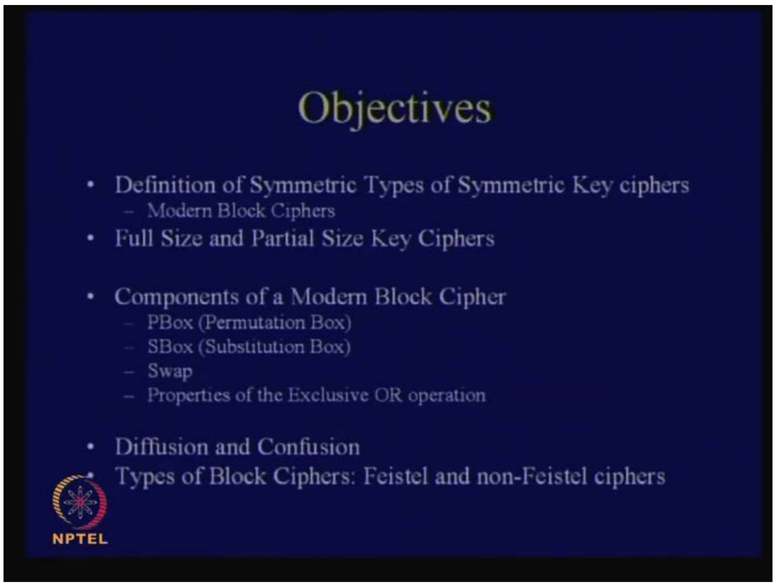
Cryptography and Network Security

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

Module No. #01
Lecture No. #10
Symmetric Key Ciphers

(Refer Slide Time: 00:25)



Objectives

- Definition of Symmetric Types of Symmetric Key ciphers
  - Modern Block Ciphers
- Full Size and Partial Size Key Ciphers
- Components of a Modern Block Cipher
  - PBox (Permutation Box)
  - SBox (Substitution Box)
  - Swap
  - Properties of the Exclusive OR operation
- Diffusion and Confusion
- Types of Block Ciphers: Feistel and non-Feistel ciphers

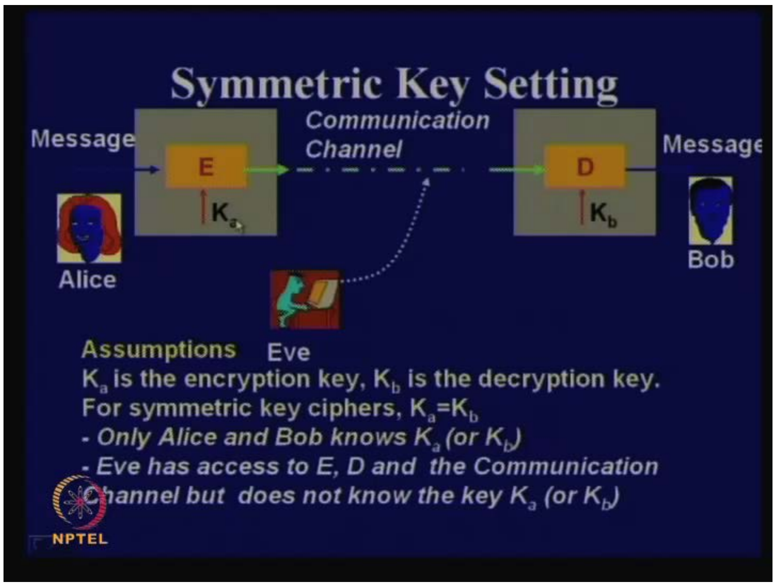
Welcome to today's class on Symmetric Key ciphers. So, the objectives of today's class, as we follow, that first we will start with definition of, symmetric types of, symmetric key ciphers. So, essentially, what I mean to say is that, we have two, as we have discussed, that there are two classes of ciphers, one is called symmetric ciphers and other one is called asymmetric ciphers.

So, we will try to understand, what is the symmetricity? What is the implication of symmetricity in that kind of ciphers? And then, talk about something called, full size and partial size key ciphers.

So, we will follow that up with discussions on certain components of modern block ciphers; so they are as follows, like P-box, S-box, which we called as permutation box and substitution box, and then discuss about another component, a very simple component called swap, but very useful component.

And then discuss about certain properties of the Exclusive OR function, because that is very, sort of, central to the construction of present block ciphers. And then, discuss about another concept, which Shannon gave in his paper, is called diffusion and confusion, and try to understand, how modern block ciphers achieved them. And then, conclude with discussion on two types of block ciphers, they are called Feistel block ciphers and non-Feistel block ciphers.

(Refer Slide Time: 01:39)



Symmetric Key Setting

Assumptions

K_a is the encryption key, K_b is the decryption key.

For symmetric key ciphers, K_a=K_b

- *Only Alice and Bob knows K_a (or K_b)*
- *Eve has access to E, D and the Communication Channel but does not know the key K_a (or K_b)*

So, we straightaway start with the symmetric key setting; this is a sort of a diagram, which shows how a symmetric key setting looks like. So, you will see that there are two players in this game, one is called Alice and the other one is called Bob. So, Alice and Bob, essentially, they want to communicate certain secret data over a communication channel.

So, therefore, you have, there are two parts of, I mean, two ends of, one is the sender and the other one is the receiver. So, we have an encrypting algorithm called E and we have decrypting algorithm called D. So, the encrypting algorithm uses a key, which is denoted by K a and the decrypting algorithm uses a key which is denoted by K b. So, there is an evesdropper whose name is Eve, who tries to sneak into the communication channel and his objective is to understand what is the message.

So, therefore, Alice and Bob are trying to use cryptography or encryption to stop Eve from knowing what he is supposed not to know, essentially. So, therefore, you will find that the, what are the assumptions? The assumptions are that K a is the encryption key and K b is the decryption key.

So, therefore, the encryption key is denoted by K a, and decryption key is denoted by K b, and in the symmetric key setting the assumption is that K a and K b are the same; therefore, the encrypting key and the decrypting key are the same things as in there.

So, therefore, for symmetric key ciphers, K a is equal to K b as opposed to the asymmetric key setting, that there is a slight distinction, which you will discuss in the following classes. So, our assumption is that only Alice and Bob know the values of K a or K b.

So, therefore, Eve does not have the key material, but Eve knows the encrypting algorithm and the decrypting algorithm, but he does not know, what is the value of the key? That is only what Eve does not know and the objective of Eve too, is to understand the corresponding value of the message. So, that is essentially setting of the symmetric key ciphers.

But the setting of symmetric key ciphers and as in this particular setting, where your encryption algorithm or decryption algorithm are in public domain, where only the secret, that is, the key is not known to the attacker or the adversary, you have to provide a guarantee of security. And as we discussed in our previous classes, that security against an unbound adversary is quite difficult to achieve, but what we are striving to achieve in our classes is what we know as, computational security.

