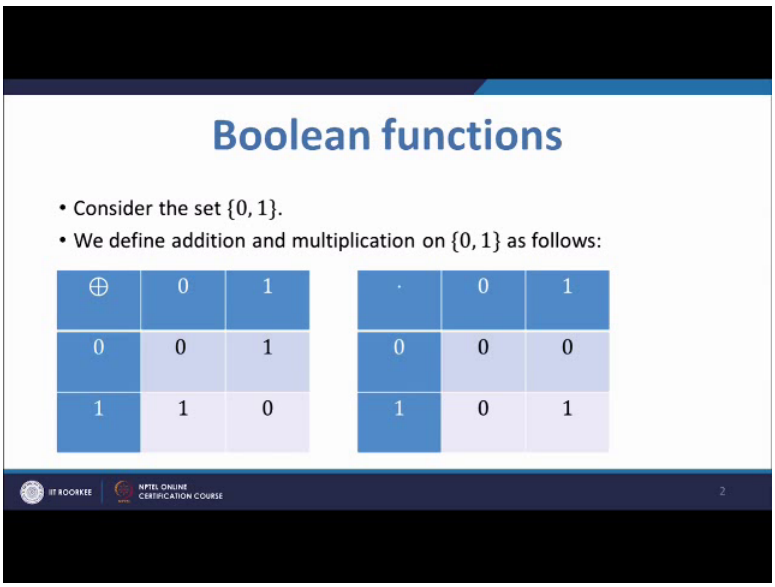
Introduction to Cryptology
Dr. Sugata Gangopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Roorkee

Lecture – 08
S-box theory

Welcome to week 2, Lecture - 3. In the previous lecture we have seen the construction of Block Ciphers, particularly Substitution Permutation Network and Feistel Ciphers. We have also encountered something called an S-box or Substitution box, where a block of bits are accepted as input and which is substituted to another block of bits not necessarily of the same length. We will have a closer look at substitution boxes or S-boxes in this lecture, but before we do that we have to have some idea of Boolean functions. So, we start with Boolean functions.

(Refer Slide Time: 01:22)

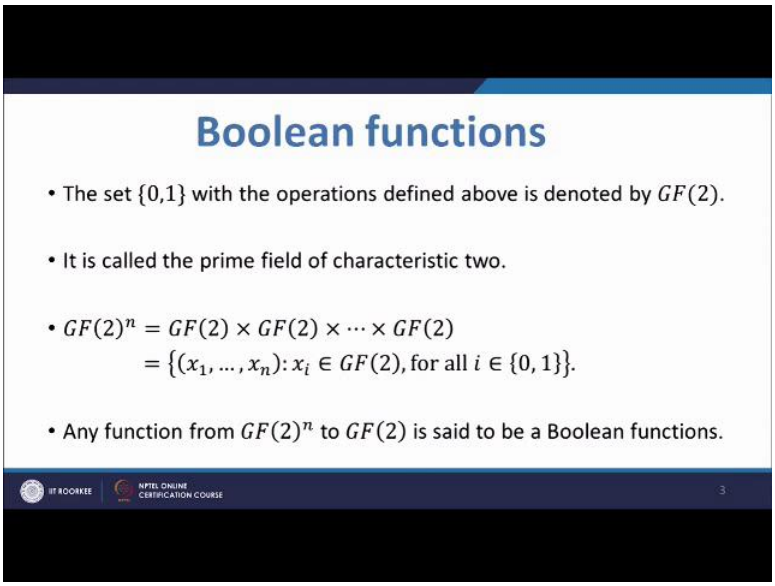


Again to start with Boolean functions we have to know Boolean algebras or a particular type of vector space on which Boolean functions are defined. So here we start from something very simple, we have a set containing just 0 and 1 and on this set we define operations. The first operation is like addition it is called XOR or addition modulo 2, and the rules are given here. And the second operation which we denoted by a dot, but when

