

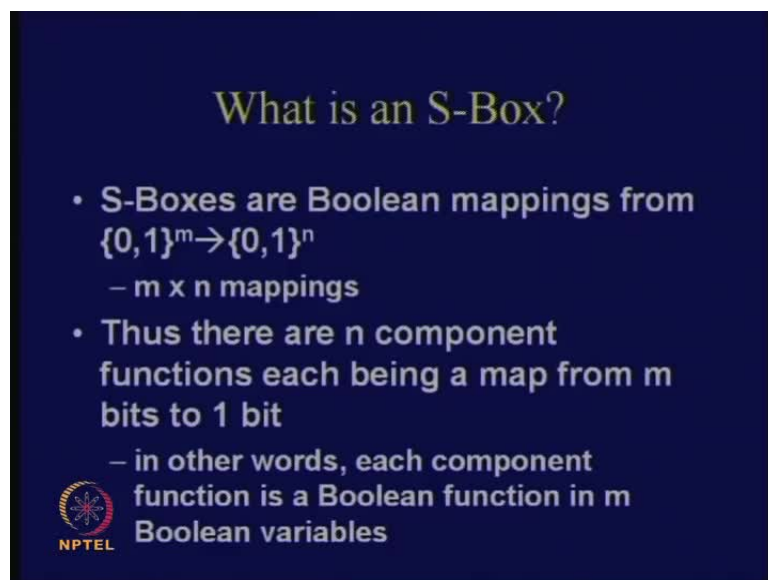
Cryptography and Network Security
Prof. D. Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. #17
Overview on S-Box Design Principles

So, in today's class we shall be discussing about the principles of designing an S-Box. So, you have seen that the S-Box plays a very key role to the security of the block ciphers; I am talking in terms of linear and differential cryptanalysis. So, today's class, the objective will be to understand some of the design criteria that the S-Box must satisfy.


So, what we have seen essentially till now, is that S-Box is a table with elements, which have been filled by entries, which look to be quite random; so, **arbit** values, but actually they are not random and there is actually quite deep science involved behind the design. So, we will try to understand how the S-Box is designed or rather, at least see some of the features or which the S-Box should satisfy some of the properties, which the S-Box should satisfy.

(Refer Slide Time: 01:09)



What is an S-Box?

- S-Boxes are Boolean mappings from $\{0, 1\}^m \rightarrow \{0, 1\}^n$
 - m x n mappings
- Thus there are n component functions each being a map from m bits to 1 bit
 - in other words, each component function is a Boolean function in m Boolean variables

 NPTEL

So, first of all, we will go straight to the question that - what is an S-Box? **What is an S-Box?** So, the S-Box we have already seen essentially till now, it is a mapping, it is a Boolean mapping from m bits to n bits; so, therefore, it takes a m bit input and it produces an n bit output; so we can also call it a m cross n mapping.

So, therefore, essentially, you can imagine that all the n outputs that you produce essentially are individual Boolean functions. So, they operate upon n Boolean inputs and each of these components functions essentially is a map from m bits to 1 bit.

So, in other words, each component function is a Boolean function in m Boolean variables. So, Boolean variables, means, the values can be 0 and 1. So, therefore, you see that **each of the output** each of the n component output bits can be either 0 or can be 1. So, therefore, we have to understand that how essentially these Boolean functions are designed.

(Refer Slide Time: 02:21)

Boolean Function

- A Boolean function is a mapping from $\{0, 1\}^m \rightarrow \{0, 1\}$
- A Boolean function on n-inputs can be represented in minimal sum (XOR +) of products (AND .) form:

$$f(x_1, \dots, x_n) = a_0 + a_1 \cdot x_1 + \dots + a_n \cdot x_n + a_{1,2} \cdot x_1 \cdot x_2 + \dots + a_{n-1,n} \cdot x_{n-1} \cdot x_n + \dots + a_{1,2,\dots,n} \cdot x_1 \cdot x_2 \cdot \dots \cdot x_n$$
- The ANF form is canonical...
- If the **and** terms have all zero co-efficients we have an **affine** function
- If the constant term is further 0, we have a **linear** function

NPTEL

So, we will try to address these issues in today's class; so that is the objective. So, let us little bit go back into what we know about Boolean functions and we know that Boolean function is essentially means what? So, if there are m input bits, I would like to obtain one corresponding output bit. So, in a Boolean function, you know, you can represent that in the form of a truth table. So, the truth table was a canonical representation, which means, that it was a unique kind of representation of a given Boolean function. So, once given a Boolean function, **I can always know that,** we know that by using, we can

express it, express it using either min terms or max terms. So, basically, we know that we are quite familiar with representation, which is known as AND, OR representation. So, means, I am expressing something - a Boolean function using AND, OR as my logic operations.

So, when we talk about cryptography and actually cryptographic Boolean functions, then another representation is commonly used, it is known as the AND, XOR form, or the AND, XORed expression. So, it is called the **ANF** form or the Algebraic Normal Form. So, how does an algebraic normal form look like? It looks like this, so, therefore, it is quite, I mean - if you observe in this way so, for this plus. In this case, actually, this plus does not mean OR, it means an XOR. So, all these operations are XOR operations and we have got XOR's and **we have** we have also got ANDs like you see $x_1 \text{ AND } x_2$.

So, similarly, you have got the final term as $x_1 \text{ AND } x_2$ and so on, till x_n . So, all these coefficients, like coefficients running from a 0, a 1 till a n, then you have got a 1,2 a 1,3 and similarly, **like** you go on till, a n minus 1,n and then finally, for the final term like, x_1 to x_n you have got an $a_{1,2}$ and so on till n. Therefore, all these coefficients can take two values, it can either be 0 or it can either be 1. So, how many such Boolean functions are possible? We can easily enumerate from this, because you know that there are how many possible Boolean functions, 2 to the power of n.

So, therefore, you see that these Boolean functions essentially comprise of **what will**, we can actually represent all possible Boolean functions using this form, and also, this form is a normal form. So, what does it mean? It means that **if I represent them in this I mean**, if I would like to represent, I can obtain a unique representation just like my truth table, this is an equivalent representation and actually, there are algorithms, quite easy algorithms, which takes a given truth table and can represent them in this particular form.

So, I am not going to those conversions, but you can easily do that. So, maybe you can take it just as homework exercise and you can just try to find out some algorithms for doing that.

This just represents coefficients; so, therefore, this can be values of this so, this is just how the notation has been denoted. So, when you are considering the term x_1 into x_2 then the corresponding coefficient is, $a_{1,2}$. Similarly, when we are considering the

coefficient of, say, x^{n-1} and x^n , then I represent that by, a_{n-1} . So, that is only a notation, one thing, therefore, these values can either be 0s and 1s.

So, you note one thing that, essentially, we will be addressing linear functions and non-linear functions and also another function, which is known as **affine** functions. So, what is affine function? We have already seen how an affine function looks like. So, you see that in this particular form of the equation, if you substitute all these values which have got AND terms as 0s, I mean to say, the coefficients corresponding to this AND terms are 0s, then you have got, a 0 XORed with a 1, x^1 and so on, till a x^n . So, apart from these, all the other terms go to 0. So, **what is the corresponding? for this particular**, for this equation which is remaining, is actually an affine equation. So, if this a 0 value, that is a constant value also goes to 0, then you have got only these terms, then it actually becomes **a linear equation** a linear function.

So, therefore, apart from this, all other functions are non-linear and, you know, why it is non-linear? It is non-linear with respect to the XOR operation; so, we have to address this in context to linear cryptanalysis, you know, what is a linear function and why is called linear and if it is non-linear, why it is non-linear? It is always with respect to a particular operation.

(Refer Slide Time: 07:41)

x_1	x_2	$y = x_1 \vee x_2$	$x_1 \oplus x_2 \oplus x_1 x_2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

$y = x_1 \oplus x_2 \oplus x_1 x_2$ — ANF.

So in this operation, our case is XOR. So, how many possible linear functions are possible? We can also enumerate from this particular expression, this description so, you

can work out how many linear functions are possible, how many non-linear functions are possible, and so on. So, I can give you a simple example. So, it is good to see some example like, suppose you have got x_1 and x_2 as two variables and let us consider the simple function of x_1 OR x_2 . So, how the truth table will look like, 0 0 0 1 1 0 and 1 1, so, apart from this, all the other terms will be 1s; so, this is simple OR function. So, this particular thing you would like to express using ANF form. So, let us just try how we can represent this. So, I call this function as y and just see that y is equal to x_1 XORed x_2 XORed with $x_1 x_2$, does it work?

So, you see what is, x_1 XORed x_2 XORed with $x_1 x_2$, does it match? So, you see **that I can actually represent this as** this particular notation or this particular representation is known as the ANF representation. So, therefore, there are actually algorithms, you can easily express them. I mean, given AND OR form, you can actually convert that into AND XOR form and vice versa also.