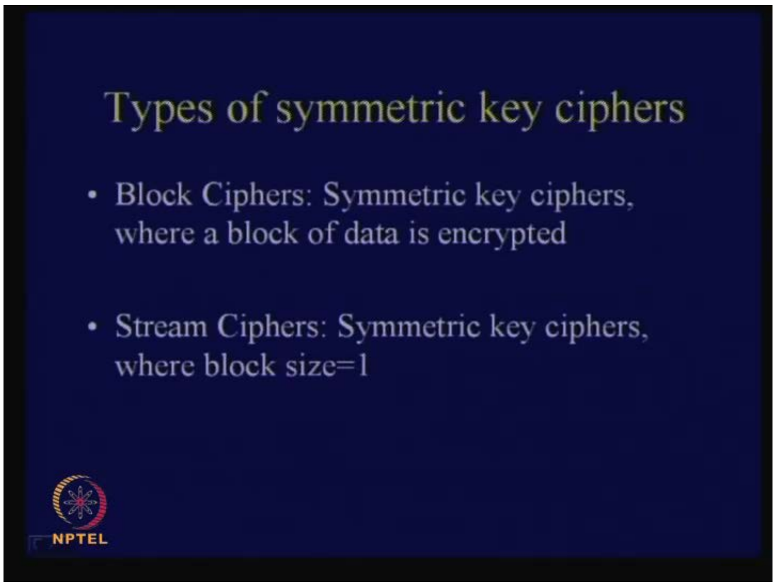
So, Eve has access to E, D and the communication channel, but does not know the key K a or K b, and how do I provide a guarantee of security to Alice and Bob? So, that is we will try to see under these setting. So, the obvious problem here, I mean, although we will not discuss right now, right away, right now, so is, that how does Alice and Bob share this piece of information that is the key. So, that is called the key distribution problem.

So, we will come to that later on, but essentially we are assuming at this point, that there is a secret channel through which Alice and Bob has established the encrypting and the decrypting key.

But that, that, the point is that to be known, I mean, to be noted here is that, here what we are encrypting is large, large traffic of information and what we are exchanging is a comparatively small bit of data.

So, therefore, we are, we are assuming, rather is, it is a practical assumption to make that although we have a secret channel through which we are establishing the key, but we are using that for a very small quantity of data because we value that channel; that channel is quite, supposedly quite costly.

(Refer Slide Time: 05:21)

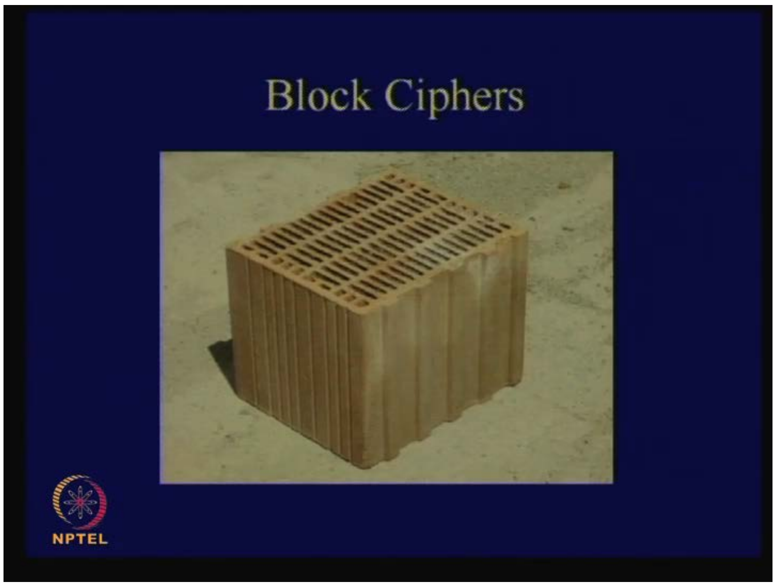


Types of symmetric key ciphers

- Block Ciphers: Symmetric key ciphers, where a block of data is encrypted
- Stream Ciphers: Symmetric key ciphers, where block size=1

So, the types of symmetric key ciphers that essentially we will encounter are two-fold, is called block ciphers and stream ciphers. So, block ciphers, as the name suggests is, when we are encrypting a block of data. And in symmetric key ciphers, if we just replace or rather substitute, that if we assume that the block size is 1, that is, if you process 1 bit at a time, it is known as the stream ciphers.

(Refer Slide Time: 05:52)

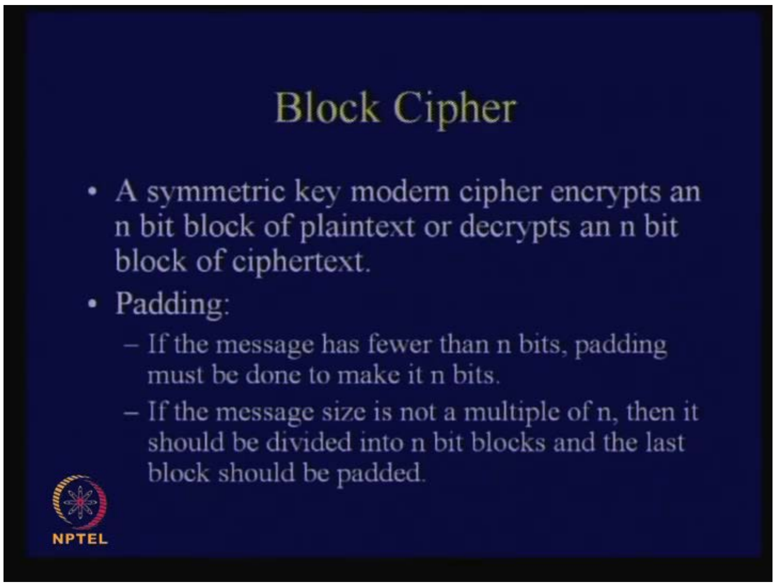


Block Ciphers

So, it is supposedly, as streams of data are coming in and they are getting encrypted. So, we will start with block ciphers and that is the objective of today's class. Therefore, as you see, that essentially, we are handling tons of data, we are handling blocks of data; so therefore, let us try to understand, how block ciphers work? So, this is the block box and we will be trying to go into the block box.

As we told, that we are striving to achieve computational security, so the thumb rule is 2 power 80 is, as in today's computational power, we say that 2 power of 80 is the reasonable amount of security; therefore, anything beyond 80 bits would be quite o.k.

(Refer Slide Time: 06:34)



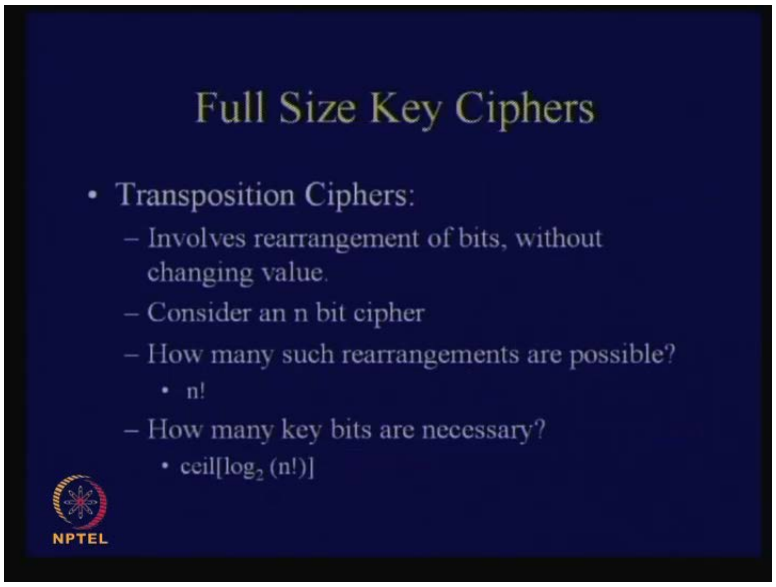
Block Cipher

- A symmetric key modern cipher encrypts an n bit block of plaintext or decrypts an n bit block of ciphertext.
- Padding:
  - If the message has fewer than n bits, padding must be done to make it n bits.
  - If the message size is not a multiple of n, then it should be divided into n bit blocks and the last block should be padded.