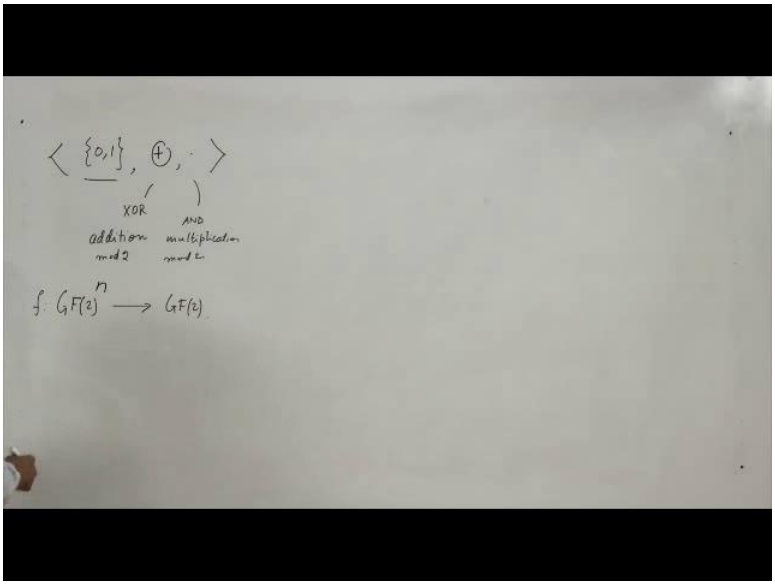
we are writing the formulas we do not write a dot, we just do not write anything just write two symbols side by side to denote this operation this can be called AND or Multiplication modulo 2.

And the rules are given here; the salient point is that of addition modulo 2 to recall again is that 1 plus 1 give me 0. And salient point about this is that 1 into 1 gives me 1 and rest of the entries as 0's.

(Refer Slide Time: 02:35)



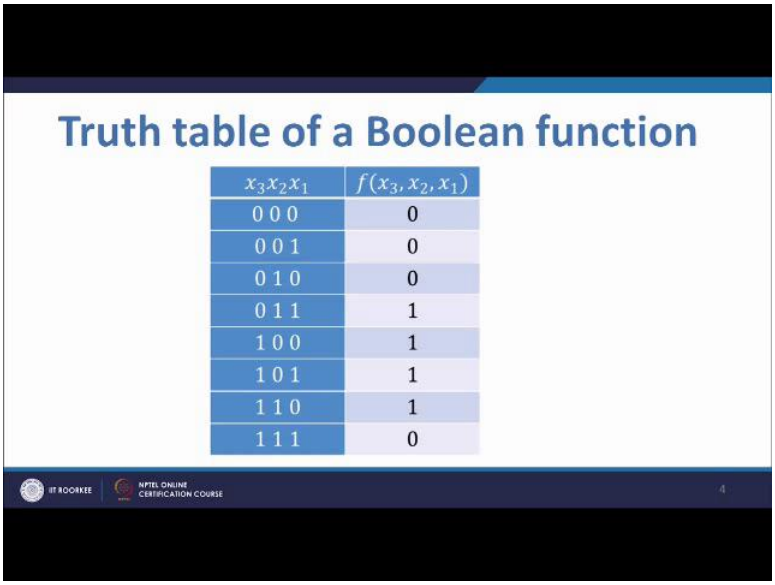
(Refer Slide Time: 02:38)



We say that if he have this set 0, 1 and attach to it the operations XOR or addition modulo 2 over here and this one is called AND or multiplication mod 2. Then we have an algebraic structure which we called GF 2, it is also sometimes called the prime field of characteristic two. We are not going into details about prime fields, why it is called prime fields and what are extension fields and all that, but what we do now is to take Cartesian product of GF 2 n times to obtain GF 2 to the power n which is essentially all binary sequences of length n. Once we have these then we define functions from GF 2 to the power into GF 2 and these functions are called Boolean functions.

So, I take GF 2 to the power n and define functions from GF 2 the power n to GF 2 and this a call Boolean functions. Now as example we see a Boolean function here

