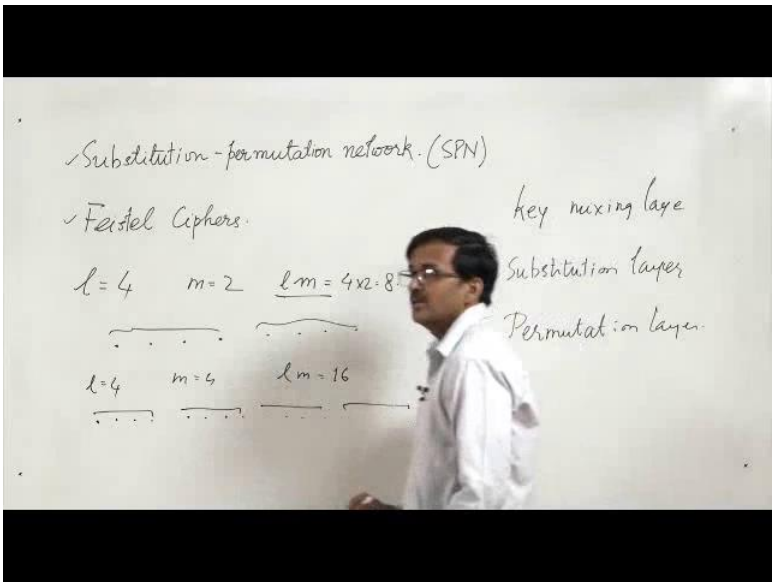
Introduction to Cryptology
Dr. Sugata Gangopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Roorkee

Lecture – 07
Substitution permutation network, Feistel cipher

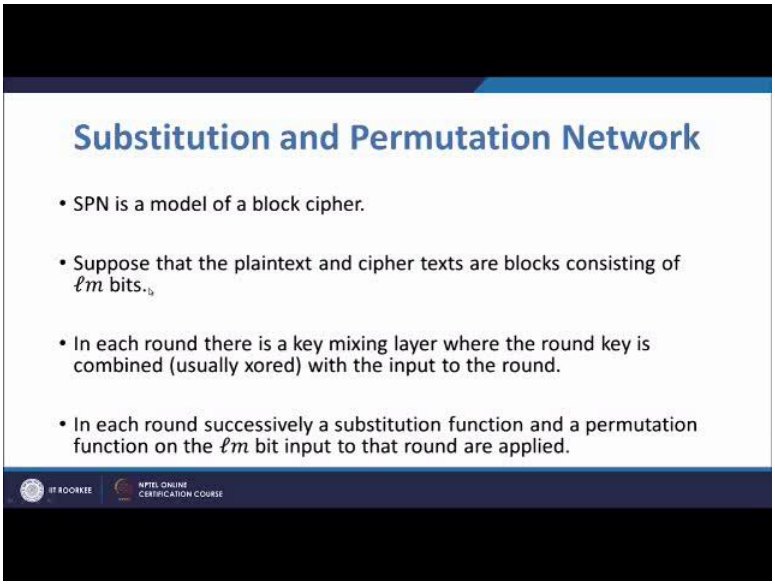
Hello. Welcome to week 2, Lecture - 2. In this lecture, we will study 2 basic designs of block ciphers namely substitution-permutation network, in short SPN and Feistel ciphers.

(Refer Slide Time: 00:40)



First, substitution and permutation network; in short - SPN.

(Refer Slide Time: 01:32)