(Refer Slide Time: 09:38)


Boolean Function

- A Boolean function is a mapping from $\{0,1\}^m \rightarrow \{0,1\}$
- A Boolean function on n -inputs can be represented in minimal sum (XOR +) of products (AND .) form:

$$f(x_1, \dots, x_n) = a_0 + a_1 \cdot x_1 + \dots + a_n \cdot x_n + a_{1,2} \cdot x_1 \cdot x_2 + \dots + a_{n-1,n} \cdot x_{n-1} \cdot x_n + \dots + a_{1,2,\dots,n} \cdot x_1 \cdot x_2 \cdot \dots \cdot x_n$$

- The ANF form is canonical...

If the **and** terms have all zero co-efficients we have an **affine** function
 If the constant term is further 0, we have a **linear** function



(Refer Slide Time: 09:45)


Boolean Function

- A Boolean function is a mapping from $\{0,1\}^m \rightarrow \{0,1\}$

$f: \Sigma^n \rightarrow \{0,1\}$ be a Boolean Function.
Binary sequence $(f(\alpha_0), f(\alpha_1), \dots, f(\alpha_{2^n-1}))$
is called the Truth Table of f

- Sequence of a Boolean Function:

$\{(-1)^{f(\alpha_0)}, (-1)^{f(\alpha_1)}, \dots, (-1)^{f(\alpha_{2^n-1})}\}$ is called sequence of f




So, therefore, we have understood what is an ANF form? So, why is an ANF form is so important? The ANF form is important for various reasons: so, one of the important reasons is that you can easily obtain the degree of a particular Boolean equation. So anyway, we will be going to this various properties, which the Boolean function, or cryptographic Boolean function should satisfy one after the other. So, consider a Boolean function which is a mapping from m bits to one particular bit. So, let f essentially, which takes in a n bit number and convert that into a 0 1 bit, be a Boolean function. So, therefore this is a simple example of a Boolean function and then we define something which is known as the binary sequence.

(Refer Slide Time: 11:30)

x_1	x_2	$y = x_1 \vee x_2$	$x_1 \oplus x_2 \oplus x_1 x_2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

$y = x_1 \oplus x_2 \oplus x_1 x_2$ — ANF.

Sequence $\langle (-1)^0, (-1)^1, (-1)^1, (-1)^1 \rangle$
 $= \langle 1, -1, -1, -1 \rangle$.

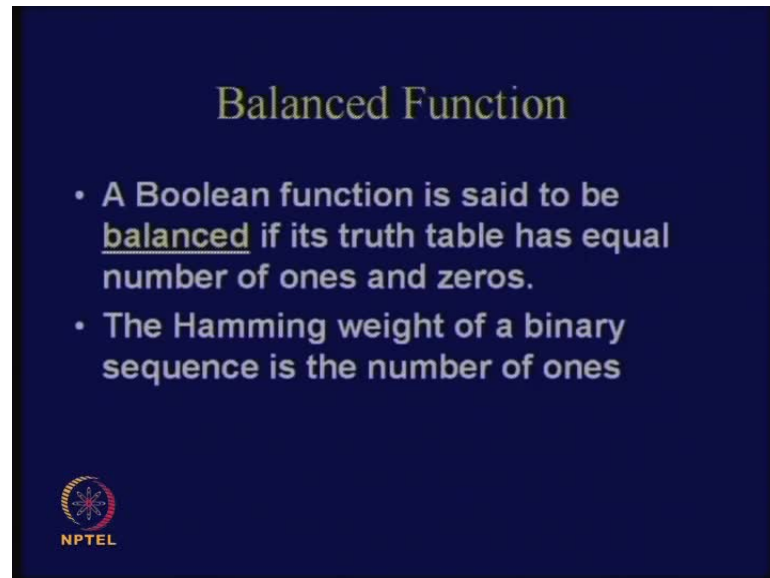


So, binary sequence essentially means I would like to represent them in the form of f alpha 0, f alpha 1, and so on till f alpha 2 power n minus 1. So, therefore, this particular notation, we know commonly as the truth table. So, what does it mean? I will take every possible assignments of my input and find out what is the corresponding output, if the input is been sensitized by that particular value. Like for example, if this Boolean function f , if I give an input of alpha 0, the corresponding output is f alpha 0 and how many such possible inputs can I give? I can give 2 power n possible inputs and **I stored them**, I store all the outputs in a table which is commonly known as the truth table; so that is the definition of a truth table.

So, **there is some** there is another quite closely stated term which is called the sequence of a Boolean function. So, what is the sequence? So, the sequence is quite **simple I mean quite** similar to the truth table, only thing is that each element is represented by minus 1 to the power of that particular element like, minus 1 to the power of f alpha 0, minus 1 to the power of f alpha 1 and so on till **minus alpha to the power of** minus 1 to the power of f alpha 2 to the power of n minus 1. So, what you do is that, you take the truth table elements and you raise each element to the power of minus 1. So, therefore, what will be the sequence in this particular example? So, for example, if the truth table elements are 0 1 1 1, then what will be my sequence? It will be minus 1 to the power of 0, minus 1 to the power of 1, minus 1 to the power of 1, and minus 1 to the power of 1, so, therefore, I will write this as 1, minus 1, minus 1 and minus 1.

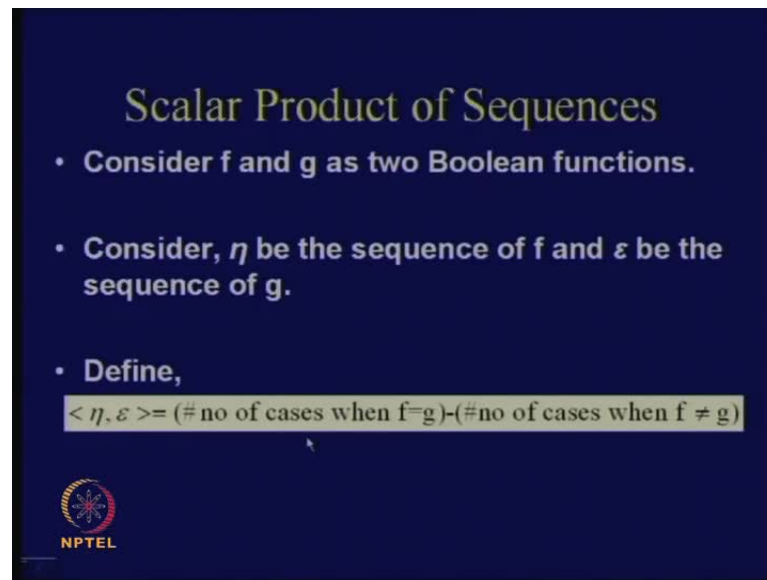
So, therefore in a sequence, each of the elements can be either plus 1 or minus 1; so, that is how a sequence of a truth table is defined. A truth table of a sequence of a truth table for a given Boolean function is defined.

(Refer Slide Time: 12:09)



So, then we will see certain criteria which a good Boolean function should satisfy. See, as I told you that for all attacks, the main objective was to find distinguishers, distinguishers from random matrix. So, we know that, if you take a random function, then what you expect is the number of zeros and ones should be essentially equal - right? So, one of the criteria, which a Boolean function should satisfy and which is called obvious also is that, the number of zeros and ones should be same and that is called the balanceness of a function. So, a balanced function is a Boolean function if its truth table has got equal number of zeros and ones.

(Refer Slide Time: 13:17)



The slide has a dark blue background with yellow and white text. At the top, the title 'Scalar Product of Sequences' is written in yellow. Below it, there are three bullet points in white. The first two are 'Consider f and g as two Boolean functions.' and 'Consider, η be the sequence of f and ϵ be the sequence of g.' The third is 'Define,' followed by a light blue box containing the formula $\langle \eta, \epsilon \rangle = (\# \text{ no of cases when } f=g) - (\# \text{ no of cases when } f \neq g)$. At the bottom left, there is a circular logo with a star and the text 'NPTEL' below it.

So, therefore, we also **note this note this** know this term, called hamming weight. So, therefore, if I have got a binary sequence, so the hamming weight of the binary sequence is the number of ones. So, therefore, you know that the number of ones and number of zeros for a given particular truth table should essentially be identical; so, therefore, if you have got an n bit Boolean functions, then how many number of ones are expected?

2 to the power of n by 2; so that is 2 power of n minus 1 that is the expected number of ones. So then, we define something which is called as scalar product of two sequences. So, what is a scalar product of two sequences? We have defined what is a sequence, therefore, given a Boolean function f and g, we can define the sequences by these two symbols; so, one of them I call eta and the other one is say, epsilon.

So, **now, we define, so,** therefore, if you take the scalar product, so you know what is the scalar product defined as. So, therefore, this is the notation of the scalar product, then you can see this is equal to the number of cases when f and g are equal minus number of cases when f and g are not equal. So, **all of these here** therefore, this is the dot product or the scalar product of the two sequences. So whenever you see that f and g are both same, then that will add up to this particular product and when f and g are not the same, then actually subtract minus 1 from that.

Because you note that your sequence has got only two elements, it has got either plus 1 or it has got minus 1. So when they are not equal, then you have got 1 in one case and

minus 1 in the other case, and when both of them are same then you have either got 1 1 or minus 1 minus 1; so in **both the** both the cases, the product computes to plus 1. So, **therefore, you have got either,** therefore, this you can enumerate by the number of cases when f and g match, minus number of cases when f and g do not match. Do you follow?

