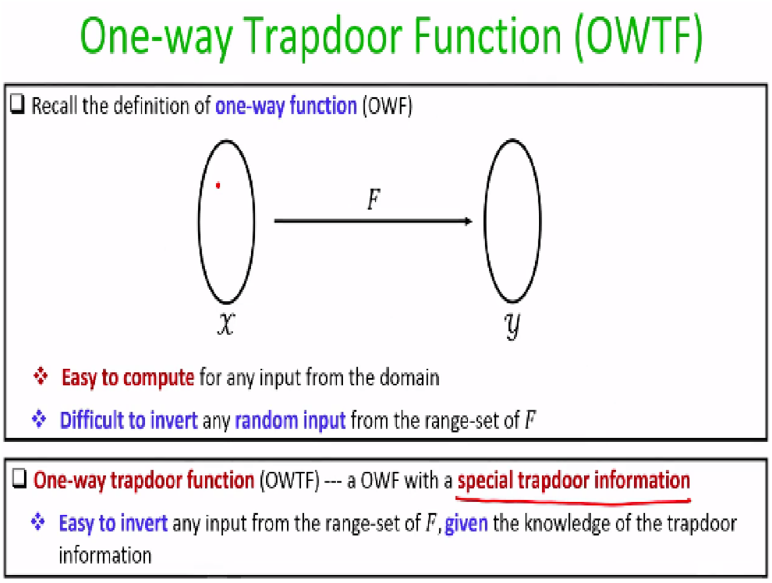
❑ Generic constructions of anonymous key-exchange protocols

❖ Constructions based on one-way trapdoor functions

Hello everyone, welcome to this lecture. Well just to recall in the last lecture we had introduced the key exchange problem and we have seen various security notions and we have also seen the idea involved behind the Diffie–Hellman Key Exchange Protocol although, we have not yet seen the exact mathematical details, but we have seen how given some special functions E and F how we can design the Diffie–Hellman Key Exchange Protocol.

Now what we are going to do here is we are going to continue our discussion on the key exchange protocols and we will see generic constructions of key exchange protocols namely we will see constructions based on one-way trapdoor functions which are very nice mathematical primitive or fundamental cryptographic primitive which are more powerful than the notion of one-way functions.

(Refer Slide Time: 01:18)



One-way Trapdoor Function (OWTF)

❑ Recall the definition of one-way function (OWF)

❖ Easy to compute for any input from the domain

❖ Difficult to invert any random input from the range-set of F

❑ One-way trapdoor function (OWTF) --- a OWF with a special trapdoor information

❖ Easy to invert any input from the range-set of F, given the knowledge of the trapdoor information

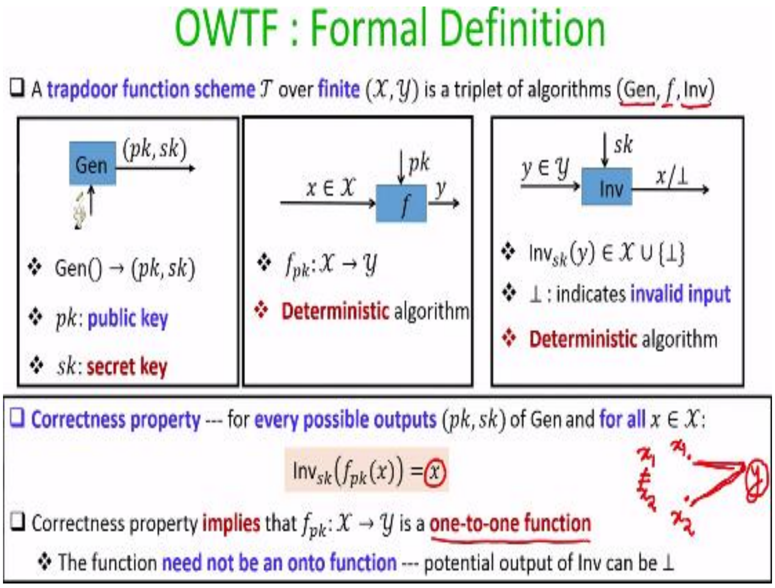
So what exactly are one-way trapdoor functions? So before going into the definition of one-way trapdoor functions let us first recall the definition of one-way function or OWF. So a one-way function F is a publically known function, a deterministic functions from the domain x to the co-domain y and the requirements from this one-way function are that it should be easy to compute for any value x from the domain x.