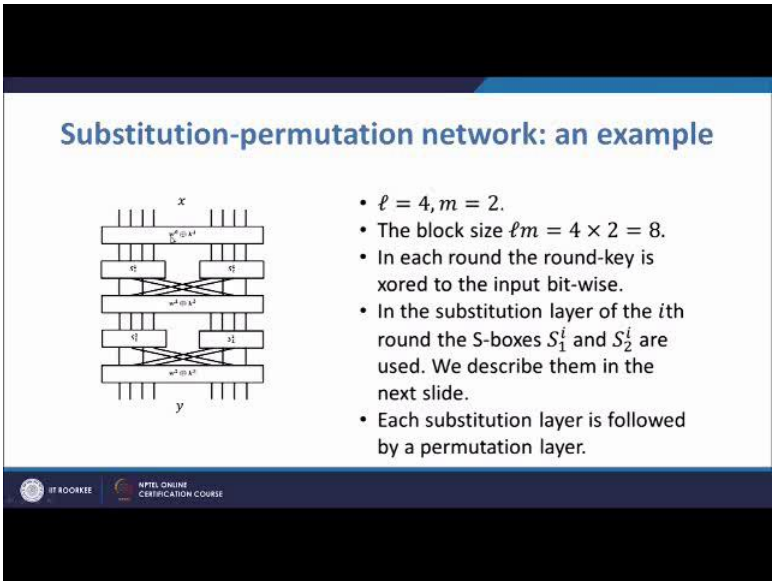
SPN is a model of a block cipher. We suppose that the plain text and cipher texts are blocks containing l into m bits. So, let us suppose we take a particular value l equal to 4 and m equal to 2, then the plaintext will be l m; that is 4 into 2 equal to 8-bit long. So, I have got 8 bits like this, 8 bits like this, 4 bits here, and 4 bits after that. I could have had other values, I could have had l equal to 4, m equal to 4; and then l m will be 16. So, my plain text and cipher text will be 16 bit blocks and intrinsically they will be divided into 4-bit sub blocks like this. It will be very soon clear to us why these are important, but this is to get the symbols clear.

Now, in a way the round functions are also standardised. The keys are also, ultimately; the keys ultimately will be having the same block size as the plain text or cipher text and in each round function there is first a key mixing layer. So, we will have a key mixing layer, and then, we will have a substitution layer. Now, you will, of course, ask me a question - that what do I mean by substitution layer? It will be very soon clear to you, and then we will be having a permutation layer, and of course, you will ask me a question right now - what do I mean by permutation layer and what is a difference between substitution and permutation in this context? That also will be clear to you right now.

So, let's see whether we have got all that terminologies discussed. So, we have round functions, and we have said that they are some layers in there, and we have talked about the blocks, and how the blocks will be divided into further sub blocks. Now, we come to a particular example of a substitution-permutation network and this is a very small

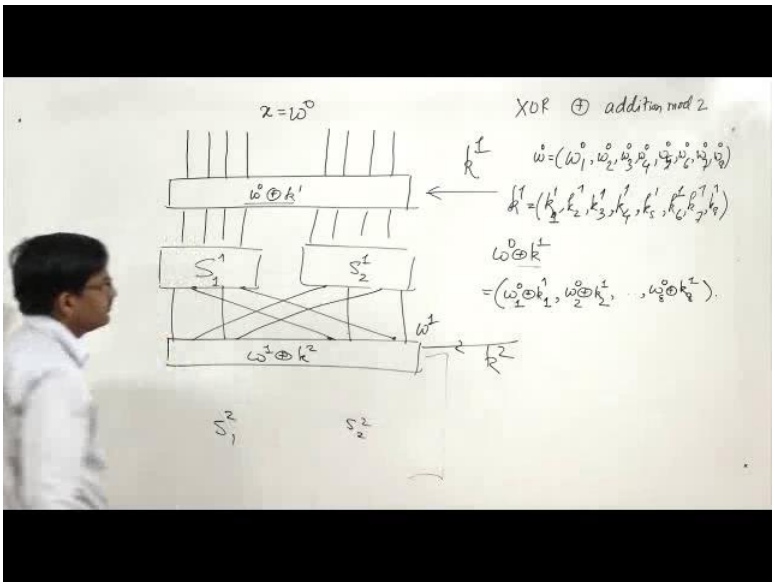
network. It is, of course, not secure or may not be secured, but this is for illustration. So, here we have taken blocks to be pretty small.

(Refer Slide Time: 05:05)



We have got 8-bit blocks and that divided into 2 sub blocks one bit; the first one 4-bit; the second one 4-bit. So, if you look at the diagram here, you will see what I mean. I have got 8 bit blocks divided into sub blocks, and then, we have the key mixing layer for the first round.