(Refer Slide Time: 15:15)

Non-linearity

- The non-linearity of a Boolean function can be defined as the distance between the function and the set of all affine functions.


$$\therefore N_f = \min_{g \in A_n} d(f, g)$$

where A_n is the set of all affine functions over Σ^n

$$d(f, g) = 2^{n-1} - \frac{1}{2} \langle \eta, \epsilon \rangle$$

$$\therefore N_f = 2^{n-1} - \frac{1}{2} \max_{i=0,1,\dots,2^n-1} \{ |\eta_i| \}$$

where l_i is the sequence of a linear function in x




(Refer Slide Time: 15:37)

x_1	x_2	$x_1 \vee x_2$	0	x_1	x_2	$x_1 \oplus x_2$
0	0	0	0	0	0	0
0	1	1	0	0	1	1
1	0	1	0	1	0	1
1	1	1	0	1	1	0

$d_1 = 3$
 $d_2 = 1$
 $d_3 = 1$
 $d_4 = 1$

$NL = \min(d_1, d_2, d_3, d_4) = 1$



So, therefore, using this particular notation, we will define something which is called a non-linearity. So, all this time we have been talking about non-linearity but we have not really quantized what non-linearity means. So, I will take an example to atleast address

you what is non-linearity. I think, after that, this particular slide will make better sense. So, take this example like $x_1 \times x_2$, and so, we know that there are possible values like 0 1 1 0 and 1 1 and take the same example $x_1 \text{ OR } x_2$ and we will try to find out what is the corresponding non-linearity of this Boolean function. So, what is the truth table? We have seen 0 1 1 and 1; so the idea is that you find out all possible linear functions in these two variables x_1 and x_2 . So, what are the possible linear functions possible? One of them is 0, like, all of them is 0. So, the other possible linear function could be x_1 itself, x_2 itself, and $x_1 \text{ XORed with } x_2$; so they are considering linearity with respect to the XOR operation. So, what are the corresponding truth table for this, 0 0 0 0, for x_1 it is equal to 0 0 1 1, for x_2 it is 0 1 0 1, and for $x_1 \text{ XORed } x_2$ it is 0 1 1 and 0. So these are my possible enumerations of the possible linear functions.

(Refer Time Slide: 16:53). Next, what I do is that I take this particular truth table and I take all the linear functions or **linear** possible linear functions and I find out what is the hamming distance of these truth tables **with this hamming** with this truth table; so, that is why, I defined the term - hamming distance. So, what is the corresponding hamming distance of these truth tables? How many cases do they differ - that is the thing, right?. So, how many cases do they differ? They differ in three cases. So that is d_1 , I call them as 3. **how many cases so how do I mean what is this** How many cases that these value differ from this particular column? It differs in one case. How many cases this differs? So, it differs in 1 case. How many case does these differ? Yeah, it is 1.

So, non-linearity is defined as the minimum of d_1 , okay, so, we call them d_1 , d_2 , d_3 and say, d_4 ; d_1 , d_2 , d_3 , d_4 , so in this case that is equal to 1. So, that is the definition of non-linearity; so, which means that now, we can generalize this. Therefore, if you have got n variable Boolean function, then you obtained all possible linear approximations, enumerate their truth tables and find out the distances. And out of them, the minimum distance corresponds to the non-linearity of the Boolean function. So, you can note certain interesting things, like for example, if you take two linear functions, how many cases do they differ? It is always half the possible number of cases so, you see that there are four 0s and there are two differences. Like so, in two cases they differ. Similarly, if you take these two linear functions, how many cases do they differ? It also differs in two cases; you take these two things, it also differs in 1 case here and 1 case here, so 2 cases. So, all possible linear functions it will differ in $2^n - 1$ cases.

So, actually you will find that a linear function and a non-linear function will the distance I mean the So, what we have essentially considered is about the distance. So, you will find that the distance is never equal to $2^n - 1$ but it is always less than $2^n - 1$. So, only 2 linear functions differ in $2^n - 1$ cases, but a non-linear function, can actually differed in less than. So, it is assumed you can just assume as if the linear functions forms a space so forms various access, and non-linear functions comes somewhere in between; therefore, designing a non-linear function is quite tricky. So, if you make it fall away from one particular linear function, it may fall close to the other linear function, but what you have to ensure is that the non-linear functions maintains quite a safe distance from all possible linear functions.

(Refer Slide Time: 20:13)

Non-linearity

- The non-linearity of a Boolean function can be defined as the distance between the function and the set of all affine functions.


$$\therefore N_f = \min_{g \in A_n} d(f, g)$$

where A_n is the set of all affine functions over Σ^n

$$d(f, g) = 2^{n-1} - \frac{1}{2} \langle \eta, \epsilon \rangle$$

$$\therefore N_f = 2^{n-1} - \frac{1}{2} \max_{l=0,1,\dots,2^n-1} \{ |\eta, l_l| \}$$

where l_l is the sequence of a linear function in x

 NPTEL

So, therefore, our idea will be essentially or rather, our objective will be to maximize this non-linearity. So, we have to maximize this particular metric called non-linearity. So, how do you do that? But before we go into that, let us define generally what non-linearity means. So, the non-linearity of a Boolean function can be defined as the distance between the function and the set of all affine functions, but actually, it is a minimum distance. So, therefore you take f and you find out all possible affine functions and I denote them by, say g , and then you find out the distances corresponding between them. You denote the minimum of them, assign minimum of them to a non-linearity. So, non-linearity is defined in this way - so you see that N_f is equal to minimum of the from f comma g , where g belongs to A_n , where A_n is the set of all affine functions over Σ^n

n. Sigma n means somewhat **over n**, therefore, it is an alphabet which is formed using n particular letters, n particular symbols; forget the notation, but the idea is simple as that.

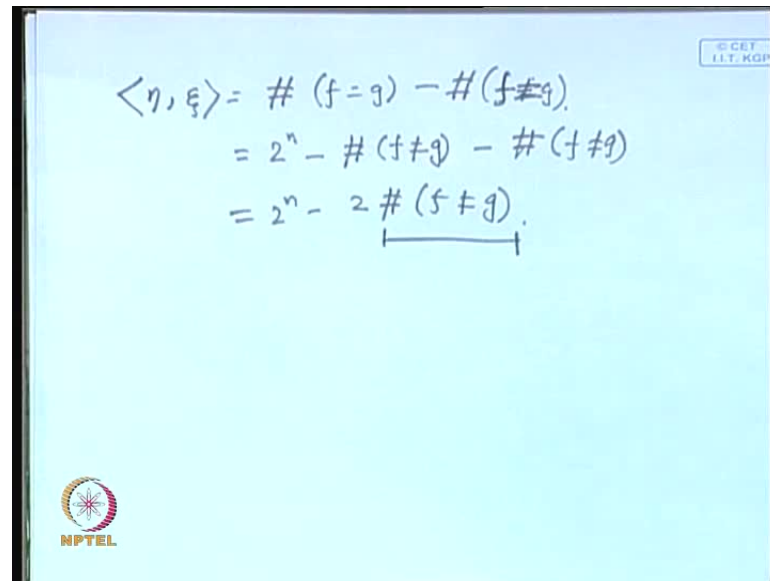
So, now, you can try to understand why your $d_{f,g}$ is equal. So, what we are essentially concerned with the distance of f with g. So, your distance with f,g will be equal to $2^{n-1} - \frac{1}{2}\epsilon$. So, that follows from the eta definition of this particular dot product.

So, I leave to you as an exercise to think - why? Then it follows straight away from the definition. Therefore, if you know this, then defining non-linearity becomes quite simple, because in that case, non-linearity becomes equal to $2^{n-1} - \frac{1}{2}\epsilon$ of this distance, and what we are interested is, making this distance minimum. So, **what we will do** if you make the distance minimum, then we will make this particular scalar product as maximum; therefore we write here maximum and that is how it is defined.

So, therefore, you see that what we absorbed intuitively, but the non-linearity is always less than 2^{n-1} . It is minus half of something, so, therefore, this term is actually a positive term. So, we will find that this non-linearity is always less than 2^{n-1} .

So, therefore, you can just give a **thought that why it works**. Any idea why it is correct or shall I show it to you? So, I will just give a hint and leave **the complete the entire thing to if leak the entire** completing the proof to you.

(Refer Slide Time: 22:55)



The image shows a whiteboard with handwritten mathematical equations. The equations are:

$$\begin{aligned}\langle \eta, \xi \rangle &= \#(f=g) - \#(f \neq g) \\ &= 2^n - \#(f \neq g) - \#(f \neq g) \\ &= 2^n - 2 \#(f \neq g).\end{aligned}$$

The whiteboard also features a small logo in the bottom left corner with the text "NPTEL" and a copyright notice in the top right corner: "© CEE I.I.T. KGP".