Whereas it should be difficult to invert in polynomial time any random input from the range set of f that means if someone gives me a y which belongs to the range of f and y is randomly chosen than in polytime it should be difficult to come up with some at least one image for this input y. Now what exactly is one-way trapdoor function right. So it sounds like a special type of function.

And indeed it is a special type of one-way function which also has an associated trapdoor information and what exactly this trapdoor information is going to help you with well it helps you to invert any input from the range set of the function f provided you have the knowledge of the trapdoor information that means any entity who knows the description of this trapdoor information then it can easily invert the function f on any random y from the range set of f.

But in the absence of the trapdoor information the function looks like a one-way function to that entity it would not be possible for that entity who does not process this trapdoor information to invert this function f on any random y from the range set of the function f.

(Refer Slide Time: 03:00)



OWTF : Formal Definition

❑ A trapdoor function scheme $\mathcal{T}$ over finite $(\mathcal{X}, \mathcal{Y})$ is a triplet of algorithms (Gen, f, Inv)

❖ Gen() → (pk, sk)

❖ pk: public key

❖ sk: secret key

❖ $f_{pk}: \mathcal{X} \rightarrow \mathcal{Y}$

❖ Deterministic algorithm

❖ $\text{Inv}_{sk}(y) \in \mathcal{X} \cup \{\perp\}$

❖ $\perp$: indicates invalid input

❖ Deterministic algorithm

❑ Correctness property --- for every possible outputs (pk, sk) of Gen and for all $x \in \mathcal{X}$:

$$\text{Inv}_{sk}(f_{pk}(x)) = x$$

❑ Correctness property implies that $f_{pk}: \mathcal{X} \rightarrow \mathcal{Y}$ is a one-to-one function

❖ The function need not be an onto function --- potential output of Inv can be $\perp$

So now let us try to formally state this requirement what exactly we mean by this special trapdoor information and is easy to invert in the presence of the trapdoor information and difficult to invert in the absence of the trapdoor information. So when we say that we have a trapdoor function scheme $\mathcal{T}$ then it is over a pair of finite sets where $\mathcal{X}$ will be domain of the function and $\mathcal{Y}$ will be co-domain of the function.

And when we say a trapdoor function scheme basically we mean a triplet of algorithms all polytime algorithms, polynomial in some security parameter the first algorithm is the key generation algorithm or basically parameter generation algorithm (Gen) and the function f is the function in this underlying context will be a one-way function and corresponding to the function f we have an inversion function (Inv).

So syntactically the parameter generation algorithm or the key generation algorithm is a randomized algorithm which does not take any explicit input, but internally it has some internal randomness it generates internal random coins and based on the value of random coin it output a pair of keys or pair of values which I denote by pk and sk. So pk is going to be the public key which will be available in the public domain.

And sk is the secret key which will be known or which will be available only with some designated entities. Now what is a function f? Well it is a function from the set $\mathcal{X}$ to the set $\mathcal{Y}$. So it is a function parameterized by a public key pk so anyone can evaluate this function provided it knows of the description of the function and the description of the public key pk and the function takes two explicit inputs.

Namely the input x on which the function needs to be evaluated and the public key and it gives you an output which I denote by y and this y will be an element from the co domain set $\mathcal{Y}$ and it is a deterministic algorithm that means every time you evaluate this function f with the same input x and with the same public key pk you will obtain the same output y.

There is no randomness which is there as part of the algorithm. Now the inverse algorithm it is also a 2 input function it takes an input y from the set $\mathcal{Y}$ which we want to invert and the secret key sk that means this function can be invoked only by an entity who possess the secret key which actually in our context is a trapdoor information. If you do not have the secret key or the trapdoor information, then you would not be able to invoke this function.

So assuming that entity, an entity who knows the input y on which this function Inv has to be invoked also knows the trapdoor information, it runs the function with this input sk and y and it is going to obtain either an output x or a special status symbol bot which denotes an invalid input right. So if the output of this inverse function is not bot then it is going to be an element from the domain of the function f.

Namely it is going to be element of $\mathcal{X}$ whereas if the output is bot then it indicates an invalid input and we will soon see what exactly we mean by an invalid input with respect to the underlying context and this inverse function is always going to be deterministic algorithm that means if you invoke this inverse function with the same y and the same secret key sk then you are going to obtain the same output again and again.