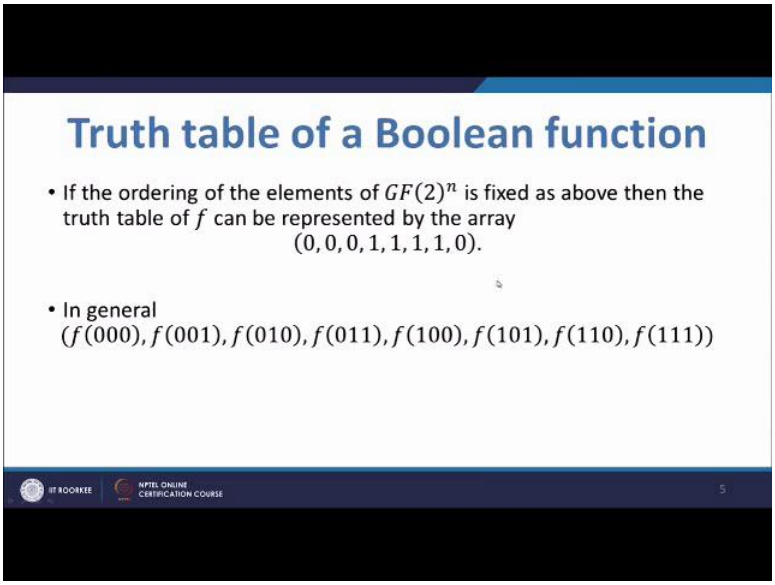
(Refer Slide Time: 04:33)



Now, if you look at these you will see that I have listed down the elements of GF 2 to the power 3 in a particular order and this order is exactly the order in which the natural numbers corresponding to these bit streams, if I consider the right hand side to be the list significant bit and left hand side to be the most significant bit. This is the order in which the numbers come I mean if you are changing these binary strings to natural numbers. So, it start from 0 0 0, 0 0 1, 0 1 0 and so on and ends in 1 1 1. And I can take any string over here of 0's and 1 and that gives me a function.

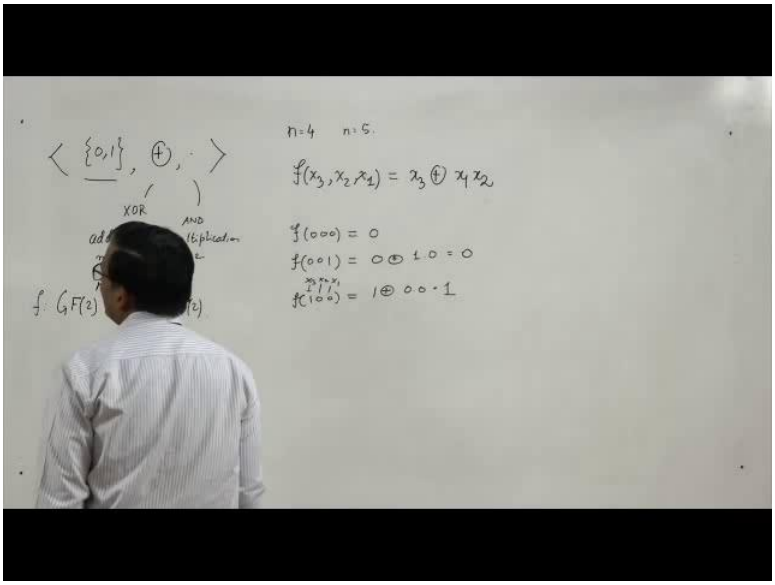
In this particular case have taken a string 0 0 0 1 and 1 1 1 0. This is a Boolean function and this particular representation of Boolean function is called the Truth table. We move forward to see another way of writing a truth table, if I assume that the ordering in which the elements of the domain comes is fixed and this is the order then I can just write the string over here which is the right hand side of the this table.

(Refer Slide Time: 06:13)



We can write one after another and we can say this is my truth table and I can always get the function from it. In general if we have let us say n equal to 3, I can write in general truth table like this. I have written here n please read this as 3. Well, you can read this as GF 2 to the power n, but I am doing this for GF 2 to the power 3.

(Refer Slide Time: 06:55)



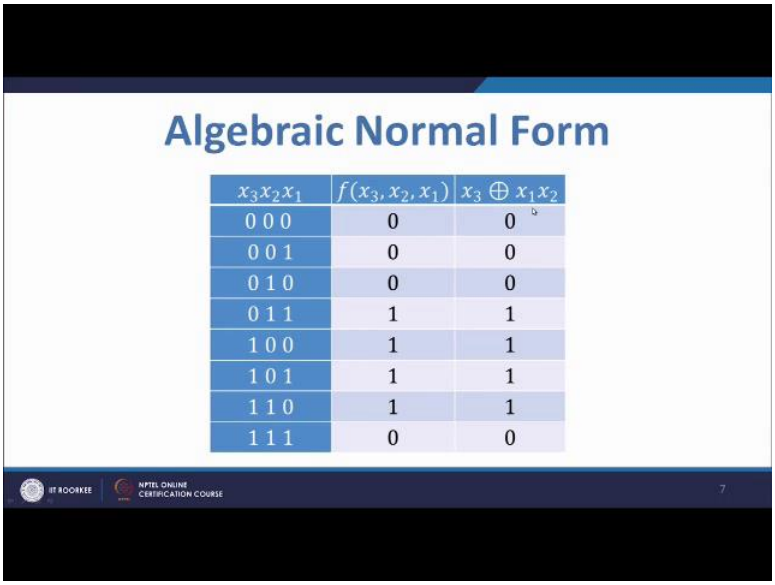
If it is any n then we will have a string of 0's starting from here of length n and then I will write 1 and all 0, then I will write 1 0 and all 0, then 1 1 and all 0 and we know have to proceed. You may try this for n equal to 4 n equal to five etcetera and see the ordering.