So, this, the answer to your question is actually time dependent, what is today correct may not be correct ten years later. So, therefore, what is a block cipher? A block cipher is a symmetric key modern cipher, which encrypts an n-bit block of plain text or decrypts an n-bit block of cipher text. Therefore, it works on n-bits at a time.

So, obviously, we, we understand that if the message is not a multiple of n bits, then what do we do? If it is less, the size is less than n bits, then we will pad it and pad it by says 0s and make it n bits, and if it is greater than n bits, but it is not a multiple of n, then what we do is that we divide them into n bit blocks, and the last block, which is left is obviously, lesser than n bits; in that case, we also pad the last block and make it equal to n bits. Therefore, we perform padding if it is not a multiple of n bits. So, we can therefore, for all our discussions, we can assume that the message is actually a multiple of n bits; it does not really make any difference to what we will discuss in this class.

(Refer Slide Time: 07:34)



Full Size Key Ciphers

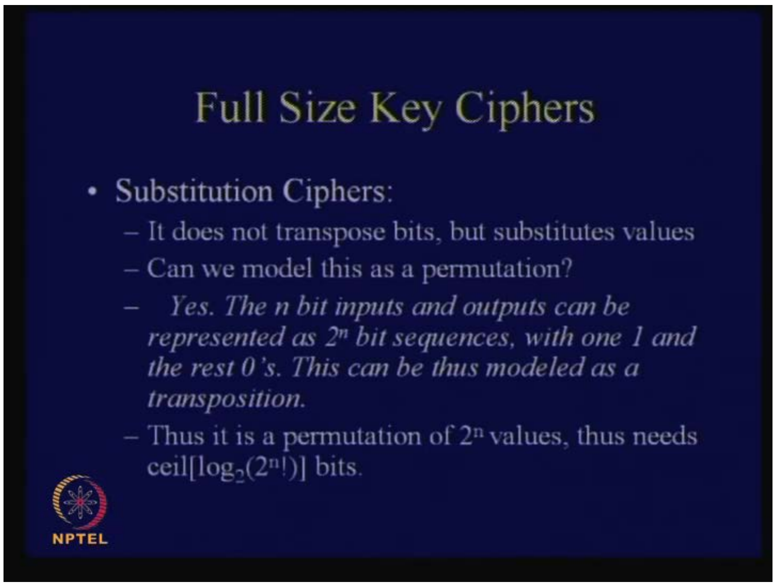
- Transposition Ciphers:
  - Involves rearrangement of bits, without changing value.
  - Consider an n bit cipher
  - How many such rearrangements are possible?
    - n!
  - How many key bits are necessary?
    - ceil[$\log_2$ (n!)]

So, first we will start with the concept of full size key ciphers. Therefore, let us, we have seen two kinds of ciphers - one, former classical discussions where you have seen, that there are, we have got the class of ciphers called transposition ciphers and there is another class, which is called substitution ciphers.

So, what were transposition ciphers? What was it involved with? It was involve with essentially, the rearrangement of the bits. Therefore, consider that we make an n bit cipher or n bit block cipher using the transposition function only, so then, what does it encompass? It encompass with rearrangement of the bits.

So, how many such rearrangements are possible? Obviously, n factorial. So, therefore, if I just try to encode each particular rearrangement by one value of the key, then how many values of the keys do we require? We require log n factorial base 2, such, so many number of bits and we will for example, ceil log.

(Refer Slide Time: 08:41)



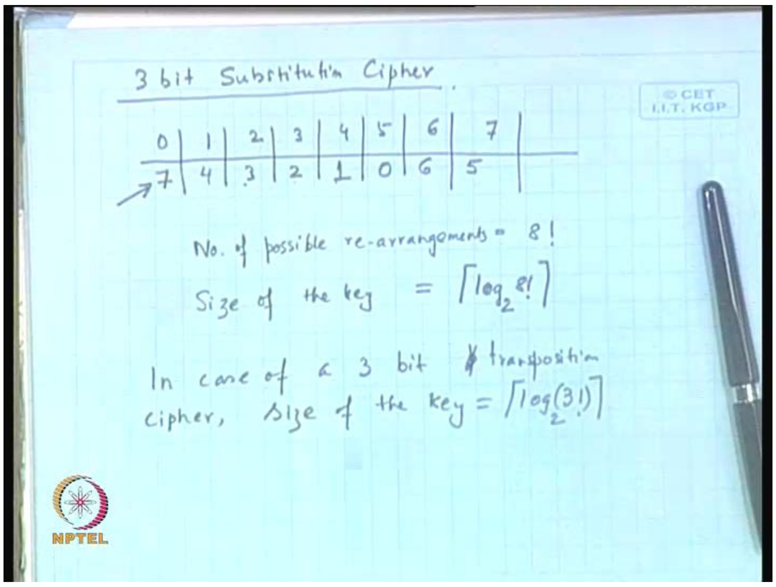
Full Size Key Ciphers

- Substitution Ciphers:
  - It does not transpose bits, but substitutes values
  - Can we model this as a permutation?
  - *Yes. The n bit inputs and outputs can be represented as 2^n bit sequences, with one 1 and the rest 0's. This can be thus modeled as a transposition.*
  - Thus it is a permutation of 2^n values, thus needs ceil[$\log_2(2^n!)$] bits.

So, therefore, it is the ceiling of log n factorial base 2; so many amount of keys are required if we want to encode each and every possible term position. So, as opposed to that, suppose this was a substitution cipher, then what would have been the size of the key? So, consider for example, that in a case of, in a substitution cipher it does not transpose the bits, but it was essentially substituting the values.

So, therefore, can we model this as a permutation? Can we model a substitution as a permutation? So, what was important to, yes we can do that right, because the object, because the point that we observe is that what, when we, if we do substitution, the mapping is still one-to-one.

(Refer Slide Time: 09:27)



3 bit Substitution Cipher

0	1	2	3	4	5	6	7
7	4	3	2	1	0	6	5

No. of possible re-arrangements = $8!$

Size of the key $= \lceil \log_2 8! \rceil$

In case of a 3 bit ~~p~~ transposition cipher, size of the key $= \lceil \log_2 (3!) \rceil$

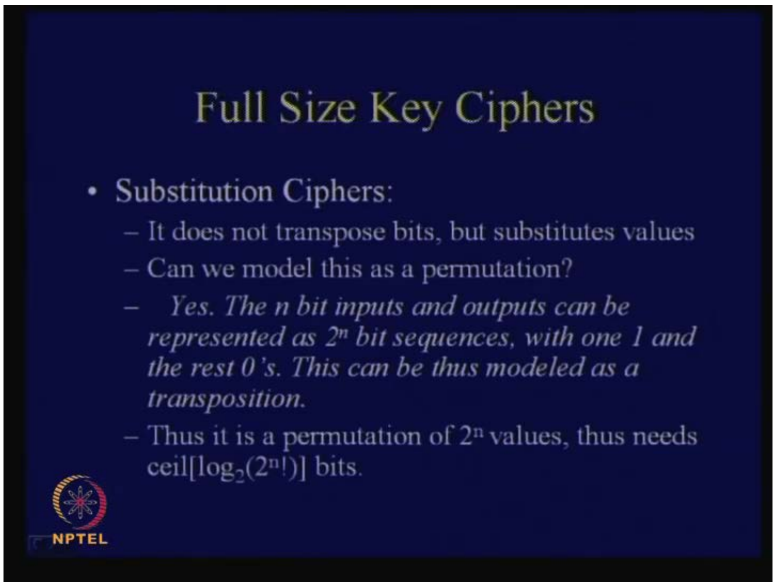
So, therefore, in a substitution cipher, also if you, for example consider a 3-bit substitution ciphers, then we can actually encode this in the form of a table. Therefore, we can say that a 3-bit substitution cipher, all the inputs if we denote in the form of a table, would be like 0, 1, 2, 3, 4, 5, 6 and 7.