Now what are the properties we require from this one-way trapdoor function? The first property is the correctness property which states that for every pair of parameters or keys pk, sk generated by your key generation algorithm and for every input x from your domain of the function f if you invoke the function on the input x with respect to the key pk and the resultant output is say is denoted as y.

And then if you try to invoke the function Inv on the input y and with the secret key sk or the trapdoor information sk then the output should be the same input x and this should hold for every pk, sk generated by your key generation or parameter generation algorithm. Now notice that as soon as we state this correctness requirement it automatically means that your function f which is there as part of your trapdoor function scheme has to be one-to-one function.

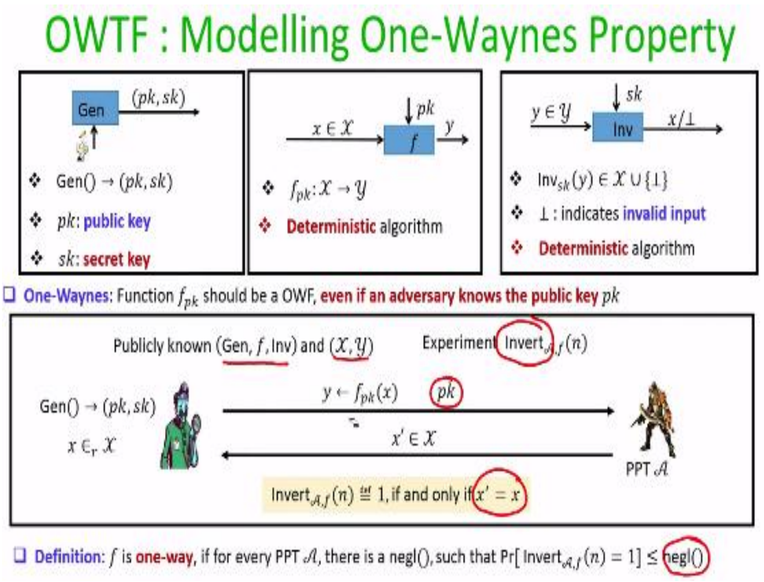
That means if you have 2 distinct inputs x_1 and another input x_2 where x_1 is not equal to x_2 right and if you try to evaluate the function f_{pk} on the input x_1 and on the input x_2 you cannot get the same output y because if both x_1 and x_2 gets map to the same output y under the function f_{pk} then what happens when you try to invert the input y with the secret key sk. You can get x_1 with some probability you can get back x_2 with some probability.

But that strictly goes against the correctness requirement, the correctness requirement says that you should obtain back the exact input with which you evaluated the function f_{pk} and then try to run the inverse algorithm unambiguously and that is guaranteed only if your function f_{pk} is a one-to-one function. However, noticed that even though the function f_{pk} is a one-to-one function that does not mean that the function has to be an onto mapping.

There might be several possible inputs from your co-domain set $\mathcal{Y}$ which may not have a pre image under the function f and that is why we have a special provision for the output of the inverse function to be a bot symbol because if we are trying to invoke the function Inv on some element y which does not have a pre image that means this y is not an element of the range set of your function f.

Then we should get the output bot indicating that we are trying to invert something which does not have corresponding pre image right. So that is a definition of one-way trapdoor function.

(Refer Slide Time: 09:42)



OWTF : Modelling One-Waynes Property

- ❖ $\mathsf{Gen}() \rightarrow (pk, sk)$
- ❖ pk: public key
- ❖ sk: secret key

- ❖ $f_{pk}: \mathcal{X} \rightarrow \mathcal{Y}$
- ❖ Deterministic algorithm

- ❖ $\mathsf{Inv}_{sk}(y) \in \mathcal{X} \cup \{\perp\}$
- ❖ $\perp$: indicates invalid input
- ❖ Deterministic algorithm

❑ One-Waynes: Function f_{pk} should be a OWF, even if an adversary knows the public key pk

Publicly known $(\mathsf{Gen}, f, \mathsf{Inv})$ and $(\mathcal{X}, \mathcal{Y})$ Experiment $\mathsf{Invert}_{\mathcal{A},f}(n)$

$\mathsf{Gen}() \rightarrow (pk, sk)$

$x \in_r \mathcal{X}$

$y \leftarrow f_{pk}(x)$ pk

$x' \in \mathcal{X}$

PPT $\mathcal{A}$

$\mathsf{Invert}_{\mathcal{A},f}(n) \stackrel{\text{def}}{=} 1$, if and only if $x' = x$