So, how was it defined? This was defined as number of cases when f is equal to g minus number of cases when f is not equal to g . Now, what we are interested is, the number of cases when f is not equal to g . So, how can you write, express these in terms of this? It is equal to 2 power n minus number of cases when f is not equal to g . So, you know, you have also got minus number of cases when f is not equal to g ; so, you have basically got 2 power n minus 2 times number of cases when f is not equal to g . Now, you can actually rearrange this, because this is what you are interested in, right? You are interested in the distance between f and g ; so you are interested in the distance means, you are interested in finding out these values. Now, you can just rearrange this and you will obtain the corresponding expression.

(Refer Slide Time: 24:05)

A Compact Representation of all the linear functions

- Hadamard Matrix: Any $r \times r$ matrix with elements in $\{-1, 1\}$ if $HH^T = rI_r$, where I_r is the identity matrix of dimension $r \times r$.
- Walsh Hadamard Matrix:

$$H_0 = 1, H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}, n = 1, 2, \dots$$
- Each row of H_n is the sequence of a linear function in x belonging to $\{0, 1\}^n$
- Each row, l_i is the sequence of the Boolean function, $g(x) = \langle \alpha_i, x \rangle$, α_i is the binary representation of i .
 Note that α_i and x are not sequences, but they are binary tuples of length n



(Refer Slide Time: 24:29)

Handwritten notes on a whiteboard showing the construction of Hadamard matrices:

$$H_0 = 1, \quad H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

1-variable sequences: $0, x$

2-variable sequences: $1, 1, 1, -1$

NPTEL logo is visible in the bottom left corner.

So, now, there is actually just we will address this, that there is a very nice commutarial tool called Hadamard Matrix, through which you can actually enumerate or compactly represent all possible linear functions. Therefore, if you see the definition of Hadamard Matrixes, if you see the definition of Hadamard Matrixes, you will see that the definition of Hadamard Matrixes works like this. You take H_0 equal to 1 and your H_2 or rather H_1 , is defined as $H_1 \ H_1 \ H_1$ and minus H_1 . So, what does it mean? It means, 1 1 1 and minus 1; similarly, you can also write H_2 . So, what will be H_2 equal to? 1 1 1 minus 1 1 1 1

minus 1 1 1 1 minus 1 and you take the negative of this minus 1 minus 1 minus 1 and plus 1.

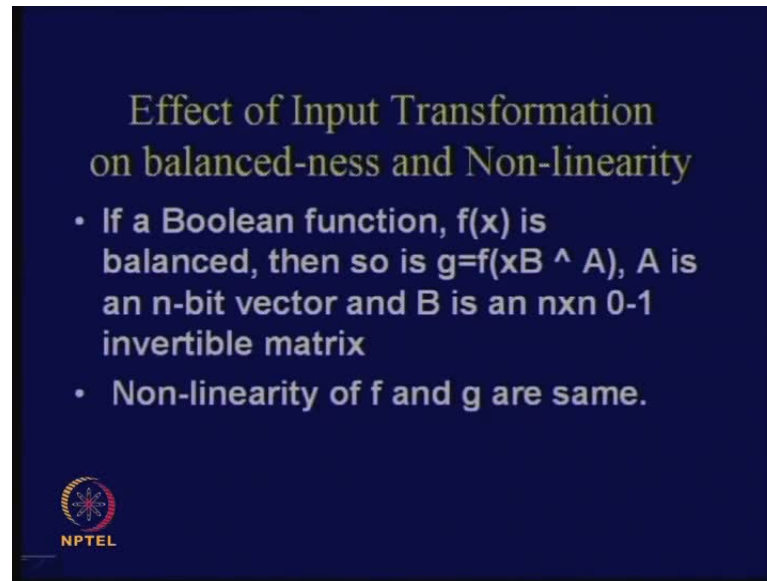
So, why are we doing all these things? So you see that Hadamard Matrix has got a nice property. The property is that, each of the rows, so, if you take each of the rows, then they denote the sequence of a So, if you are considering for example H1, then each of the rows will denote the sequence of a one variable Boolean function. So, what is the one variable Boolean function? Possible, it can be either 0 or say x.

So, what is the corresponding sequence for 0? So, if you consider one variable Boolean function, so what are the possible Boolean functions, what are the linear functions possible? It is either 0 or x. So, what are the sequences possible? What are the possible sequences? It is either 1 1 or it is 0 1; so, what is the corresponding sequence for 0 1?

It is minus 1, 1 minus 1. So, 1 minus 1. So, you see that precisely that is what your Hadamard Matrix gives. It gives you 1 1 and 1 minus 1. Similarly, you see that, you see that for you can extend that to H2 also; first one represents all possible. So, therefore, you see 1 1 1 1, so that represents the linear function 0. So, next one is 1 minus 1, 1 minus 1 so, that is, x 1 for example, what about this, 1 1 minus 1 minus 1? So that is another variable say, x 2 What about this 1 minus 1 minus 1 1? That is x 1 XORed with x 2. So, you see that you can form all two variable functions, two variable linear functions in this fashion.


So, therefore, if I am able to find out - recursively express H_n , in terms of H_{n-1} , then essentially, each of the rows of the matrix H_n denote linear functions of n variables, the sequence of the linear functions of n variables. So, what you have to do is essentially, if you have to obtain the corresponding. So, therefore, this is a very compact expression so, if I want the corresponding linear expression for a given row, then what I have to do is that, I have to take the corresponding binary representation of this, so, therefore, if I know that for example, 1 minus 1 1 minus 1 is my sequence, then I can obtain the binary sequence from this and the binary sequence would have been 0 1 0 and 1 and then, I have got my variable list as, say. So, I have got my corresponding variable list so, therefore, what is the variable list? It is Therefore, if I take the dot product of this with the corresponding Boolean function, then I can obtain from this, that is, my Boolean function is equal to x 1.

(Refer Slide Time: 28:30)



Effect of Input Transformation
on balanced-ness and Non-linearity

- If a Boolean function, $f(x)$ is balanced, then so is $g=f(xB \wedge A)$, A is an n -bit vector and B is an $n \times n$ 0-1 invertible matrix
- Non-linearity of f and g are same.


NPTEL

Similarly, I could have obtained for this particular row, it is equal to x_2 and for this one, it is equal to x_1 XORed x_2 . So, given the Hadamard Matrix, from there I can obtain all possible linear functions. So, therefore, if you told to generate all possible linear functions, then that is a nice recursive. We have doing that using this commutarial tool. Now, we are addressing the issue of non-linearity. So, you see that, **if there is**, so there are some properties which we will just address, but will leave the proof butut, it just for this, just of timing, just try to understand that it works. Try to basically understand the insight, so it says, that if a Boolean function $f(x)$ is balanced, then so is $f(x) \oplus B$ XORed with A , so A is an n bit vector and B is an n cross n 0-1 invertible matrix. So, it means that if you have got f function who has got a certain amount of non-linearity, then doing an affine mapping in the inputs does not really make any change to the non-linearity.

(Refer Slide Time: 29:41)

Strict Avalanche Criteria

- Informally, if one bit input is changed in an S-Box, then half of the output bits should be changed
- For a function, f to satisfy SAC the following condition is satisfied:
$$f(x) \oplus f(x \oplus \alpha) \text{ is balanced, where } wt(\alpha)=1$$
- Higher order SAC, when more than one input bits change

Both the SAC and the higher order SAC together make Propagation Criteria (PC)

NPTEL

So, if you do an affine mapping to the inputs, **this** the input variables x using an invertible matrix, that is, if this particular matrix, B is invertible and if you take this and if you do this operation, still then, the non-linearity does not change. Therefore, what you say is that non-linearity of f and g are the same. So, therefore, non-linearity really does not change to operations like this, but our S-Box should also satisfy other very important criteria, which is known as the strict avalanche criteria, but I think I have told somewhere what is meant by avalanche effect. So, it means that, if I change only 1 bit of the input, half of the output gets changed, but the thing is that formally, we will try to define.

So, what he says is that, if there is a Boolean function $f(x)$, and you define a vector called α , whose weight is one, so what does it mean? That only 1 bit is changed. So, if you take $x \oplus \alpha$, so if **its weight is** weight of α is 1 means, it is basically a binary string in which only one of them is 1, the rest are 0s. So, if you take $x \oplus \alpha$, so you see that, in $x \oplus \alpha$ in x , how many bits are changed? Only 1 bit is flipped; so, this particular function, that is, **$f(x \oplus \alpha)$** , $f(x \oplus \alpha)$ is actually a balanced function; so, that is the criteria of strict avalanche criteria.

(Refer Slide Time: 30:54)

The slide contains the following handwritten text:

$$f(x) \quad f(x \oplus \alpha)$$
$$\alpha = \langle \alpha_1, \alpha_2, \alpha_3 \rangle \quad \checkmark$$
$$\alpha = \langle 0, 1, 0 \rangle \quad (w + \alpha = 1)$$
$$x \oplus \alpha = \langle x_1, x_2 \oplus 1, x_3 \rangle$$
$$= \langle x_1, \bar{x}_2, x_3 \rangle \quad \checkmark$$

$f(x) \oplus f(x \oplus \alpha)$ — Strict Avalanche Criteria (SAC)

— Propagation Criteria of Order 1.