So, therefore, all, for all these possible, so therefore, for all these possible inputs, we essentially allocate or substitute them by some other values. So, therefore, if I consider a 3-bit substitution cipher, so essentially they, for example, if I just substitute 0 by say 7, 1 I would have substituted by some other value, so therefore, it could be say, 4 or something like that. Therefore, just imagine, therefore, if I just fill up this with some, some, such essential values, so that is a, therefore for example, we can fill up this with say 3, we can fill up this we say 2, we can fill up this with 1, 0, 6 and 5.

So, what is the observation value we make out here? Is that the output of this table is nothing but a rearrangement of these 8 terms. So, how many are possible, rearrangements are possible? It is 8 factorial. So, therefore, number of possible rearrangements, number of possible rearrangements is equal to 8 factorial.

So, therefore, how many, so if I represent them by, by keys, what would have been the size of the key? The size of the key would have been log 8 factorial base 2 and ceiling of that. So, therefore, you note that in our previous case when we had a, when we are discussing about a permutation cipher and a 3-bit permutation cipher, so in case of, in case of a 3-bit transposition cipher error, transposition cipher, what was the size of the key? It was 3 factorial and log base 2 and a ceiling of that.

(Refer Slide Time: 11:50)

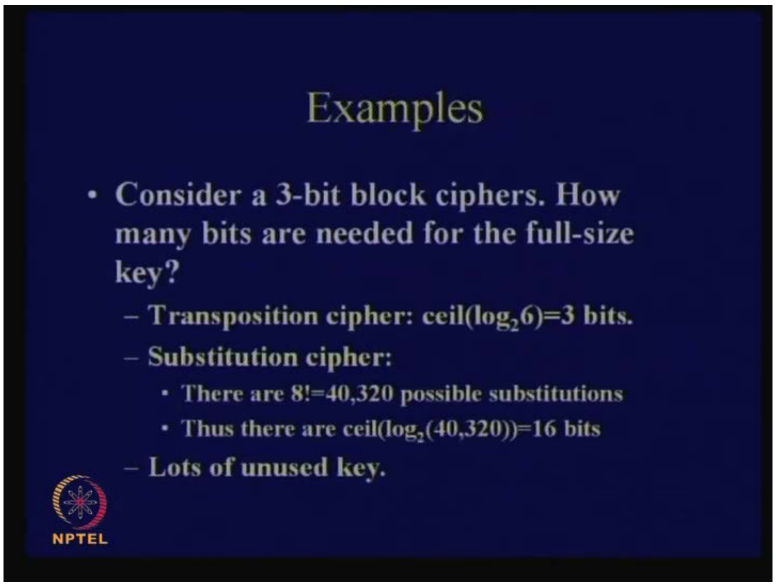


Full Size Key Ciphers

- Substitution Ciphers:
  - It does not transpose bits, but substitutes values
  - Can we model this as a permutation?
  - *Yes. The n bit inputs and outputs can be represented as 2^n bit sequences, with one 1 and the rest 0's. This can be thus modeled as a transposition.*
  - Thus it is a permutation of 2^n values, thus needs $\text{ceil}[\log_2(2^n!)]$ bits.

So, therefore, coming back to our generalized discussion on n-bit ciphers. So, if it is a permutation, so it is an essentially, if you are considering n-bit substitution ciphers, then it is a permutation of 2 to the power of n values. So, therefore, how many size, so what is the size of the key? In that case, it is logarithm of 2 to the power of n factorial base 2 and we take a ceiling of that. So, I guess, this is clear.

(Refer Slide Time: 12:20)



Examples

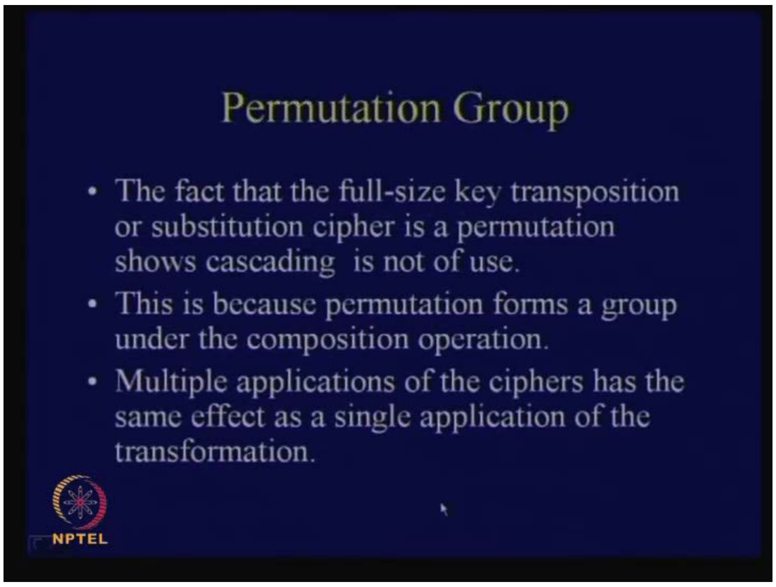
- **Consider a 3-bit block ciphers. How many bits are needed for the full-size key?**
  - **Transposition cipher: $\text{ceil}(\log_2 6)=3$ bits.**
  - **Substitution cipher:**
    - **There are $8!=40{,}320$ possible substitutions**
    - **Thus there are $\text{ceil}(\log_2(40{,}320))=16$ bits**
  - **Lots of unused key.**

So, therefore, this is what we discussed just now. So, we can just see that if I consider, I mean, the value of 8 factorial works to around 40,320. Therefore, there are so many possible substitution values, so if we take ceil of that, then the size of the key is around 16.

So, how many, so if we, if we can see, that 2 to the power of 16 is actually a much larger term than 40,320, so that means, that there are lots of unused keys. Similarly, for your transposition cipher also, it works to 3-bits, so how many possible values mappings are therefore, possible? 8 values, but out of them, you are just using 6 values. So, you see that in such kind of ciphers, when we are using, considering full size ciphers, full size key ciphers, then there is lot of unused values are in a key.

So, this is one observation that we make and also the size of the key is also quite large. So, for example, so we will come to that, why, I mean, about that point. Therefore, next thing that we observe is that… So, therefore, so we discuss about something, which is called a permutation group.

(Refer Slide Time: 13:22)



Permutation Group

- The fact that the full-size key transposition or substitution cipher is a permutation shows cascading is not of use.
- This is because permutation forms a group under the composition operation.
- Multiple applications of the ciphers has the same effect as a single application of the transformation.