(Refer Slide Time: 05:36)



So, in our example, 8 input bits are coming in. This is my plain text x and I write as x equal to w zero, and here I have got the key mixing because I am accepting the first-round key, and I am bitwise exploring the first-round keys. So, if I want to be more explicit, I will write w 0 as w 1 0, w 2 0, w 3 0, w 4 0, w 5 0, w 6 0, w 7 0, and w 8 0, and the key k 1 as k 1 0, k 2 0, k 3 0, sorry it is not 0, but 1, 1, 1, k 4 1, k 5 1, k 6 1, k 7 1, and k 8 1, and by this operation I mean that I am taking each individual bits which are components of the plain text and the key, and adding them modular 2 which I called XOR, and denote by this. So, if this symbol will mean this vector, this is not k 0 this is k one. So, k 1 1, k 2 1, and so on up to w 8 0 XOR k at 1; this will mean this.

After that, we have the substitution layer. Here, we are splitting this 8-bit block into the sub blocks - 4-bit sub blocks - and applying a transformation which is called a substitution box or an S-box in short. This simply takes the 4-bit input and transforms it to some other 4-bit, some other 4-bit which as an output; now, how to design an S-box or how to design S-boxes with good cryptographic properties is a very difficult question of current interest. Anyway, right now we are not going into those questions, but we are just saying that there is something here which takes as 4-bit input, which takes 4-bit 4 bits as input and sends out 4-bits as output.

So, this is a substitution layer. So, I will draw it over here. Now, I will call this s 1 1 and s 1 2. This is S-box of the first round, S-box of the first round; the first S-box of the first round, and the second S-box of the second round. After that we have the permutation layer. So, this is a substitution layer, and then, we have the permutation layer where a bit coming out of this S-box is going to this side, and bits - some of the bits - coming out of S-box is going to side, and we have to specify the functions which is doing that. This is what we mean by permutation here, we are permuting the order in which the bits are coming out of these 2 S-boxes.

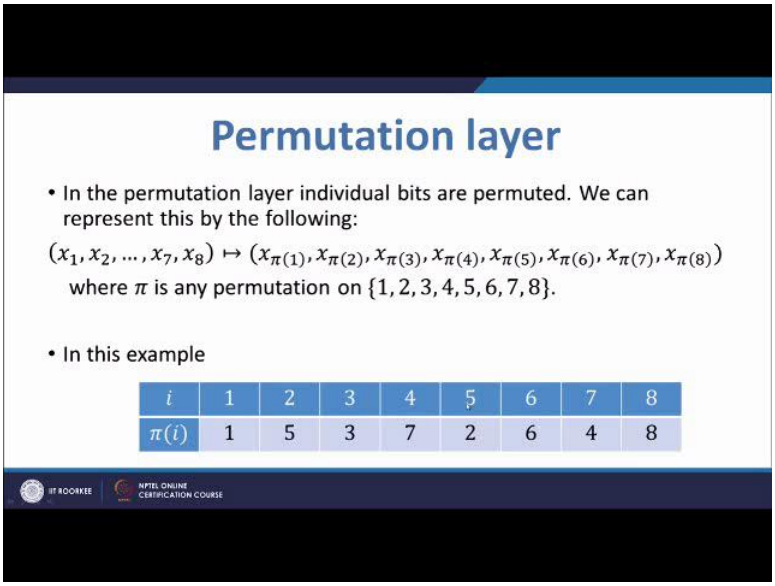
Once I do that, I come to the beginning of the next round. So, here more or less what I did is that one is going to one, and the second one is going to one over here, third one is going here, and forth one is going to the one, but this one over here. So, last one is going here, this one is coming here, and this one is coming here, and this one is coming over here. So, if you look at the slide you will understand and we have got; I have given the exact function in the latest slides. This is what happens.

And now we come to a kind of self-similar part of the cipher. Here I will say that this is my w one which I obtained from w 0, and I now accept the second-round key which is k two, and I will take bitwise XOR of w one with k 2 and then apply the same things over here. And here I have written the S-boxes to be s 2 1, s 2 2, but in reality, I mean in this particular example, I have assumed that all the S-boxes are the same and we will give an explicit description of the S-box that we are using.