Logos for NPTEL and IIT KGP are visible in the bottom left and top right corners of the slide respectively.

So, it says that, if there is a function, say for example, $f(x)$ and there is a function called $f(x \oplus \alpha)$. So, let us try to understand, why only 1 bit is changed. So, you check, for example x , as say, x_1, x_2 and x_3 , and here α as say, $0, 1, 0$. So, what does $x \oplus \alpha$ represent as? It is equal to x_1, x_2 XORed with 1, and x_3 . So, you know for the properties of XOR that x_2 XORed 1 means what? Complement of x_2 . So, you can represent this as x_1, x_2 complement, x_3 . So, you see that, **from this x** between this x and this $x \oplus \alpha$, that is only 1 bit which has been changed. And essentially, the idea is that $f(x) \oplus f(x \oplus \alpha)$ should be a balanced function.

So, essentially, should be still balanced; and what we say is that, there is also another way of representing this, so we call this as strict avalanche criteria or often referred as SAC. When we also say that it satisfies propagation criteria, something which is called propagation criteria of order one, order one why? Because the rate of α was equal to 1; so, the rate of α was equal to 1 in this case.


So, similarly, you can understand since I have said propagation criteria of order one, I can also have propagation criteria for order two. So, what does it mean? I will just make the rate of α to be two and check for a similar property.

(Refer Slide Time: 33:05)

Strict Avalanche Criteria

- Informally, if one bit input is changed in an S-Box, then half of the output bits should be changed
- For a function, f to satisfy SAC the following condition is satisfied:
$$f(x) \oplus f(x \oplus \alpha) \text{ is balanced, where } wt(\alpha)=1$$
- Higher order SAC, when more than one input bits change

Both the SAC and the higher order SAC together make Propagation Criteria (PC)

 NPTEL

So, idea is that we are always trying to find out distinguishers from a random mapping. So in a random mapping, if you change one bit of the input, you are expect to get half of the output bits to get changed. So, this idea, what we are trying to do as a designer is try to maintain that all the property looks **meaningly** random; trying various kinds of test and ensuring that really the function looks like random. So, you see that when we define something which is called **f x** f x XOR alpha and say that weight of alpha is 1, we say that the S-Box or the Boolean function satisfy what is we know as the strict avalanche criteria, but higher order SACs are also possible when more than one input bits change. So, both the SAC and the higher order SACs together make what something which is called as the propagation criteria. **So, you see that can you** Can you see that where these propagation criteria can be useful in terms of linear and differential crypt analysis? Immediately you can see that if you maintain all these things, then you essentially get a good differential property.

Because what you doing here is basically subjecting the input to a differential, various kind of differentials; so, remember your different distribution table. So, what do you do in your difference distribution table? You are subjecting the input to differential, means what? You are considering various cases where you have given a differential to the input and you are checking that what is the possible output differential. So, basically, what you are doing is that taking one instance of the cipher or the S-Box and applying an input and checking its output and maintaining a differential, say alpha, and giving an input, say x

XOR alpha, and again obtaining the output, and then you are obtaining what is the corresponding differential between $f(x)$ and $f(x) \oplus \alpha$. If you can show that half of the cases, I mean, if it is a uniform distribution, **then you are not** then you will get a distribution table, which is seemingly quite strong; which is I mean which you really cannot exploit so much for doing a differential attack. So, if you see **that** the previous study of non-linearity was helping something to welcome linear crypt analysis, whereas this particular study is helping to somewhat protect against differential crypt analysis.


(Refer Slide Time: 35:03)

How to make a Boolean Function satisfy SAC?

- Consider a Boolean function, $f(x)$
- Consider a non-singular $\{0,1\}$ matrix of dimension $n \times n$.
- If for each row of the matrix A if:

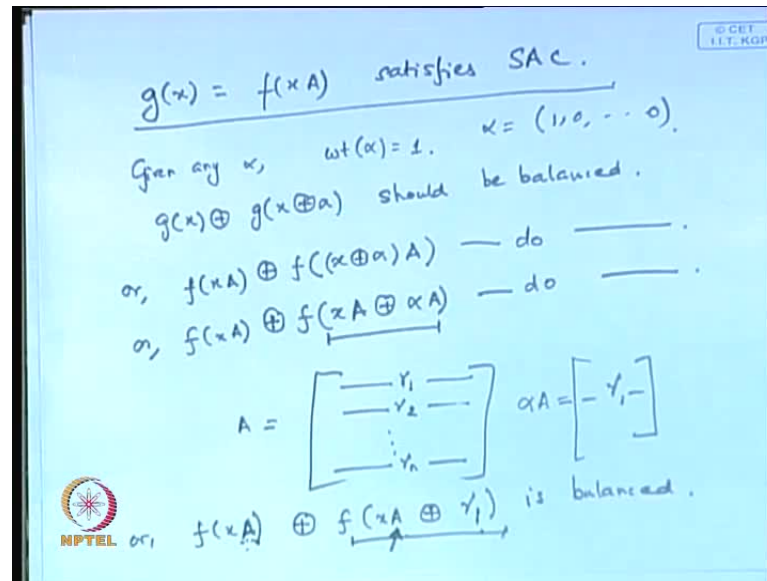
$f(x) \oplus f(x \oplus \gamma)$ is balanced, γ is a row of the matrix A

then $g(x) = f(xA)$ satisfies the SAC.

 NPTEL

So, therefore, given any Boolean function, we can actually make certain transformations and make it satisfy SAC. So, what it says here is, consider a Boolean function $f(x)$ and consider a non-linear singular matrix, say $0,1$ matrix of dimension n cross n . So, what it says that for each way of the matrix A , if this property satisfies, that is, $f(x)$ XORed with $f(x)$ XORed with γ is balanced, where γ is the row of the matrix A , then actually $g(x)$, which is equal to $f(xA)$ satisfies the SAC. So, do you see that why it is true? So, also, it is called trivial if you follow the definition of SAC.

(Refer Slide Time: 35:47)

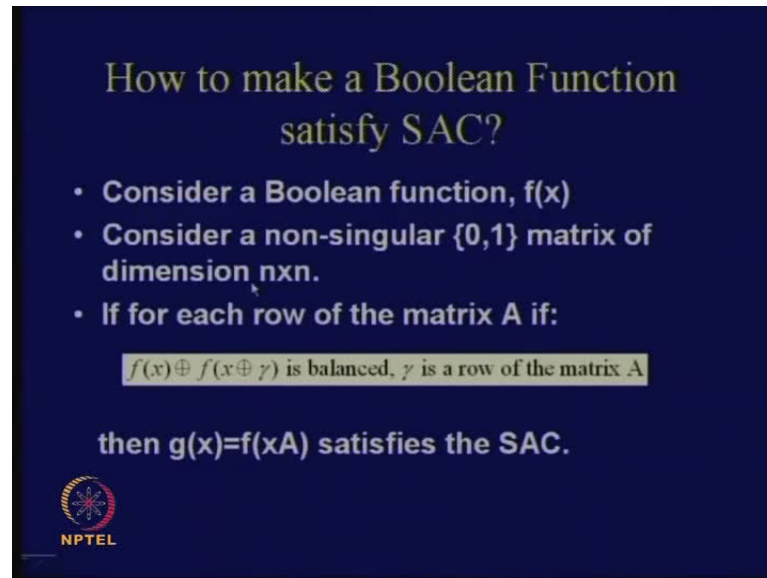


So, let us see what it says - that $g(x)$ is equal to $f(xA)$ satisfies SAC. So, we have **we have** to try to understand why it does so - right? So, if you say, if you claim that $g(x)$ satisfies SAC, then what is the property that $g(x)$ should satisfy? Given any **alpha** such that weight of α is equal to 1, $g(x)$ XORed with $g(x \oplus \alpha)$ should be balanced. So, what is $g(x)$? This means that $f(xA)$ XORed with $f(xA \oplus \alpha A)$ should be balanced. Instead of x , we have written $x \oplus \alpha$. So, you see that what it means is, $f(xA)$ XORed with $f(xA \oplus \alpha A)$ should be balanced; and what is the property that you have? First of all, what is $xA \oplus \alpha A$? What is αA ? So, give any matrix A , I define that A , suppose A has got various rows like γ_1 , so suppose A is a matrix, you have got γ_2 , and similarly, you have got say, some γ_n rows, so what does αA represent? So, remember that α has got weight of only 1, so what does it represent? What does α mean? **Column or row**?

It means so for example, you can just imagine that α is equal to 1 0 and so on 0. So, what does αA represent? **First row**? You are **you are you are you are you are** having α at the beginning and then A - right?. So, you have got the first row, so you have got γ_1 here, only one row which is γ_1 . So, what you are basically saying, is that $f(xA)$ XORed with $f(xA \oplus \gamma_1)$ is balanced, and what was my design criteria? Design criteria was the $f(x)$ XORed with $f(x \oplus \gamma)$ is balanced. Another thing is important here, is that, this non-singularity property of the matrix - why? Because, what we have showed here is that $f(xA)$ XORed with $f(xA \oplus \gamma_1)$

gamma 1 is balanced; and in order to ensure that this particular function $f \times A$, if this therefore, if this matrix A is a non-singular matrix, then this this function will look exactly like the $f \times$ XORed gamma function.