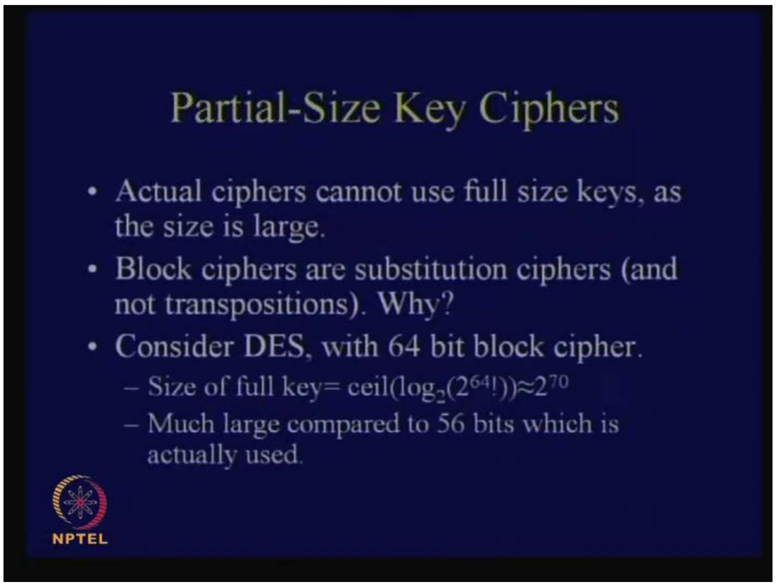
So, the fact, that the full size key's transposition or substitution cipher is essentially, a permutation, so therefore, transposition cipher is a permutation of n-bits and a substitution cipher of n-bits is a permutation of 2 to the power of n values.

So, therefore, you can essentially denote a transposition cipher or a substitution cipher by a permutation. So, therefore, if I cascade such kind of operations, if I just repeat the same operations again and again, then you still get another permutation. So, therefore you do not get some other kind of transformation, which means what? The permutations essentially, form something, which is like a group under the composition operation; so, under the composition operation your permutation forms a group.

So, this is because permutation forms a group under the composition operation, we can, we can conclude, the multiple applications of the cipher has got the same effect as the single application of the transformation.

So, therefore, it really does not help in getting something that we strive for in the last day's class, the concept of rounds. So, we want to increase our security, therefore this essentially, really, does not help that much. So, a full size key cipher, we found, that has got certain disadvantages. So, one of the disadvantages is that the, which we found, that there are lot of unused keys; another thing was that, if I, I mean, cascading of such kind of ciphers really does not help; another point, that we observe is that, I mean, that we will be observing is that the size of the key is actually, quite large.

(Refer Slide Time: 15:16)



Partial-Size Key Ciphers

- Actual ciphers cannot use full size keys, as the size is large.
- Block ciphers are substitution ciphers (and not transpositions). Why?
- Consider DES, with 64 bit block cipher.
  - Size of full key= ceil($\log_2(2^{64}!)$)$\approx 2^{70}$
  - Much large compared to 56 bits which is actually used.

So, therefore, we essentially, introduce the concept of partial-size keys ciphers and that is essentially, used in all modern day cryptographic systems, symmetric cryptographic systems. So, actual ciphers cannot use full size keys, why? And because of size is quite large, therefore, let us see for example, in the case of DES, so DES has got typically 64-bit, is typically a 64-bit block cipher.

So, therefore, if we, if we had used full size key, then the value of the, or the size of the key would have been around logarithm base 2 of 2 to the power of 64 factorial because it is a substitution cipher. Can you tell me that why block ciphers or substitution ciphers are not transposition ciphers? Modern day block ciphers are substitution ciphers and not transposition ciphers, why?

Something, can we relate this to what we have learnt, like can we relate this to information of the plaintext. So, that is some information, which has been leaked in transposition ciphers, can you guess what?

Typically all types of the block are replaced by some other block.

There is a substitution cipher so…

It is a rearrangement or a….

It is a rearrangement, but can we conclude that can we get some information of the plaintext, if we had only transposition ciphers?

(( ))

Yeah, that happens in substitution ciphers also.

(())

Yeah, but can you quantize the some information is getting leaked?

(( ))

Frequency of the letters in the English language are same…

Something much more simple. Consider that your plain text is made of only 1s and 0s and just consider a transposition cipher, so what is the information that is getting leaked?

Number of 0s and number of 1s.

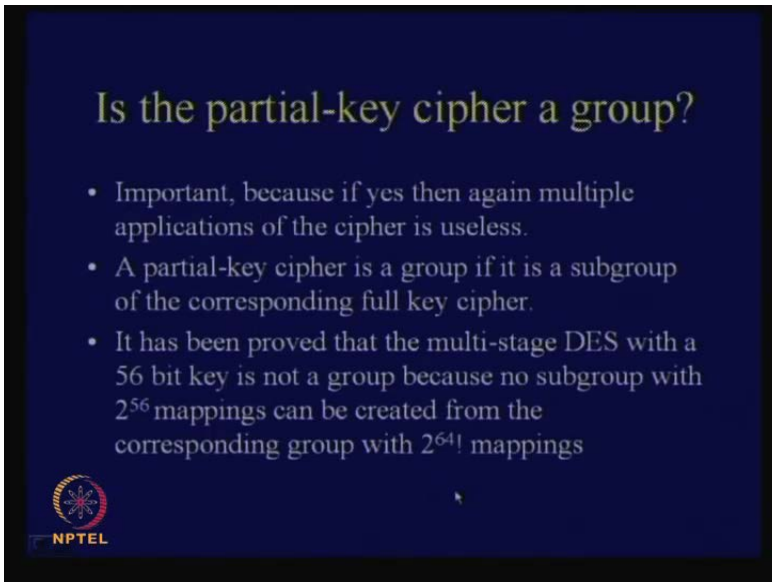
Number of 0s and number of 1s. Therefore, you see that the humming rate of your inputs gets leaked, so humming rate, you know, the number of 1s is essentially, so you see, that have got a very straight forward information, which gets leaked in transposition ciphers.

But it is not the same case in case of substitution ciphers. It is a completely different mapping; it has got nothing to do with the number of 1s in an input and number of 0s in the input. So, therefore, you see that modern day block ciphers are essentially substitution ciphers and not transposition ciphers.

So, therefore, DES which is a 64-bit block ciphers is also a 64-bit substitution block ciphers. Therefore, the number of or the size of the full key would have been logarithm of 2 to the power of 64 factorial because it is a permutation in 2 to the power of 64 values. So, therefore, there are 2 to the power of 64 factorial possible arrangements. So, if I represent them by keys, the size of the key, you can find out would have worked out to 2 the power of 70. So, that is the very large number, so we cannot really handle a, such a large key, so we require a much shorter key.

So, therefore, you will find that in our standards, we find that we use 56-bits key, which is actually much smaller; so, we use 56-bit keys compared to 2 to the power of 70. So, you see, that there is a large amount of keys, which is being required for a, for realizing a full size key ciphers. Therefore a, so therefore, a partial size key cipher is also practical, much more practical.

(Refer Slide Time: 18:51)



Is the partial-key cipher a group?