Now, we have a concept of algebraic normal form what we can do is that, we can associate variables to the input bits, we have already done that. Please see that this is the left most input bit and I associate x 1 to it then after that I associate x 2 to it and then the next to the last one I associate x 3. If I come here, I can basically write an algebraic expression of this type and that is called algebraic normal form of the Boolean function. The algebraic normal form of Boolean function is unique I am again not going into the proofs of that, but we more or less understand algebraic normal form.

So if we come here, we come back to the same function then we see that if we had evaluated an algebraic expression like this, then if I take the input f 0 0 0 then of course it is 0. Then if I take the input f 0 0 1 then of course here x 1 is 1 and x 3 is 0, so 0 plus 1 times 0 so it give me 0. And we come down and let us say if I take x 1 0 0 we see that this is the input weight corresponding to x 3 this is corresponds to x 2 and this corresponds to 1, therefore will get 1 plus 0 into 0 so it its 1.

In that way we can evaluate all the values and I definitely would advise you to do that and you will see that you are getting the same sequence as this. So, this Boolean function has the algebraic normal form x 3, XOR, x 1, x 2.

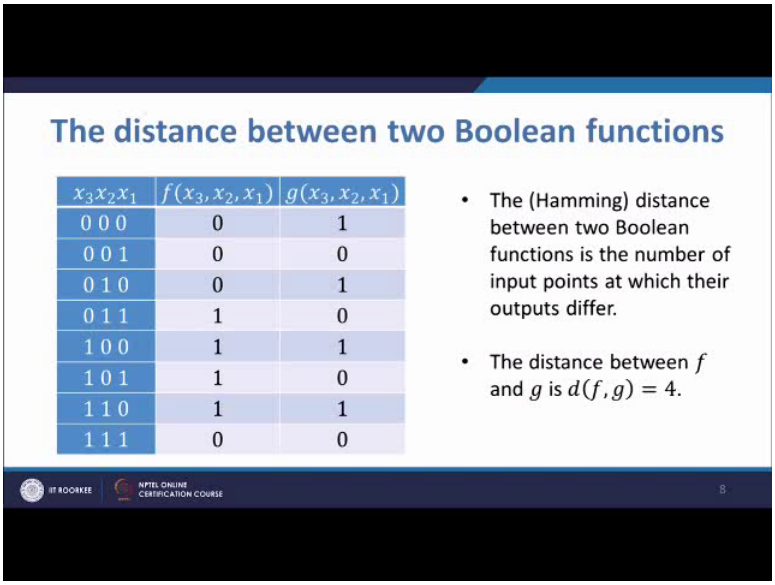
(Refer Slide Time: 09:21)



Now there is of course a question, how to derive algebraic normal form from truth table and truth table to algebraic normal form. I will set this in the assignment for you to check literature to find out rule and I will discuss that rule when I discuss the assignment problems.

So, let us go forward. Now we see that essentially if we fix the ordering of the domain elements then only thing that matters is a sequence in which the functional values appear.