❑ Definition: f is one-way, if for every PPT $\mathcal{A}$, there is a negl(), such that $\Pr[\,\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leq \mathsf{negl}()$

So we have seen the correctness requirement now we have to model the One-Wayness property and remember the One-Wayness property requires that any entity which posses the secret key or the trapdoor information should be able to invert the function or compute the output of the function Inv on any y from the range set of the function f, but in the absence of the trapdoor information it should be computationally difficult to compute the output of the inverse function on any y belonging to the range set of the function f right.

And this should hold even if that entity or that adverserial algorithm knows the description of the public key or public key pk. So let us try to model this requirement by an experiment which we call as an $\mathrm{Invert}_{\mathcal{A},f}(\mathrm{n})$ experiment. This experiment is almost the same as the way experiment which we used to model One-Wayness property for one-way function which does not have a trapdoor information associated with it.

But it will now have some different security requirement here or the output of the experiment is defined in a different way and rules of the experiment are also slightly different here. So what is known publicly is the description of a trapdoor scheme namely the triplet of algorithm (Gen, f, Inv) and the domain and co-domain of the function f. Now the challenge is generated for the adversary is as follows.

The challenger of the experimenter runs the parameter generation algorithm to obtain the pair of public key and the secret key and it picks a random x from the domain of the function f and the challenge which is thrown to the adversary are as follows. The public key is given to the adversary and the value of the function f on the random input x namely y is also given as a challenge to the adversary.

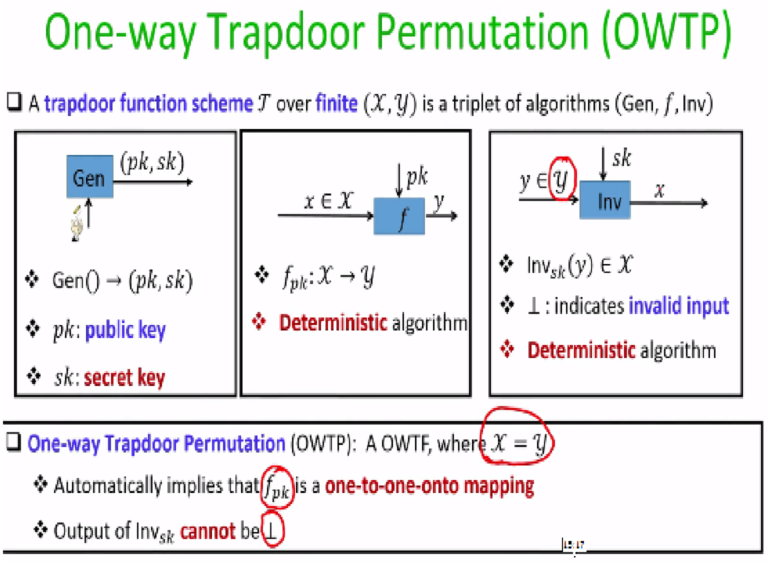
And the goal of the adversary is basically to compute the pre image of this y without knowing the corresponding secret key or the trapdoor information sk. So basically adversary after polynomial amount of time is going to output some x’ from the domain of the function f and we say that the experiment output is 1, meaning that adversary has won the experiment if and only if adversary A is able to correctly come up with the input x which has given the output y.

And my security definition is we say that a function f which is a part of your trapdoor function scheme is one-way if for any polytime adversary who participates in this experiment there is some negligible function in the security parameter such that the probability of that adversary winning this experiment is upper bounded by some negligible function where the probability is taken over the random choice of the experiment.

Namely the randomness involved in generating the parameters pk, sk and the randomness used for selecting the value of x here right. So basically the essence of this experiment is the challenge for the adversary is to come up with a right x even without knowing sk and security definition demands that if the adversary does not know the trapdoor information or the secret key sk then except with some very small probability it should be computationally difficult for a polytime adversary to come up with a correct x.

In the definition we do not bound a success probability of the adversary winning the experiment to be 0 because there is always a guessing strategy by the adversary who can just guess some random x’ from the set x and with some non zero probability it may so happen that a guessed x’ indeed turns out to be the correct x. So the best that we can hope for is that any polytime adversary should not be able to do anything better than guessing the random x on which the function f is evaluated to compute the challenge y.