- Important, because if yes then again multiple applications of the cipher is useless.
- A partial-key cipher is a group if it is a subgroup of the corresponding full key cipher.
- It has been proved that the multi-stage DES with a 56 bit key is not a group because no subgroup with 2^{56} mappings can be created from the corresponding group with $2^{64}!$ mappings

So, so, so, as we have discussed, that our full size key cipher was a permutation group, so which will reflect upon this question that, is the partial key cipher also a group? So, it is important because if yes, then again, multiple applications of the cipher are useless and we want to apply them the same operation again and again. Something to, like rounds, we keep on applying the same operation again and again.

So, what I am trying to say to you is that if I apply DES for example, for two times it is still DES. So, can I, I mean, suppose I take DES and I apply that by using two values of keys, I take k 1 and I take k 2 and I apply them. Can I represent that by DES with a third value of the key? You understand, what I am saying?

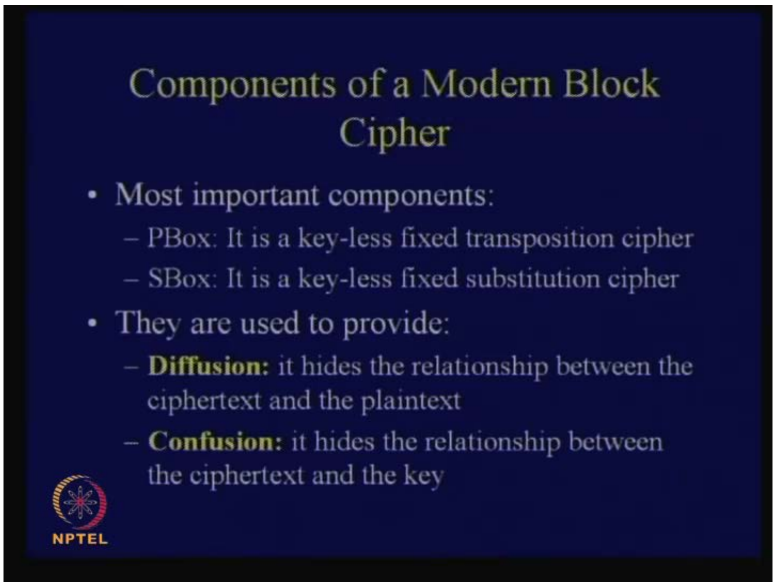
So, that is, if that would have been the case, then DES also would have formed a group. So, but the point is that therefore, a partial key size cipher is a group, if it is actually a sub-group of the corresponding full key cipher. So, therefore, you can understand that a sub-group means essentially it will, if you have got n number of values in your group, then you, in your sub-group we have got n values, where n is a subset of n and it still forms a group under the composition operation, under, under whatever operations, it could be plus, so in those operations in which the group was defined, because that is a definition of the sub-group.

So, it has been proved that that the multistage DES with a 56-bit key is not a group because no sub-group with 2 power of 56 mappings there, are possible. How many possible mappings with 56 bit key? There are 2 power of 56 mappings. So, no sub-groups with 2 power of 56 mappings exist from the corresponding group, which has actually got 2 power of 64 factorial mappings. Therefore, the original mappings has, which forms the group, has got 2 power of 64 factorial mappings.

So, out of them, these 2 power of 56 mappings does not form a sub-group. So, that proves that multiple applications of DES essentially enhances security, that means, it is the key, is actually not 1 value of the key, but actually it is a 2 values of the key, so that is, it was not only you cannot encode them by another 56-bit value.

So, you can actually, in order to encode that you require a 112-bit key, 56 into 2, so that means, your security is actually increasing. So, therefore, that is another advantage of partial size keys ciphers.

(Refer Slide Time: 21:22)



Components of a Modern Block Cipher

- Most important components:
  - PBox: It is a key-less fixed transposition cipher
  - SBox: It is a key-less fixed substitution cipher
- They are used to provide:
  - **Diffusion:** it hides the relationship between the ciphertext and the plaintext
  - **Confusion:** it hides the relationship between the ciphertext and the key

So, then we will come to certain discussions about components of a modern block ciphers. This has been proved for DES, so therefore, we have to be careful that it has to be, it holds or not. So, there is actually a paper, which says, does DES form a group? We will discuss about that.

So, so, therefore, and we, I mean, so therefore, initially when DES was evolving, at that time, the construction were very, sort of, nascent, so, they were just discussing, I mean, developing. But now that we know lot of construction methodologies through which we can actually make block ciphers and know ahead of times, that it does not form a group because we know how to maintain this. But during those times actually, when DES was made at those times, actually the techniques through which DES was made was really not made by, on public domain. Those were the days when ciphers were classified, so they were kept in details of books only, but now actually, we use it for non-needed application; also, it is more of an academic study that we do.

So, therefore, the other components are, we will be discussing is the components of a modern block cipher. So, the most important components are P-boxes and S-boxes. So, what is a P-box? It is a key-less fixed transposition cipher and an S-box is a key-less fixed substitution cipher. So, you see that the same concepts exist in today's ciphers also and they are used to provide something, which we called as diffusion. So, what is diffusion? The diffusion is a property, which hides the relationship between the ciphertext and the plaintext to the attacker and confusion is the property, which hides the relationship between the ciphertext and the value of the key.

So, we will try to understand, how diffusion and confusion are achieved in today’s ciphers? So, you see that for example, in a transposition cipher there was an absence of diffusion, why? Because we were able to relate the hamming weight of the ciphertext with the hamming weight of the plaintext. Similar to that, we essentially, we will try to hide the relationship between the ciphertext and the plaintext and try to make the mapping as random as possible.

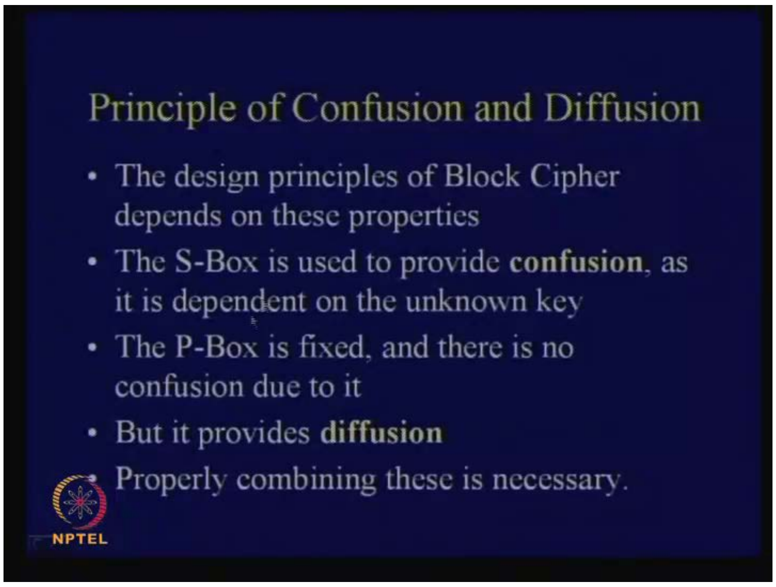
So, the central idea behind any attack is a distinguisher form, a random mapping. If we can distinguish a mapping from a random mapping, then because, actually, I mean, when we are saying that there is an algorithm to make a cipher, it automatically implies it is not random. So, therefore, it is called something, which is related, called pseudorandom.

Therefore, any attacker would try to find out a property, which distinguishes the corresponding mapping with the random mapping. So, if you can find out a distinguisher that is equivalent, you can, it has been shown historically, that you can actually transform a distinguisher into an attacker.