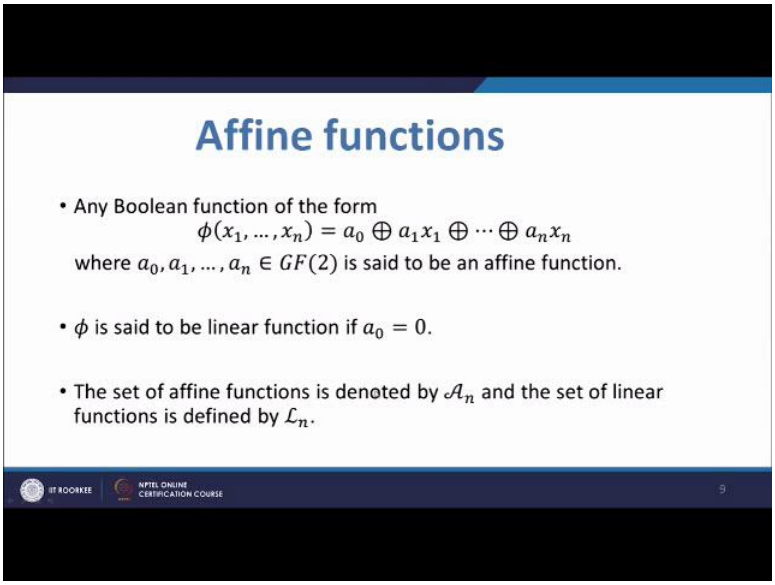
(Refer Slide Time: 10:11)



And suppose, we have got two functions then we can talk about distances between Boolean functions and that has got a name; it is a famous distance it is called Hamming distance. And this distance is exactly the count of the number of places the two functions do not match. So, the hamming distance of f and g is 4 here, because there are 4 places where the functions do not match. If we check in this table we see it in the first place the function do not match and third place the functions do not match, the fourth place the functions do not match, they match here so I have already counted 3 and there is a last place here the 6 place the functions do not match. Therefore, the hamming distance is 4.

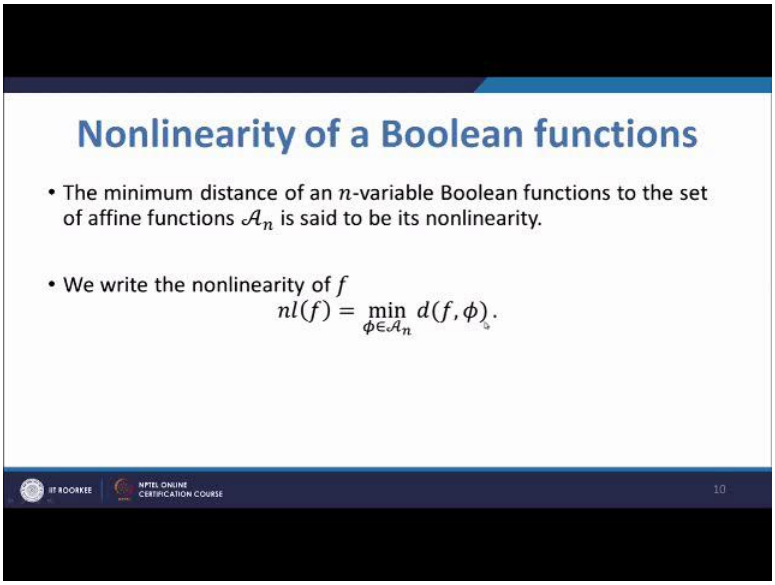
So, next we have Affine functions. Now, we have talked about algebraic normal form. And let us go back if you look at algebraic normal form in general you will see that there are many, many terms over here, particularly there is a constant term without any x and there are terms with only single exercise so to say and there are term with product of two exercise and then ultimately a term at the end which contains a product of all the exercise.

(Refer Slide Time: 12:11)



But if we consider a function whose algebraic normal form contains a constant term here and only the terms which have single variable then that function is called an Affine function. And the set of affine functions is denoted by the symbols script A sub n. And the set of affine functions has a subset which is also very famous this subset it called the set of linear functions and denoted by script L sub n. A function is linear if a 0 is 0. If the constant term is 0 then the function is called Linear.