(Refer Slide Time: 13:34)



One-way Trapdoor Permutation (OWTP)

❑ A trapdoor function scheme $\mathcal{T}$ over finite $(\mathcal{X}, \mathcal{Y})$ is a triplet of algorithms (Gen, f, Inv)

- ❖ Gen() → (pk, sk)
- ❖ pk: public key
- ❖ sk: secret key

- ❖ $f_{pk}: \mathcal{X} \to \mathcal{Y}$
- ❖ Deterministic algorithm

- ❖ $\text{Inv}_{sk}(y) \in \mathcal{X}$
- ❖ $\perp$: indicates invalid input
- ❖ Deterministic algorithm

❑ One-way Trapdoor Permutation (OWTP): A OWTF, where $\mathcal{X} = \mathcal{Y}$

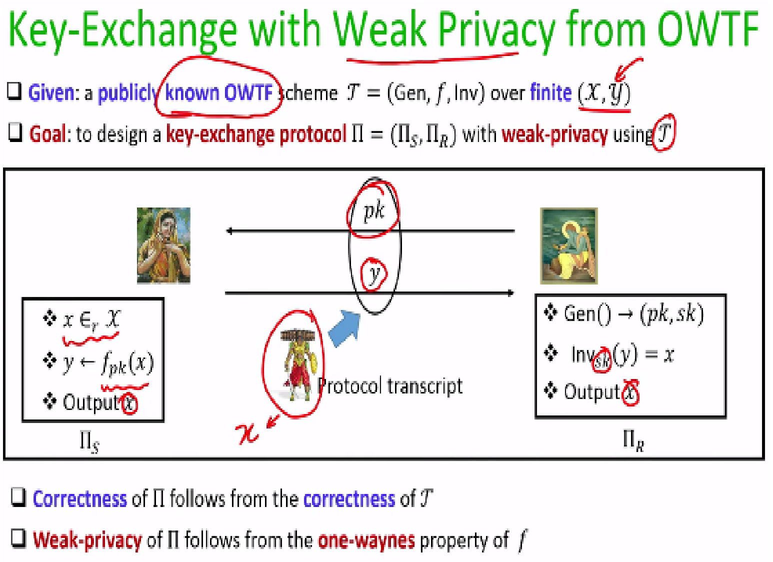
- ❖ Automatically implies that f_{pk} is a one-to-one-onto mapping
- ❖ Output of Inv_{sk} cannot be $\perp$

Now once we have the definition of one-way trapdoor function let us define a related notion namely one-way trapdoor permutation OWTP right. So one-way trapdoor permutation is a special type of one-way trapdoor function where the domain and the co-domain of your function f are the same namely the set $\mathcal{X}$ and the set $\mathcal{Y}$ are same and this automatically implies that your function f_{pk} is one-to-one onto mapping that means it is a bijection.

This is because we know that if the function is a one-way trapdoor function then the correctness property implies that your function f has to be a one-to-one function but it need not be onto function, but as soon as I ensure that my domain and the co-domain are same and if both co-domain and domain are finite sets which is indeed going to be case for the construction which we will see later. Then if my $\mathcal{X}$ and $\mathcal{Y}$ are same and they are finite and if my function is a one-to-one mapping then it automatically implies that the function is also onto mapping, that means for every x I am going to have a unique image and I cannot have 2 different x mapping to the same y and for every y from the set $\mathcal{Y}$ I am going to have a distinct x such that f(x) would have given me that y.

That automatically means that if I am considering one-way trapdoor permutation then the output of my inverse function can never be a bot because indeed I am going to invoke my inverse algorithm on some input y belonging to the set $\mathcal{Y}$ then since my function is a invertible function or it is one-to-one onto mapping I am always going to obtain an element x belonging to the domain of the function f namely $\mathcal{X}$. So bot cannot be a possible outcome here. So that is the only difference for the case of one-way trapdoor permutation.