The first objective is to find a property, find, which essentially proves that the given cryptosystem is not random and if you are able to find out the distinguisher, then you can convert the distinguisher into a real life attack. So, therefore, finding out property is essentially alarming.

So, therefore, the objective of diffusion and confusion is essentially to hide those properties. So, it tries to hide the relationship between the ciphertext and the plaintext and a confusion tries to hide the relationship between the ciphertext and the key.

(Refer Slide Time: 24:54)

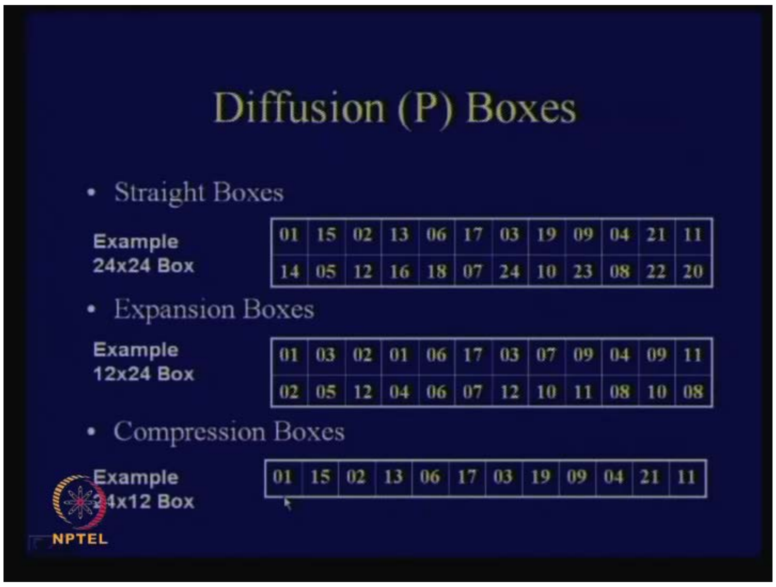


Principle of Confusion and Diffusion

- The design principles of Block Cipher depends on these properties
- The S-Box is used to provide **confusion**, as it is dependent on the unknown key
- The P-Box is fixed, and there is no confusion due to it
- But it provides **diffusion**
- Properly combining these is necessary.

So, the principle of confusion and diffusion, we will see, I mean, the design principles of block ciphers depends on these properties. The S-box is generally used to produce, to provide confusion as it is dependent on the unknown value of the key, we will see how? And the P box is fixed, there is no confusion due to it, but it does provide diffusion. So, properly combining these is actually very much necessary for obtaining security in the block cipher.

(Refer Slide Time: 25:20)



Diffusion (P) Boxes

- Straight Boxes

Example
24x24 Box

01	15	02	13	06	17	03	19	09	04	21	11
14	05	12	16	18	07	24	10	23	08	22	20

- Expansion Boxes

Example
12x24 Box

01	03	02	01	06	17	03	07	09	04	09	11
02	05	12	04	06	07	12	10	11	08	10	08

- Compression Boxes

Example
24x12 Box

01	15	02	13	06	17	03	19	09	04	21	11

So, let us see one example of a diffusion box, so this is a straight box. Therefore, the diffusion boxes are of generally of three types, it is: straight box, expansion box and compression box.

So, I have referred this P in bracket because I was little bit hesitant in writing a P-box or a permutation box implying an expansion and compression because they are not truly permutations by definition, but it generally used in literature. Therefore, we just keep in mind that a diffusion box or a P-box essentially, are of three types: one is straight box, other one is expansion box and other one is compression box. So, as the name suggests, in a straight box you have got a bijective mapping, therefore you take inputs of 24 bits for example, your output is also 24 bits, so that you can actually encode in the form of a table. So, you see that table, what does it imply?

It implies that of, so you see, that there are 12 columns in each row, so therefore, you see that there are, the first bit of the output is derived from the first bit of the input, but the second bit of the output is derived from the 15th bit of the output, the third bit of the output is derived from the second bit of the input; therefore, it is just a permutation.

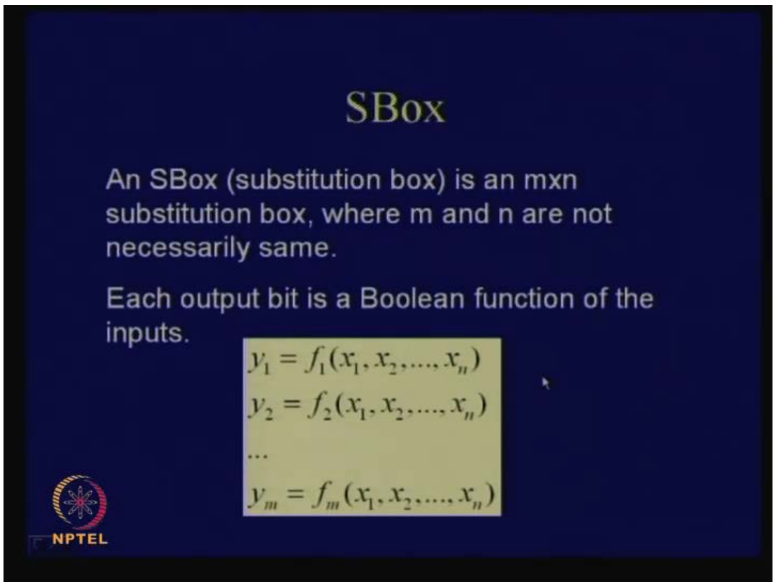
Similarly, you see, for expansion boxes, you have got a 12 cross 24, which means that your input size is actually small and your output size is large, so what does it imply? There are repetitions. So, you see, that in your first case, this one is 1 and you find that there is another one, so I made the table in such a way, there are actually, there are two repetitions of each.

So, you will find a 1 1 here, a 3 here and 3 here and similarly, we will find that there frequency of each element is actually 2, so what about a compression box? So, just the opposite of expansion, expansion box, you take 24- bits and compress that into 12-bits of the output, so which means, you have to drop some bits.

So, can you tell me from the point of loss of information, which is essentially a, which is essentially gives you the same information in the output as the input and which gives you the minimum number of information in the output or which gives you a more information in the output? Can you order them by information leakage?

See for example, compression box obviously leads to a loss of information. So, in an, I mean, in an expansion box we have got redundancy and in a straight box, it is essentially as same thing, what is one-to-one. So, you see, that information, through information actually, you can nicely, sort of, quantize, if you would like to do so. So, essentially the constructions of all these boxes, follows its root from information theory; although, we will not be discussing those things in our class, but just thought of hinting in the class.

(Refer Slide Time: 28:11)



SBox

An SBox (substitution box) is an mxn substitution box, where m and n are not necessarily same.