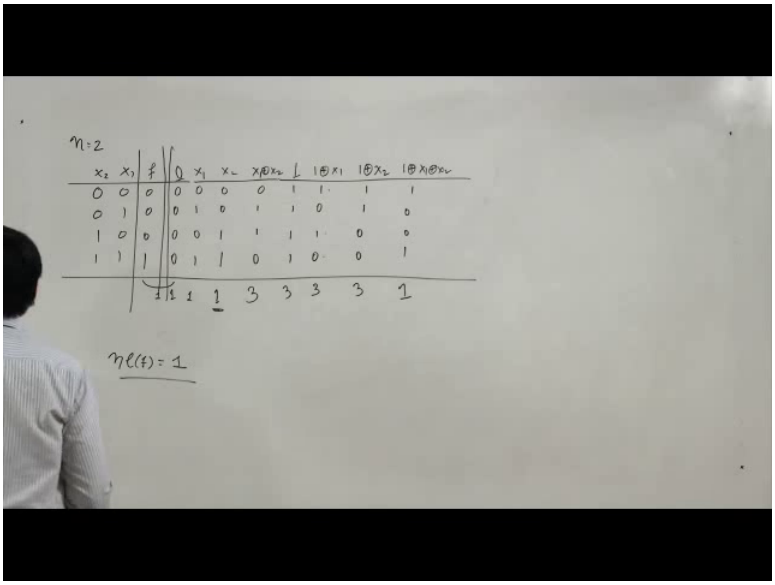
(Refer Slide Time: 13:02)



Now we come to an idea to which I had in mind and I just driving at it, so this idea of nonlinearity of a Boolean function. So, suppose we have a Boolean function we think of it as a string of length 2 to the power n and we are considering the set of all affine functions and we are considering the hamming distances of each affine function to that particular Boolean function. And if we find the list hamming distance, then that list value is called the nonlinearity of the function.

So, I have to consider the set of all affine functions and vary phi over the set of all affine functions and keep f fixed, and nonlinearity of f is a minimum of all d f phi. This needs an example, so let us do that for n equal to 2.

(Refer Slide Time: 14:46)

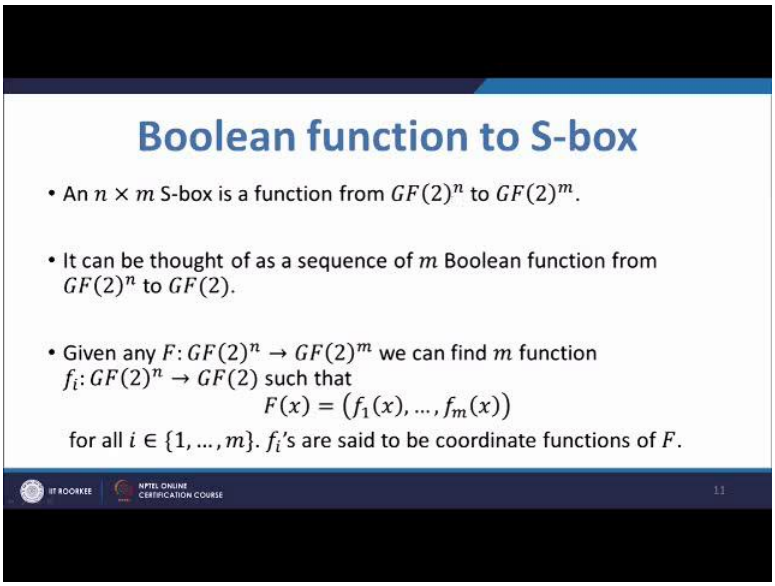


So, I am taking n equal to 2 and my values are; 0 0 0 1 1 0 1 1 these are my domain points, and my function is 0 0 0 1 this is my function and I will now list down all the affine functions. I have got a one special function which is all 0, so I write all 0 over here. And then I have got the function x 1 and x 2. So x 1, is 0 1 0 1 and x 2 is, 0 0 1 1. And I take x 1 XOR x 2 this is 0 1 1 0. And then I take complement of all these functions, so I have got 1 that is for all 1, then this is 1 XOR x 1 it is complement of this so 1 0 1 0. And then I have 1 XOR x 2, so this is 1 1 0 0. And this is 1 XOR x 1 XOR x 2, this is 1 0 0 1.

Now, have to compare this function with all these functions and check the distance. Here the distance is 1, because it defers only in one place. And now see that for example here, the distance is again 1. Then here the distance is again 1, but here please see the distance is 1, 2, 3, 4, so distance is 4. Here the distance is 3. Here the distance it differs over here, here and here the distance is 3. Here again the distance should be 3, yes. And here the distance is 1. And let us say here, yes the distance is no this distance is not 4 I mean like it is wrong here, this is same and here it is different so it is 3.