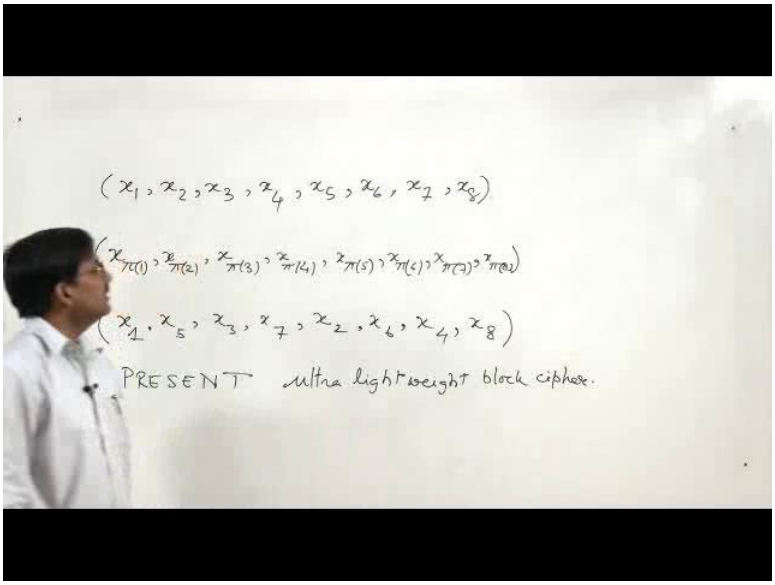
Now, the first let us come to the easier layer, which is called the permutation layer. The key mixing layer we have already seen. It is essentially easy because we take the key, we take the plain text, and we just add them bitwise modulo 2. Of course, when I go into the rounds it need not be the plain text; it might be the output from the previous round added to the corresponding round key. So, that is the key mixing layer.

And in the permutation layer it is a permutation.

(Refer Slide Time: 13:32)



Now, let us see how we specify these functions. (Refer Slide Time: 13:47)



So, the input to a round or let us say the input to a permutation layer, in this case, will be 8 bits x 1, x 2, x 3, x 4, x 5, x 6, x 7, and x 8. This need not be the plain text; it may be just some intermediate 8-bit block which is going into, which is coming after the substitution layer, going into the permutation layer. And I define a permutation in this way. So, I have got the numbers from 1, 2, 3, 4, 5, 6, 7, 8, and I am saying that pi is my permutation, and this is a value of pi i. So, this is a value of pi 1, this is a value of pi 2, and so on. So, pi 1 is 1, pi 2 is 5, pi 3 is 3, pi 4 is 7 and so on.

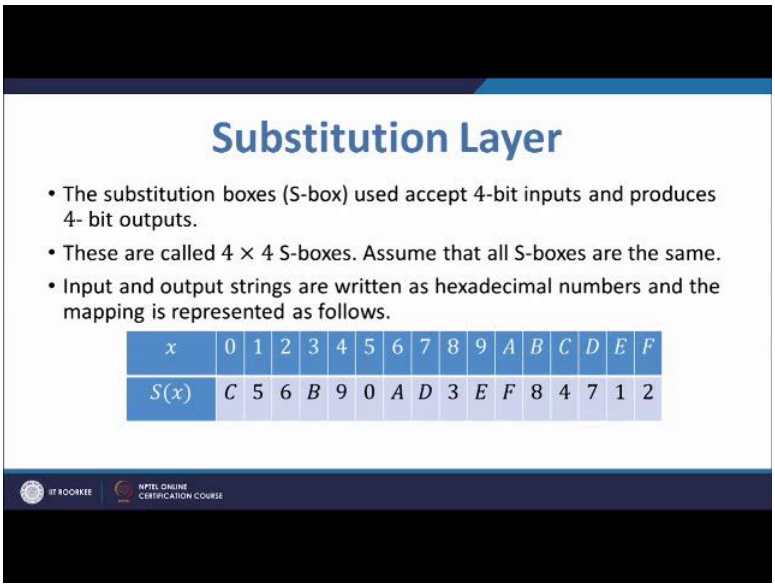
So, formally we will write like this, that this is my input to the permutation layer, and the output is x pi 1, x pi 2, x pi 3, x pi 4, x pi 5, x pi 6, x pi 7, x pi 8. And now, I put the values from the table this is pi 1, is one pi 2 is 5, pi 3 is 3, pi 4 is 7, pi 5 is well pi 5 is 2, pi 6 is 6, pi 7 is 4, and pi 8 is 8. So, I have got the permutation.