(Refer Slide Time: 39:04)




How to make a Boolean Function satisfy SAC?

- Consider a Boolean function, $f(x)$
- Consider a non-singular $\{0,1\}$ matrix of dimension $n \times n$.
- If for each row of the matrix A if:

$f(x) \oplus f(x \oplus \gamma)$ is balanced, γ is a row of the matrix A

then $g(x) = f(xA)$ satisfies the SAC.



So, if this function A is a non-singular matrix, then $f \times A$ and $f \times$ will be in the identical way, there will be one to one relation between the both between both. Therefore, what you see that, if the condition is that there is a non-singular matrix of dimension n cross n , and such that, if each row of the matrix A , if $f \times$ XORed $f \times$ XORed gamma is balanced, then $g \times$ equal to $f \times A$, satisfies the strict avalanche criteria.


(Refer Slide Time: 39:34)

Example

- $f(x) = x_1 x_2 \oplus x_3$ does not satisfy SAC?
- Why? Consider $\alpha = (001)$
- $f(x) \oplus f(x \oplus e_1)$ is balanced, $e_1 = (100)$
- $f(x) \oplus f(x \oplus e_2)$ is balanced, $e_2 = (010)$
- $f(x) \oplus f(x \oplus e_3)$ is balanced, $e_3 = (111)$

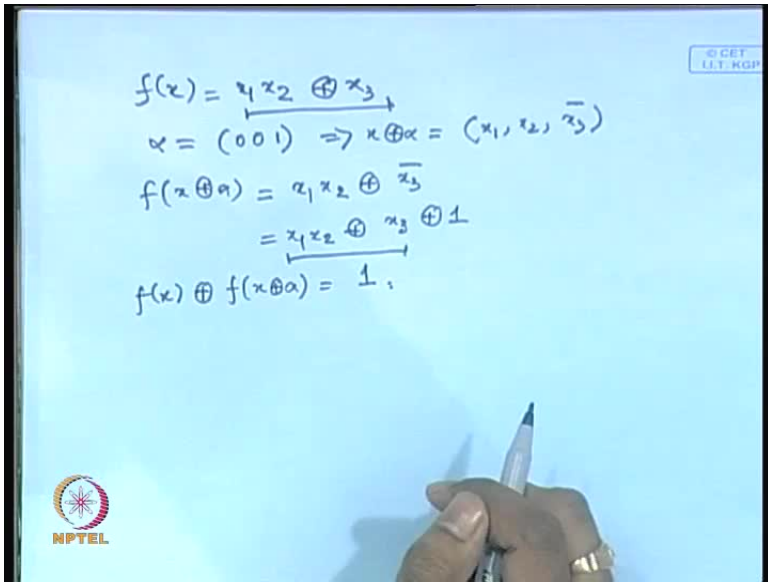
$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Check that $g(x) = f(xA)$ satisfies SAC




So, we will just see one example in order to understand. So, consider like x_1 into x_2 XORed with x_3 , which does not satisfy SAC - why? Just consider this example, say α is equal to 001 , so **if you address this**, if you obtain $f(x)$ and I would like to obtain $f(x \oplus \alpha)$, then what does it mean? I am not actually flipping x_1 x_2 , but only flipping x_3 ; so what does it become? It becomes $x_1 x_2$ XORed x_3 XORed with 1 .

(Refer Slide Time: 40:15)



$f(x) = x_1 x_2 \oplus x_3$
 $\alpha = (001) \Rightarrow x \oplus \alpha = (x_1, x_2, \bar{x}_3)$
 $f(x \oplus \alpha) = x_1 x_2 \oplus \bar{x}_3$
 $= x_1 x_2 \oplus x_3 \oplus 1$
 $f(x) \oplus f(x \oplus \alpha) = 1$



So, if I now take the XOR between these two are obtain one, and which is not a balanced function. Do you see this? Therefore, what am saying is that $f(x)$ is equal to $x_1 x_2$

XORed with x_3 . So consider α is equal to 0 0 and 1. So, what is f XORed with α ? What is x XOR α first of all? It is equal to x_1, x_2, \bar{x}_3 , right? So, therefore, that means, that f XOR α is $x_1 x_2$ XORed with \bar{x}_3 , so what does it mean? It is equal to $x_1 x_2$ XORed with \bar{x}_3 XORed with 1, I can write in this fashion. So, now, if you take f XORed with f XOR α then what do you obtain? You obtain only 1, because this part and this part gets cancelled.

(Refer Slide Time: 41:33)

Example

- $f(x)=x_1x_2 \oplus x_3$ does not satisfy SAC?
- Why? Consider $\alpha=(001)$
- $f(x) \oplus f(x \oplus e_1)$ is balanced, $e_1=(100)$
- $f(x) \oplus f(x \oplus e_2)$ is balanced, $e_2=(010)$
- $f(x) \oplus f(x \oplus e_3)$ is balanced, $e_3=(111)$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Check that $g(x)=f(xA)$ satisfies SAC

NPTEL

So, now is this function a balanced function? No, because everything is 1, therefore, it does not satisfy the strict avalanche criteria. So, we will see how we can make it satisfy strict avalanche criteria. So, which means that we have to find out that particular matrix A ; and here is an example so, without going how to find, we will see that it works like you take A , where A is equal to 1 0 0 0 1 0 1 1 1. So, what you have to do is that you have to find out e_1, e_2 , and e_3 such that, f XORed with f XORed e_1 is balanced, f XORed f XORed e_2 is balanced, and f XORed f XORed with e_3 is also balanced. So, therefore, here also, some example e_1, e_2 , and e_3 , and you note that all these rows or rather all these vector are also linearly independent. Why is that important? Because we would also make the matrix A as a non-singular matrix. It should be an invertible matrix, it should have full rank.

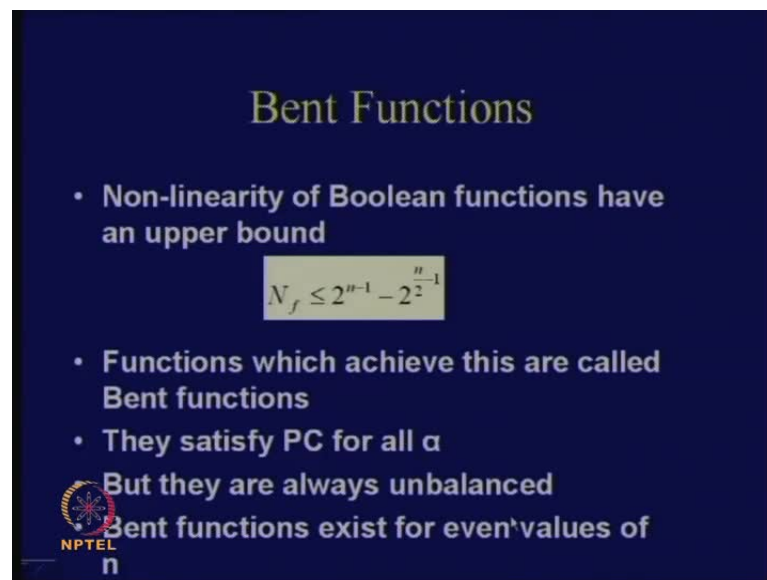
So, this matrix A , essentially will look like this, and now you can check actually that g x and defined is like f x A will actually satisfy the strict avalanche criteria, which means,

if you make this transformation of the input variable say x , using this particular function, or rather using this matrix transformation, then this composed function or transformed function will actually follow the strict avalanche criteria.

So, which means, that if there is a particular Boolean function which does not satisfy the strict avalanche criteria, you can actually make it satisfy the strict avalanche criteria; and not another thing, that what you have done, is that you have just taken x and multiplied that with an invertible matrix.

So, therefore by my previous discussion, the non-linearity function of the $f(x)$ and $f(x) \oplus A$ are actually the same; so, therefore, this particular transformation $f(x) \oplus A$ does not disturb the non-linearity of $f(x)$. So, therefore, without disturbing the non-linearity of $f(x)$ you can actually satisfy the strict avalanche criteria also. Therefore, if you know how to make good non-linear function, which does not satisfy the strict avalanche criteria; after that you can **actually make**, satisfy strict avalanche criteria without disturbing the non-linearity. Do you follow that?