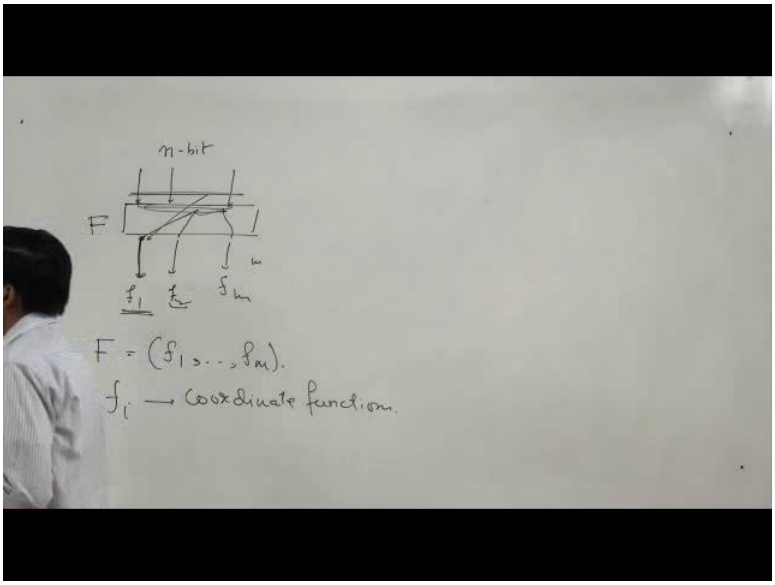
So, there is a pattern like this. Here the distance is 1. So, these are all the affine functions. If you try to compute the nonlinearity, nonlinearity is the minimum distance as this. So, nonlinearity of the function f is going to be 1.

(Refer Slide Time: 19:18)



Now, the question is how we connect Boolean functions to S-boxes, the answer is in the slide. Suppose, I have got an n by m S-box, so that S-box is taking n bit input and giving me m bit output. So, I can consider that S-box has a function from GF 2 to the power n to GF 2 to the power m. So, to say each coordinate functions, let me draw it over here.

(Refer Slide Time: 19:29)



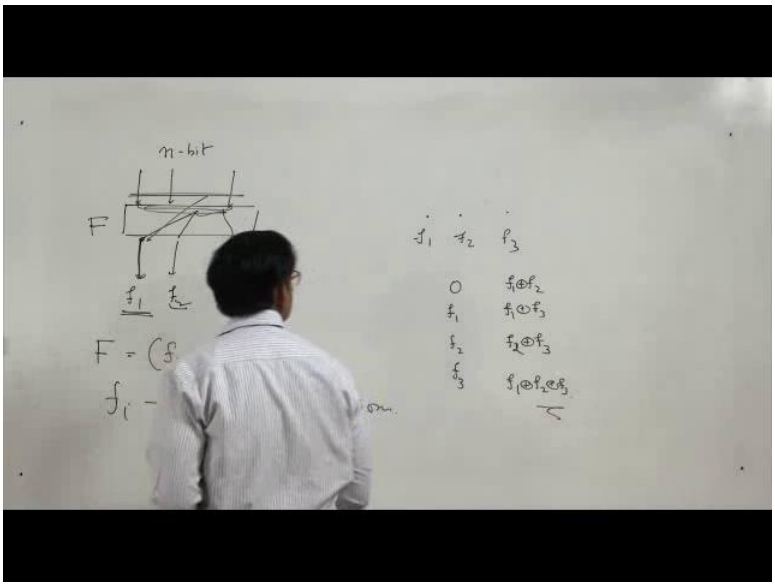
I have got n bit input and some n bits output, so what I am doing I am taking all of these things and concentrating on this I call it f 1 and then I call it f 2 and so on, that is what I have written here. So, f x can be split up into a sequence of Boolean functions f 1 up to f m. Here for example, I have got this input and I will consider only this output and call this a Boolean function f 1, then take this input and consider here this Boolean function f 2 and up to f m. Therefore the function F, suppose this is a function F which is my S-box is a sequence of Boolean functions like this. Now these functions are called coordinate functions of f, f i's - are coordinate functions.

Now what one would presume that if we have to have a good Boolean function then sorry good S-box then it is coordinate functions much be good. In the sense that that should be cryptographically good properties like, possibly the coordinate functions should have good nonlinearity which will make it difficult to approximate by linear functions. But, it is not enough to check only the coordinate functions we have to do a bit more over here, we have to check so called component functions.

So what we have to do is basically to take all the linear combinations of the coordinate functions which is called component functions and which I have written over here. So, I

take a vector in GF 2 to the power n and call it v, and then I am just taking a linear combination by multiply v 1 with f 1 and XOR with v 2 with f 2 and so on.