Here, we see that the transformation on the block is not very complicated. What it is doing is basically it is taking the bits from one portion to the other portion. So, that is what it is doing. So, that is the permutation layer.

Now we will come to possibly the most important layer which is the substitution layer, and this is this one, and this is where we encounter S-boxes. So, here we have got a description of a 4 by 4 S-box, which is a 4-bit input, 4-bit output, and in fact, this is an important real life S-box, because this is the S-box of the block cipher called PRESENT, which is a block cipher which has been proposed recently as an ultra-light weight block cipher. And it is quite possible that we will be having many real-life situations where

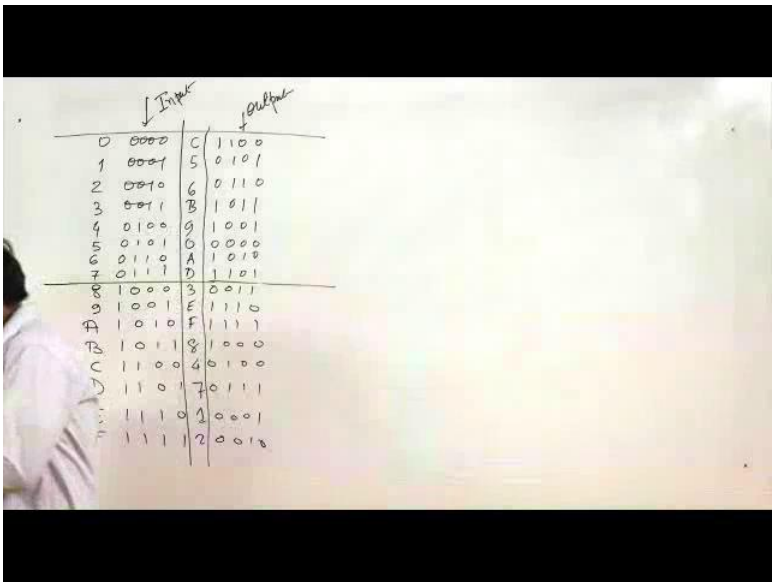
PRESENT will be used. And the PRESENT use one S-box and that is S-box is this 1, but we have to learn how to read this S-box. This, a 4 by 4 S-box well, but and it is a good one. So, now, let us see how to read this.

(Refer Slide Time: 17:48)



So, here we will see numbers – 0, 1, to up to f. These are hexadecimal numbers. In order to read them in our context we have to transform them into binary sequences. So, when I say 0, it means all 0, 0 0 0 0.

(Refer Slide Time: 18:10)



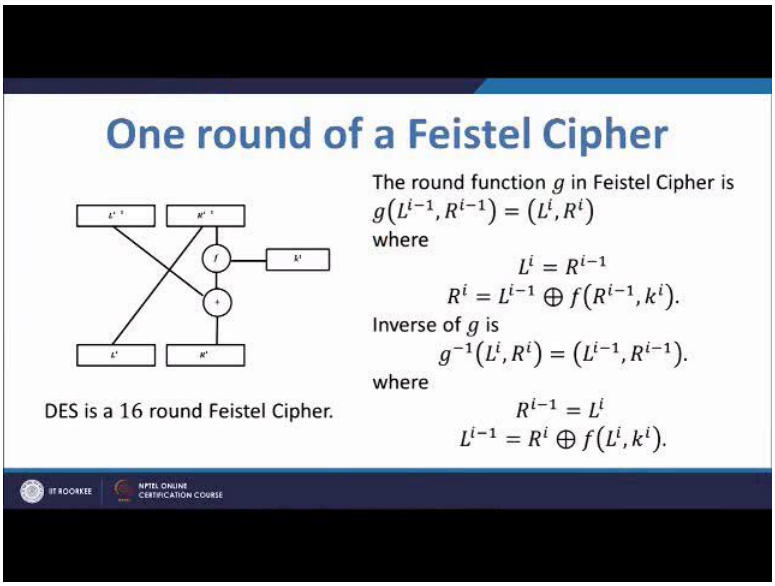
When I say 1, it is 0 0 0 1. When I say 2, it is 0 0 1 0. When I say 3, it is 0 0 1 1; 4 is 0 1 0 0; 5 is 0 1 0 1; 6 is 0 1 1 0; 7 is 0 1 1 1. I have come half way and the next half is essentially the same thing over here with one in the most significant bit. So, this is 8, this is 0 0 0 0; 9, this is 1 0 0 1. Now, this is a, a is 1 0 1 0; b is 1 0 1 1; c is 1 1 0 0; d is 1 1 0 1; e is 1 1 1 0; and lastly f 1 1 1 1. So, these are my domain points or these are all possible 4-bit inputs and what I do over here is that I write the functional values. At 0, I see that it is getting mapped to C; 1 is mapped to 5; 2 is mapped to 6; 3 is mapped to B; 4 is mapped to 9; 5 is mapped to 0; 6 is mapped to A; and then, 7 is mapped to D.