(Refer Slide Time: 43:37)



Bent Functions

- Non-linearity of Boolean functions have an upper bound

$$N_f \leq 2^{n-1} - 2^{\frac{n}{2}-1}$$

- Functions which achieve this are called Bent functions
- They satisfy PC for all α

But they are always unbalanced

Bent functions exist for even values of n

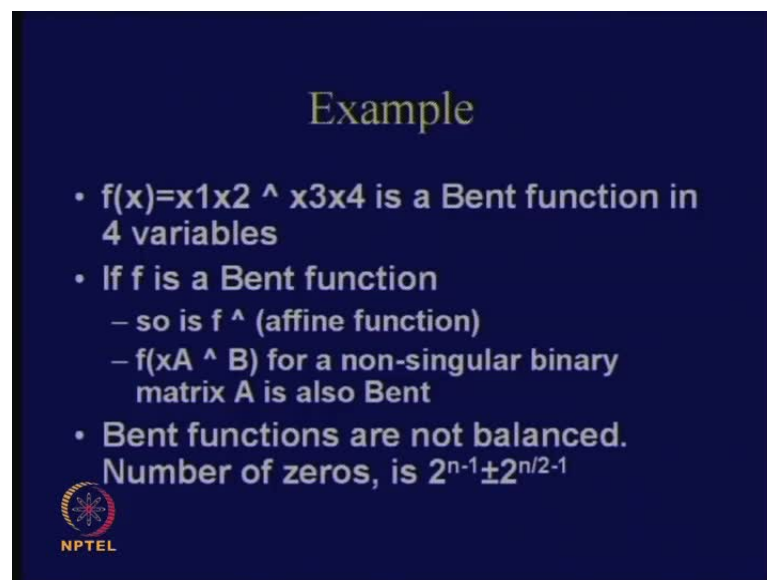
NPTEL

So, therefore these are some things which you can just keep in your mind that for any n variable Boolean function actually, there is an upper bound to what the non-linearity can be. So, we will not prove this, but actually, you can show non-linearity of any n variable Boolean function is actually lesser than $2^{n-1} - 2^{\frac{n}{2}-1}$

1; and this works when n is even. So, you see that n by 2 must be an integer; and these functions are sometimes refers to something which is called Bent function.


So, these are called Bent functions and actually, they can be shown that they satisfy the propagation criteria for all value of α for all orders, but there is a problem with bent functions also. The problem is that, it is actually an unbalanced function; so what does it mean? The number of zeros and ones are not the same, can you say why? So, therefore, this Bent function has got a non-linearity of maximal value. So, the non-linearity of the Bent function is actually equal to $2^{n-1} - 2^{n/2-1}$; for all of that cases, it is lesser than that. So, therefore, the non-linearity of the Bent function is actually equal to $2^{n-1} - 2^{n/2-1}$.

(Refer Slide Time: 45:16)



Example

- $f(x) = x_1 x_2 \oplus x_3 x_4$ is a Bent function in 4 variables
- If f is a Bent function
 - so is $f \oplus$ (affine function)
 - $f(xA \oplus B)$ for a non-singular binary matrix A is also Bent
- Bent functions are not balanced. Number of zeros, is $2^{n-1} \pm 2^{n/2-1}$

 NPTEL

So, now, can we relate to the fact that why it is an unbalanced function? So, we will just give a second to it, I will come back to it. Therefore, this is an example of a Bent function. **You can take like x** , There are four variables x_1, x_2, x_3 , and x_4 , you just take $f(x)$ is equal to $x_1 x_2$ XORed with $x_3 x_4$; so this is an example of a Bent function with four variables. So, if it is a Bent function, what do you expect the non-linearity of this to be equal to? $2^4 - 1 - 2^{4/2 - 1}$. So what is that equal to, how much does it work to? To 6. So, we can check that all four variable Boolean functions, actually they have a non-linearity which is not a bent function, you have a non-linearity which is less than that. So, there are some affine properties like, if you take

an f , which is a Bent function then, so is f XORed with affine function so, if I take this and I XOR that with x_1 XORed x_2 , or something like that, then still the non-linearity does not change. Similarly, you can also see for any non-singular binary matrix A and if you do a similar kind of transformation to your inputs, then it still remains Bent, the Bent is does not changed.

So, now, I again come back to the problem that again why Bent functions are not balanced? So, you can see that, since Bent functions maintain a non-linearity of at least $2^{n-1} - 2^{n-2}$. So, therefore, if I consider the distance from the non-zero function, still then the distance should be $2^{n-1} - 2^{n-2}$ plus I mean plus $2^{n-1} - 2^{n-2}$. So, therefore, the number of zeros in a balanced function is how much? $2^{n-1} - 2^{n-2}$. So, therefore, you see that the number of ones is actually more than $2^{n-1} - 2^{n-2}$.


So, why I have written minus also here? Because, actually, what am considering is that all one functions also. If you maintain that, if you look for both the functions then actually you can ensure that you are not anyway, you are not able to obtain that number; you are not able to maintain the number of zeros or number of ones to equal to $2^{n-1} - 2^{n-2}$. So, therefore, that is not possible. So, Bent functions are not balanced, that is, the idea, although it maintains high amount of non-linearity, but they are not balanced.

(Refer Slide Time: 47:33)

Creating Balanced Non-linear function

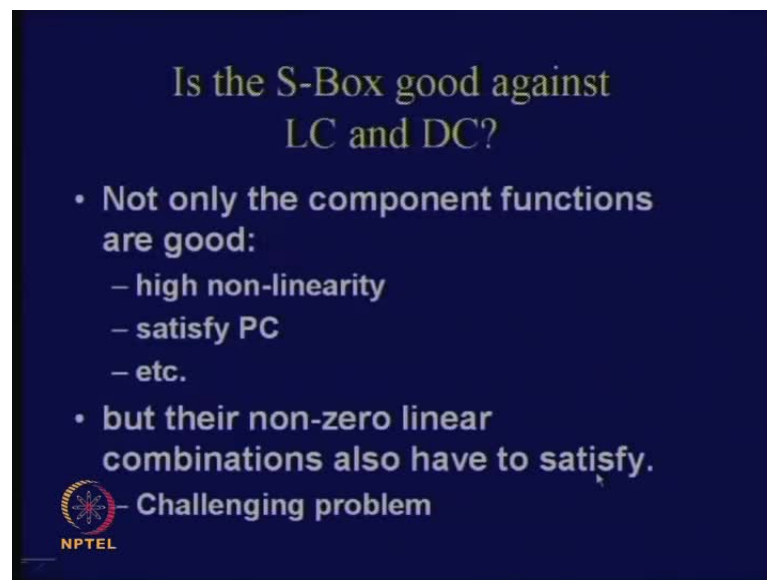
- Take 2^{n-k} , k -variable linear function, where $k > n/2$
- Concatenate the truth-tables
- Thus, we obtain a $n \times k$ mapping which is non-linear
 - $N \geq 2^{n-1} - 2^{k-1}$
- Balanced

Can be made to satisfy SAC.

 NPTEL


So, therefore, this is the idea which I have already told you. So, actually, what you can do is that you can actually create balanced non-linear functions. So, therefore, I am not really going to this, what it just says is that you can take some linear functions and you start concatenating them.

(Refer Slide Time: 48:21)



Is the S-Box good against
LC and DC?

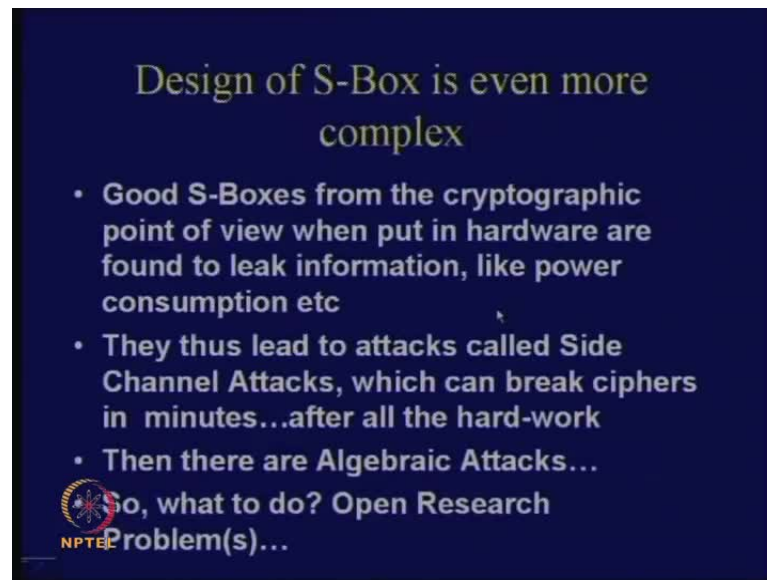
- Not only the component functions are good:
 - high non-linearity
 - satisfy PC
 - etc.
- but their non-zero linear combinations also have to satisfy.

 – Challenging problem

NPTEL

So, you can just see this and you will obtain that if you take so, 2^n k -variable linear function, where k is greater than n by 2, and start concatenating linear truth-tables then you will obtain a n cross k -mapping which is non-linear. Therefore, you can just take it as an extra and try to work out. I mean, you can just also prove that the non-linearity of this will be actually greater than $2^n - 1 - 2^{k-1}$, it is also be balanced and it can be made to satisfy strict avalanche criteria also. So, therefore, the question is the S-Box which you design is good against linear crypt analysis and differential crypt analysis. So, not only the component is good, but they have to satisfy high non-linearity like satisfy propagation, etcetera. But also the problem is that their non-zero linear combinations also should satisfy these properties. So, therefore, it is a quite challenging problem to design a good S-Box.