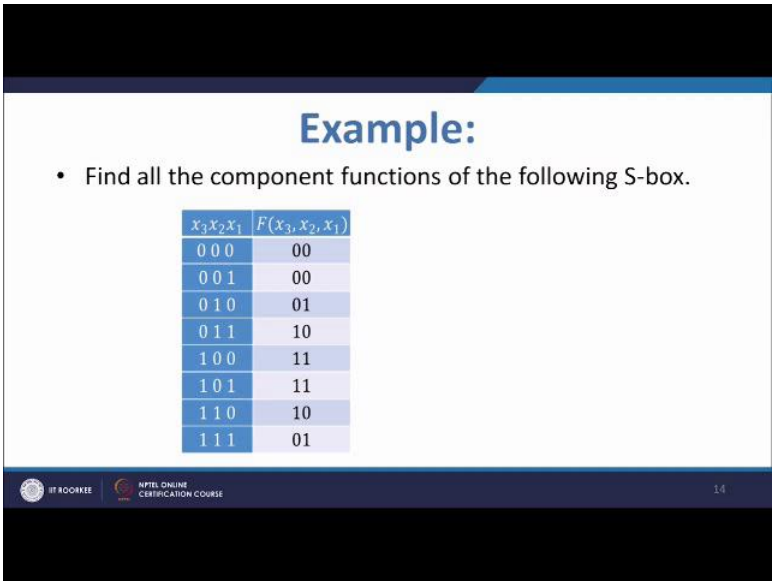
(Refer Slide Time: 22:13)



Let us say if I have got a just three coordinate functions; f 1 f 2 and f 3, then my component functions will be of course the 0 function and the function f 1 f 2 and f 3 which are the coordinate functions. And then I will have f 1 XOR f 2, f 1 XOR f 3, and f 2 XOR f 3, and finally f 1 XOR f 2 XOR f 3 I will have that. And that is what I have written over here and had request you to check that after the lecture.

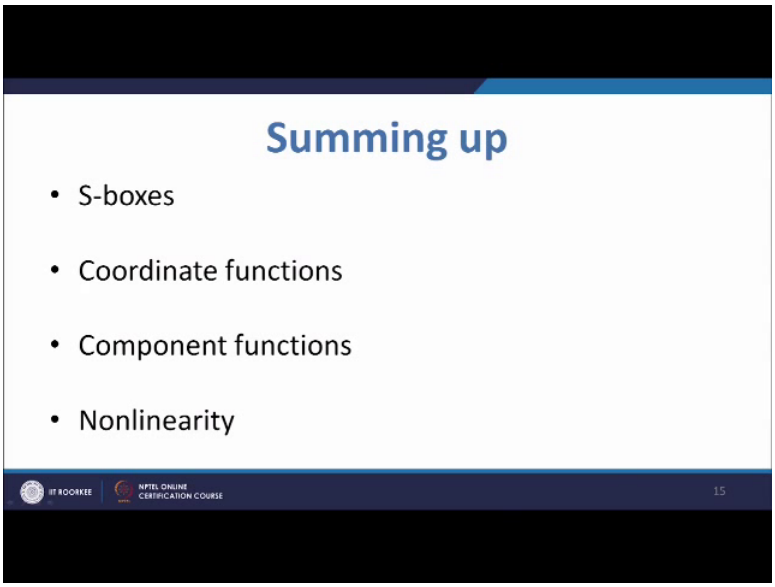
So what we know is that, if we have to have a good S-box from the point of view of cryptography then the coordinate functions are not enough, the component functions have to be strong and by being strong I mean they should have good cryptographic properties. Now what are these properties? Is a very difficult and difficult question and all the properties are not yet known. We will be discussing some of the properties in a next lecture, probably one we will be discussing only one property that is nonlinearity in the next lecture and try to understand how nonlinearity controls the cryptography vulnerability.

(Refer Slide Time: 24:36)



Now, there is one example that we can check. I give you an S-box so it is a 3 by 2 S-box it is completely listed over here and I will ask you to find all the components functions. That is an exercise.

(Refer Slide Time: 24:57)



And we have come to the end of today's lecture to sum up; we have studied S-boxes, we have studied Coordinate functions, we have studied Component functions, and most importantly we have introduced the idea of Nonlinearity. That is all for today.

Thank you.