Then 8 is mapped to 3; 9 is mapped to E; a is mapped to F; B is mapped to 8; and C is mapped to 4; D is mapped to 7; E is mapped to 1; and F is mapped to 2. So, this is a mapping to hexadecimal symbols. I will convert them to bit strings. So, this is C is 1 1 0 0; so this 1 1 0 0. Then 5 is 0 1 0 1; 6 is 0 1 1 0; B is 1 0 1 1; 9 is 1 0 0 1; 0 is of course, 0 0 0 0; A is 1 0 1 0; and 7 is 0 1 1 1; 8 is 3, so, it is 0 0 1 1; 9 is E, so it is 1 1 0. 1 1 1 0; F is all 1; 8 is 1 0 0 0; 4 is 1 0 0 0; 7 is 0. I made a mistake over here, D is 1 1 0 1, 1 1 0 1; 7 is 0 1 1 1; 1 is 0 0 0 1; and lastly 2, that is 0 0 1 0.

So, I have got the image. So, here this is the input bit string to the S-box and this is the output. So, if we now go 2 steps back to the substitution-permutation network that we are looking at, we see that we are putting the S-boxes here; this is that S-box. So, whatever bits come in, it will look at this table, and substitute it with the other bit, other sequence of bits, and transfer it out, and then you will have the permutation. So, this is why it is called substitution-permutation network.

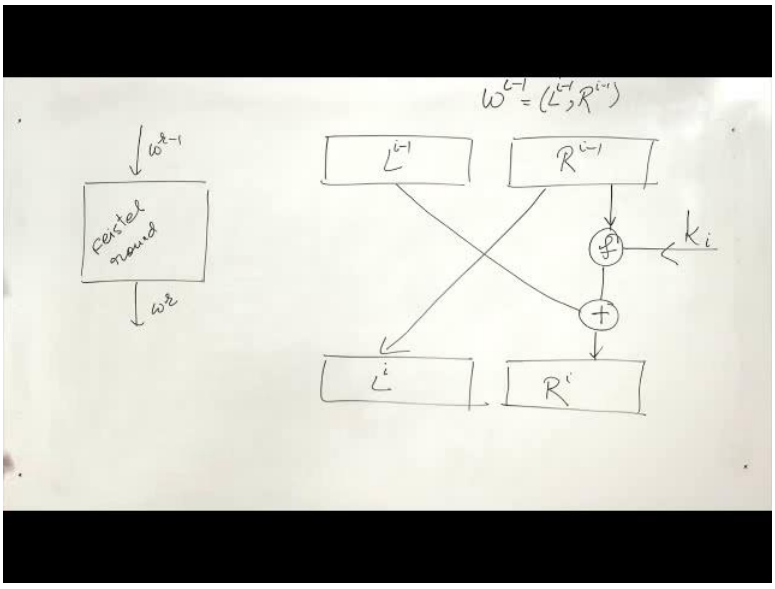
So, you are key mixing which is simple XOR, then substitution, then permutation, and then again key mixing and so on. You will see that substitution-permutation networks can be generalised to construct many different ciphers. So, if you look at the literature in the internet you will find many examples of that.

(Refer Slide Time: 23:24)



There is another general model which is used and which is very famous and very good. It is called a Feistel cipher. Now, we will give a description of just one round of a Feistel cipher.