(Refer Slide Time: 48:47)



Design of S-Box is even more complex

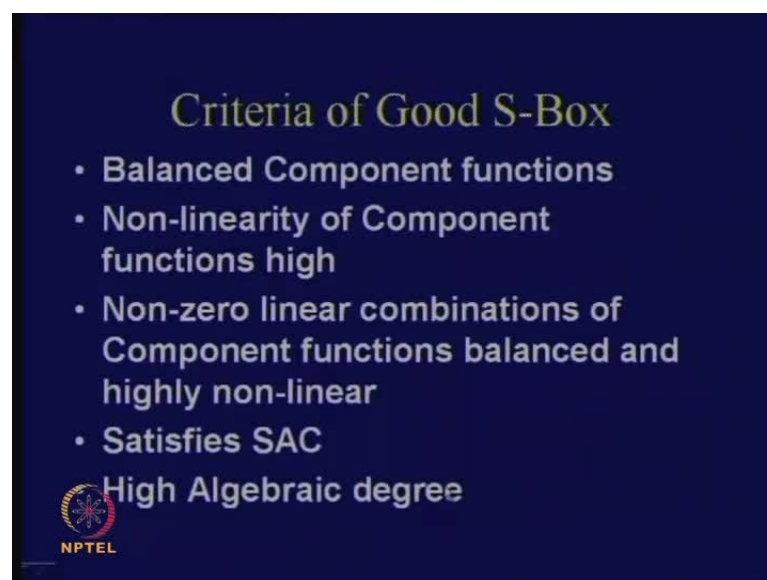
- Good S-Boxes from the cryptographic point of view when put in hardware are found to leak information, like power consumption etc
- They thus lead to attacks called Side Channel Attacks, which can break ciphers in minutes...after all the hard-work
- Then there are Algebraic Attacks...

So, what to do? Open Research Problem(s)...




So, I will just conclude with this that S-Box designing is quite complex. Although you have to satisfy so many things and then also finally what you can ensure, you can end up, is that this leaks informations in the form of some side channels and things like that. So, therefore, it can happen. Although you do so much mathematics, you can actually break these ciphers in minutes even after all the hard work.

(Refer Slide Time: 49:15)



Criteria of Good S-Box

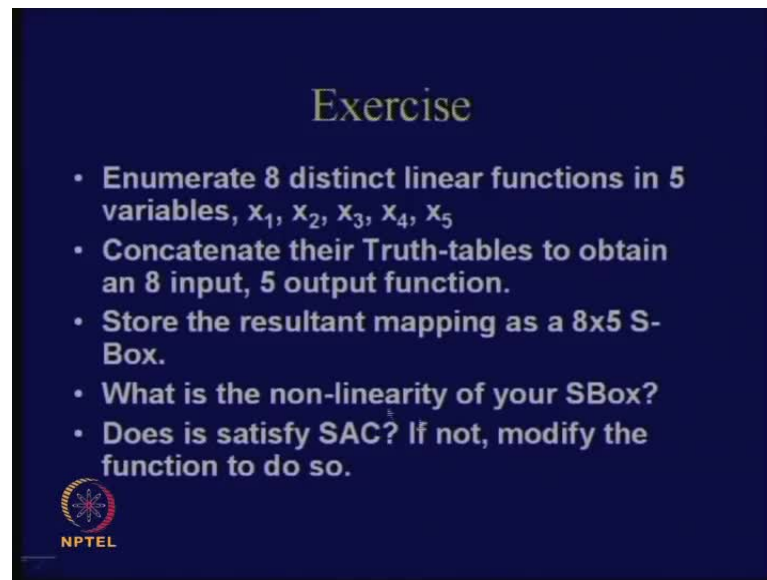
- Balanced Component functions
- Non-linearity of Component functions high
- Non-zero linear combinations of Component functions balanced and highly non-linear
- Satisfies SAC
- High Algebraic degree



So, then the other classes of attacks or algebraic attacks, which are not really touched into. So, actually it is quite hard to design a good S-Box, so therefore, I just conclude


with this, these are the properties which would satisfy balanceness, non-linearity and also strict avalanche criteria; and also high algebraic degree, that is, number of and products in your algebraic normal form should also be high.

(Refer Slide Time: 49:31)



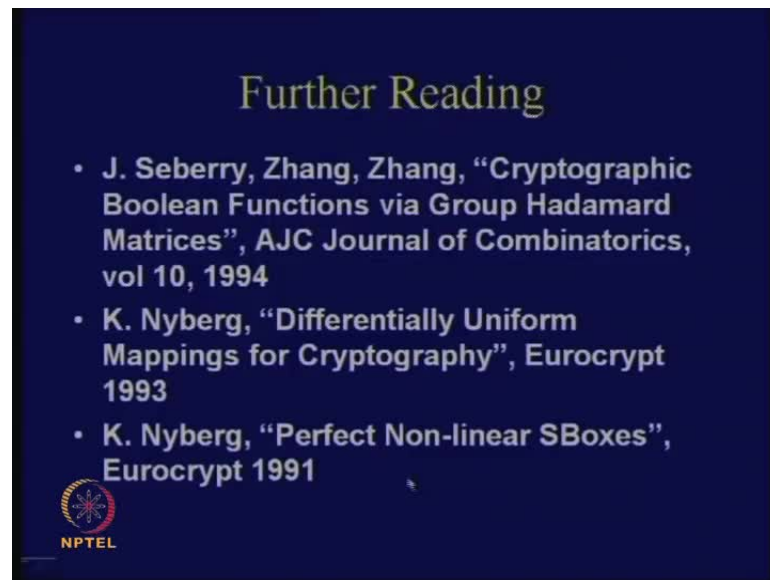
Exercise

- Enumerate 8 distinct linear functions in 5 variables, x_1, x_2, x_3, x_4, x_5
- Concatenate their Truth-tables to obtain an 8 input, 5 output function.
- Store the resultant mapping as a 8x5 S-Box.
- What is the non-linearity of your SBox?
- Does it satisfy SAC? If not, modify the function to do so.

 NPTEL


So, you can take this an exercise, that is, you can enumerate 8 distinct linear functions in 5 variables, x_1, x_2, x_3, x_4 , and x_5 . So, you can just choose 8 distinct linear functions in 5 variables. So, you just take 5 variables and form 8 possible linear functions and then concatenate their truth tables to obtain an 8 input, 5 output function. So, you can see just that why it works? So, you can just take these functions, linear functions and you start concatenating their truth tables. So then, you store the resultant mapping as an 8 cross 5 S-Box like, what we have seen for DES for example and other S-Boxes. So, question is that what is non-linearity of your S-Box and does it satisfy strict avalanche criteria? If not, then modify the function to do so. So, you can take this an exercise along with your previous submission, which is still pending like that DES S-Box. You also submit this as an exercise.

(Refer Slide Time: 50:31)

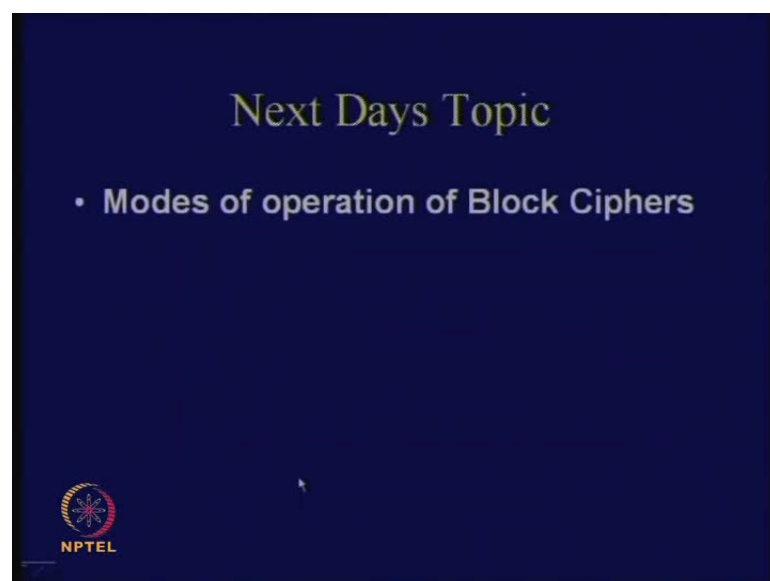


Further Reading

- J. Seberry, Zhang, Zhang, "Cryptographic Boolean Functions via Group Hadamard Matrices", *AJC Journal of Combinatorics*, vol 10, 1994
- K. Nyberg, "Differentially Uniform Mappings for Cryptography", *Eurocrypt 1993*
- K. Nyberg, "Perfect Non-linear SBoxes", *Eurocrypt 1991*


 NPTEL

(Refer Slide Time: 50:39)



Next Days Topic

- Modes of operation of Block Ciphers

 NPTEL

So, these are the some of the references that I followed. Seberry's paper is quite interesting, you read that; and then, there is also Nyberg's paper which you can read that. So, next day, we will talk about modes of operation of Block Ciphers.