(Refer Slide Time: 15:30)



Key-Exchange with Weak Privacy from OWTF

- ❑ Given: a publicly known OWTF scheme $\mathcal{T}$ = (Gen, f, Inv) over finite $(\mathcal{X}, \mathcal{Y})$
- ❑ Goal: to design a key-exchange protocol $\Pi = (\Pi_S, \Pi_R)$ with weak-privacy using $\mathcal{T}$

- ❖ $x \in_r \mathcal{X}$
- ❖ $y \leftarrow f_{pk}(x)$
- ❖ Output x

Π_S

- ❖ Gen() → (pk, sk)
- ❖ $\text{Inv}_{sk}(y) = x$
- ❖ Output x

Π_R

- ❑ Correctness of Π follows from the correctness of $\mathcal{T}$
- ❑ Weak-privacy of Π follows from the one-waynes property of f

So now assume for the moment that you have given a candidate one-way trapdoor function we have not yet seen how whether indeed we have not yet seen a candidate one-way trapdoor function later on when we will introduce number theoretic assumption and number theoretic hard problems we will see examples of one-way trapdoor function, but for the moment assume that you are given a one-way trapdoor function we will see how using that one-way trapdoor function we can design an anonymous key exchange key protocol achieving the notion of weak privacy.

So what is publically known is a description of the one-way trapdoor function scheme over some finite sets x, y and the goal here is to design a key exchange protocol namely a pair of protocols one for the sender and one for the receiver achieving the notion of weak privacy where using this one-way trapdoor function scheme and a resultant scheme is going to be an element from this set y .

And from the description of the one-way trapdoor function what exactly one-way trapdoor function does : it allows anyone to compute the evaluate the function f, but until and unless other party owns the trapdoor information it would not be possible to invert the output of the function f. So now based on this intuition it is very simple to come up with a key exchange protocol.

So the protocol Π_S and Π_R for the sender and the receiver respectively will be as follows. What the receiver can do here is it can run the parameter generation algorithm of the one-way trapdoor functions scheme and obtain the pair of public key and secret key and it can give the description of the public key to the sender here right and once the sender learns the description of the public key what it does is : it takes the random x from the domain of the function f and it knows the description of the function f and it now knows the public key pk. What it can do is it can just evaluate the function f using the key pk and the input x and obtain an output y which it communicates to the receiver. Now what the receiver can do is since the receiver posseses the secret key or the trapdoor information sk, it can compute the output of the inverse function on the input y under the key sk and it can obtain the same x with which has been used by the sender to compute a output y and both sender and receiver can now output a common value or a common key namely x. Now the correctness of this key exchange protocol automatically follows from the correctness property of your one-way trapdoor function scheme right.

Because since we are in the passive world right a copy of the pk which has been communicated by the receiver to the sender will be an authenticated copy it will be the same pk which receiver has generated and corresponding sk is used by the receiver to compute the inverse of y under the secret key sk : $Inv_{sk}(y)$. So the correctness of this whole key exchange protocol follows from the correctness property of your underlying trapdoor function scheme.

And a weak privacy of this whole key exchange protocol follows from the One-Wayness property of the underlying function f. So if we have a polynomial time eavesdropper who eavesdrops the communication and take the protocol transcript so what exactly is the protocol transcript for the adversary. From the viewpoint of the adversary it is a random pk whose corresponding trapdoor information sk is not known to the adversary.

And adversary is now seeing the value y which is the output of the function f_{pk} on some random x where the random x is not known to the adversary and the requirement of weak privacy is that the resultant x which is the output for the sender and receiver here should not be known in its entirety to the adversary. We are not aiming here for the indistinguishability based secrecy definition.

We are actually aiming for all or nothing kind of security requirement here because we are in the weak privacy setting here. So if at all the polynomial time adversary by learning the public key pk and a function output y can come up with x in polytime with significant probability then it means that we have an instance of the weak privacy or the Invert experiment where a polytime adversary can win that experiment with non negligible probability which goes against the assumption that the trapdoor function scheme is a one-way trapdoor function scheme.

So only with negligible probability this polytime eavesdropper could come up with the correct x, otherwise the x will not be known to the adversary and that ensures that this key exchange protocol is indeed gives you the notion of weak privacy. So that brings me to the end of this lecture. Just to summarize in this lecture we have introduced a notion of one-way trapdoor function and we have also introduced a notion of one-way trapdoor permutation.

So one-way function is a special type of one-way function which is easy to invert on any random or any value from the co-domain or the range set of the function provided you have a special trapdoor information, but in the absence of trapdoor information it is very difficult to invert any random input from the range set of the function and we have seen that if you are given a candidate one-way trapdoor function then we can design a key exchange protocol achieving the notion of weak privacy. Thank you.