(Refer Slide Time: 23:47)



In general, if we want to go into several rounds, it will be like this, that in the r th round I have got an input, let us say w r minus 1, and then, I have got something like this where I have this Feistel round, and then, it comes out as w r, and within this I have got this. So, what is coming in, is a block, it may be of any length, it may be 8-bit, it may be 32-bit, it

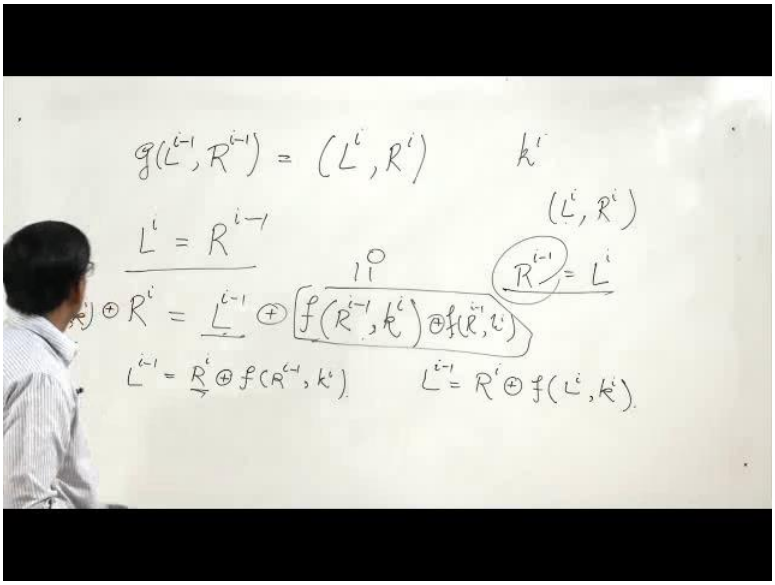
may be 64-bit, anything, but it as to be even. So, we specify the block length to be even, and I will split the block into 2 parts: one we will call the left half and the other I will call the right half. So, in the ith round I will be getting the output from i minus 1th round. So, I write the l i minus 1.

So, the input to the ith round is w i minus 1 which is split up into 2 parts l i minus 1 and r i minus 1. And here, it is r i minus 1, and I have to compute the output which is split up again into 2 parts l i and r i. What we do is that r i is as it is transferred to l i. So, I have a direct link here; there is no tempering, nothing; it is coming to l i and here we have the key. This is the ith round. So, we have the ith round key which is x, which is combined with r i minus 1 with a function f, you may have some XOR in between or something and then some other function, but it is like this, and then this whole thing is XORed with l I, l i minus 1 and push to r i.

So, you have a flow like this: r i minus 1 goes directly to l I, r i minus 1 and k i is taken as argument of a small function f, and that function computes its output, and it will be again having the same size, same length as r i minus 1, which will be bitwise XOR by l i minus 1, and I get r i. So, I have got arrows like this. I have written down in my slide the structure of a Feistel round.

Now, we can write this mathematically by symbols in this way.

(Refer Slide Time: 27:23)



We are saying that the function is l i minus 1, function is g, let us say the whole function here; let us suppose this whole function is g. So, it is operating over an ordered pair, left half and the right half, the ordered pair of that, and then it is giving me as an output the right half is coming to the left half. So, it is quite a moment.

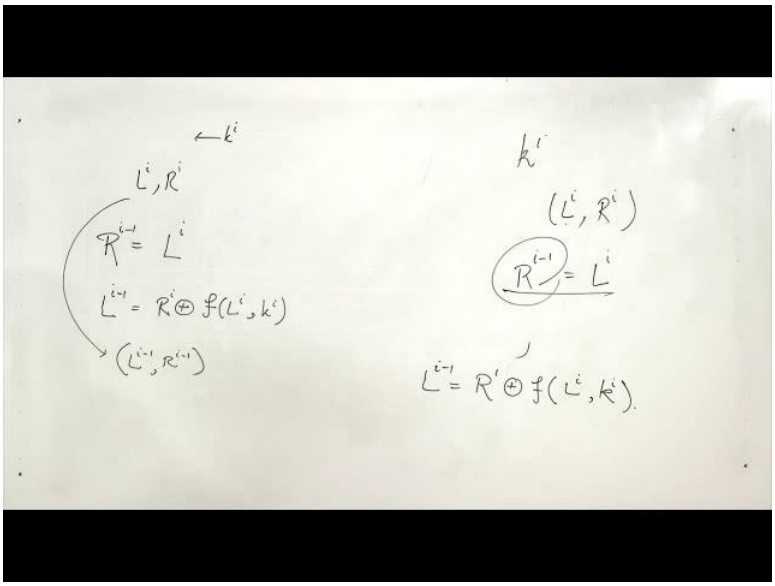
So, the output, let us say, that I am saying that output is l i r I, where l i is forced to be r i minus one because the right half is going to the left half, and r i is l i minus 1 here plus f of r i minus 1 combined by the ith round key which is k i. So, I have got something like this. Now, the question is that this is encryption, but what is decryption? What we will see and which is amazing is that this function is invertible no matter what is; no matter what is the structure of f. So this, just for f we just need a function - any function, the invertibility of the function g has nothing to do with a invertibility of f or otherwise.

It goes like this that suppose I know the round key k I, and I know that at the end I have got l i and r I, and I say that I would like to know l i minus 1 r i minus 1. The question is - Can I do it? And one will say yes, of course, you can do at least half of it because you know these. So, if you know this, you can always go back and say that r i minus 1 is equal to l i. So, this l i is going to be r i minus 1, there no trouble.

Now, 1 would say that let us look at this, I can probably do something because I want to know l i minus 1. I have already obtained r i minus 1, I want to know l i minus 1, but l i minus 1 is over here. So, I can, of course, write l i minus 1 is equal to r i XOR f of r i minus 1 k i. This is because since in XOR 1 plus 1 is 0 and 0 plus 0 is 0. So, if I add here modulo 2 f r i minus 1 comma; wait a moment, k i this quantity, and if i add this quantity here, f of r i minus 1 k I, then this thing will vanish, because this is the same thing added modulo 2. So, this will give me 0, and therefore, I will have this. So, l i minus 1 is this. And then, I will say that, but you know if have got r i. So, I know r i and r i minus 1; I do not know r i minus 1, but then you will say, that oh, but I have found it in the previous step.

So, I can always plug in l i instead of r i minus 1, and therefore, I can always say that l of i minus 1 equal to r i XOR f of r i minus 1 is l i l i into k i l i and k i.

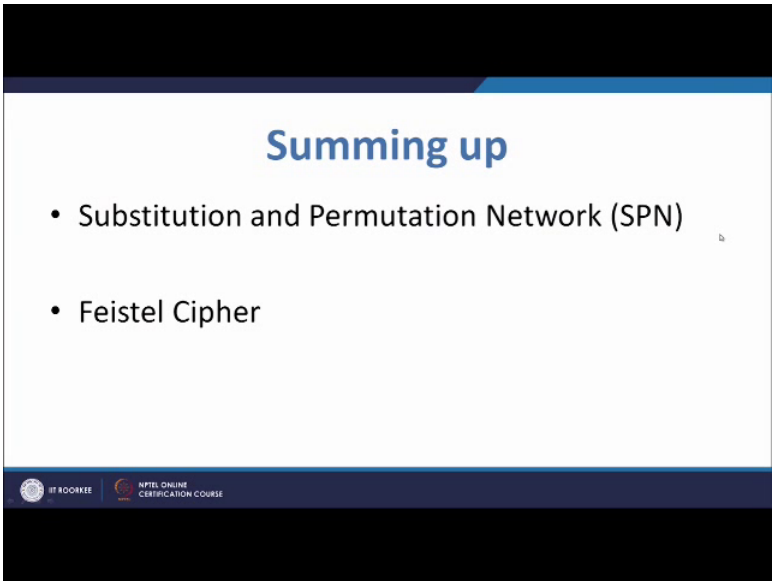
(Refer Slide Time: 31:58)



So, I will remove this portion and write the inverse function. So, if I have got l i and r i, and I know the round key, then I can get r i minus one which is l I, and l i minus one is r i plus f of l i comma k i. So, I can go back from here to l i minus one r i minus one, and therefore, the function that is associated to the Feistel cipher in one round is invertible no matter what is a property of that one function small f.

In summing up in this lecture we have studied substitution-permutation network, in short SPN.

(Refer Slide Time: 33:04)



We have also studied Feistel cipher. We will do some examples in the assignment and the discussion lecture towards end of this week.

Thank you and good bye.