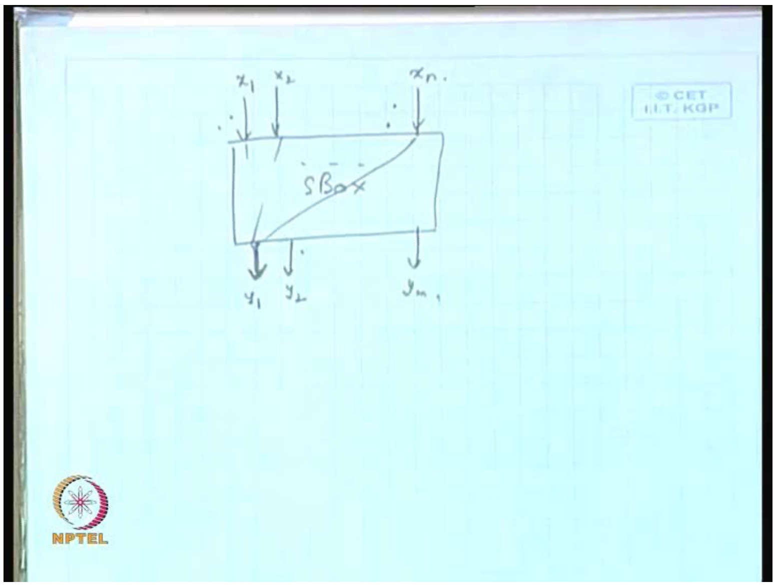
Each output bit is a Boolean function of the inputs.

$$y_1 = f_1(x_1, x_2, \ldots, x_n)$$
$$y_2 = f_2(x_1, x_2, \ldots, x_n)$$
$$\ldots$$
$$y_m = f_m(x_1, x_2, \ldots, x_n)$$

Say, the other important thing that we will be using is an S-box or a substitution box. So, a substitution box is typically an m cross n substitution box, which means, that there are m-bit inputs and n-bit corresponding outputs, where m and n are not necessarily the same. So, if the m and n are not the same, then obviously it is not a bijective mapping, it is not invertible mapping anymore.

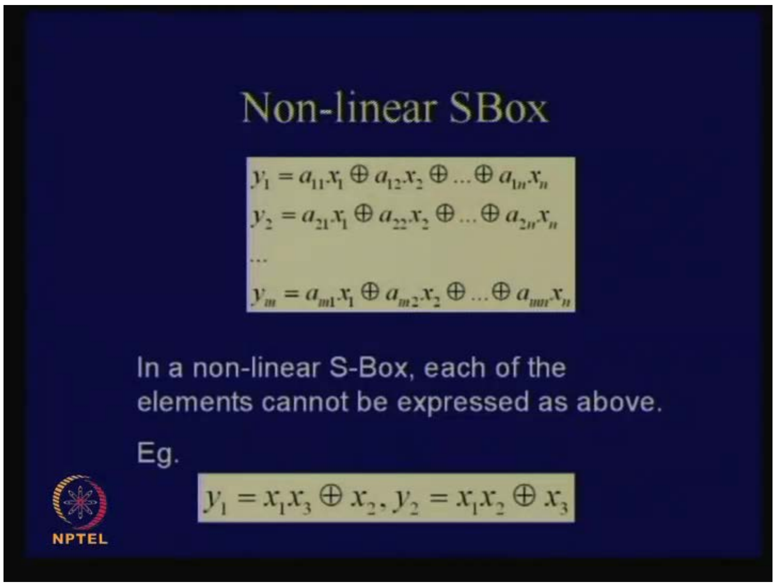
So, you can obviously understand, that each output bit, which is a, has to be a Boolean function of the inputs, so that if you consider, that your inputs are nothing but 0s and 1s, that is, their Boolean functions, then all these n outputs, you can indicate or denote them by n corresponding Boolean functions in the input. So, x 1 to x n are your input and y 1 to y m are your output. Do you understand this?

(Refer Slide Time: 28:59)



So, what I am saying is this, that is, your S-box essentially takes in inputs which are denoted by say x 1, x 2 and so until x n and your outputs are y 1, y 2 and so until y m. So, these y 1 is supposable a mapping of or a Boolean function of x 1, x 2 and so until x n. So, all of them, all of the outputs are essentially a Boolean map of x 1 to x n. Similarly, y 2 is also a Boolean function of x 1 to x n and there are certain properties which the S-box must satisfy, they are called interesting properties. So, we will try to keep one class to understand how or rather, the some of the principles behind design of S-boxes.

(Refer Slide Time: 29:48)



Non-linear SBox

$$y_1 = a_{11}x_1 \oplus a_{12}x_2 \oplus \ldots \oplus a_{1n}x_n$$
$$y_2 = a_{21}x_1 \oplus a_{22}x_2 \oplus \ldots \oplus a_{2n}x_n$$
$$\ldots$$
$$y_m = a_{m1}x_1 \oplus a_{m2}x_2 \oplus \ldots \oplus a_{mn}x_n$$

In a non-linear S-Box, each of the elements cannot be expressed as above.

Eg.

$$y_1 = x_1x_3 \oplus x_2, y_2 = x_1x_2 \oplus x_3$$

But temporarily, let us assume, that just let us note one thing, that a non-linear, what is a definition of a non-linear S-box? So, I essentially give you, given you some Boolean mappings here, like y 1 to y n expanded upon the previous f 1 to f m functions. So, for example, here you note that I can denote y 1 as y 1 is equal to a 1 1 x 1 XORed with a 1 2 x 2 and so until y a 1 and x n. Similarly, I do it for the all n-bit outputs.

So, this mapping is actually a linear mapping with respect to, with the Exclusive OR operation, why? Because easily you can understand that you can represent this by a matrix transformation; I can represent this by a matrix transformation and this is the something, which is called linear function.

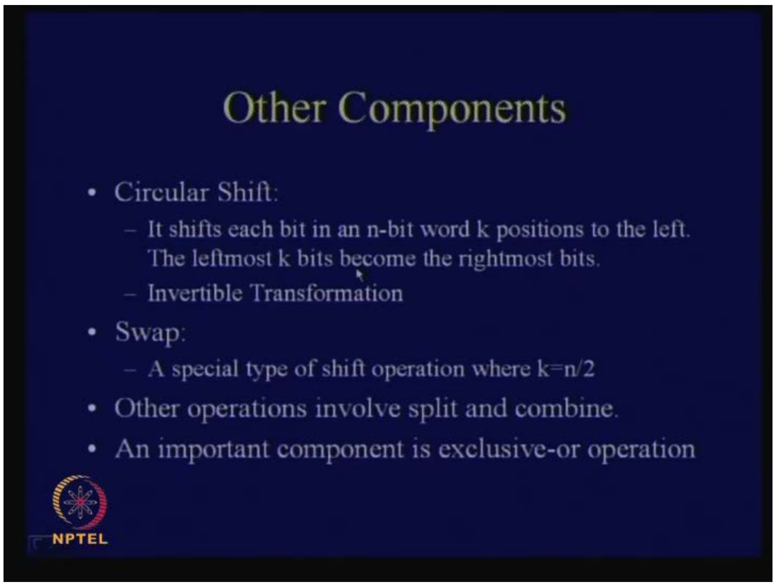
So, we have defined, what is the linear function in the last day's class; so, this is linear with respect to the Exclusive OR operation. So, if in a particular S-box, each of the elements cannot be expressed as above, then we call that to be a non-linear S-box.

So, in our last day's class, we discussed that actually, we require non-linearity to define the concept of rounds. Therefore, we, if we essentially require non-linearity also, therefore, this is a typical, this some example of a non-linear S-box you see, that y 1 is equal to x 1 into x 3 XORed with x 2 and y 2 is equal to x 1 into x 3 x 2; this is a Boolean add, so x 1 into XORed with x 3.

So this is a mapping for example, which you cannot denote in this form. Therefore, you see that this is an example of a non-linear S-box; so, this is the typical example of a non-linear S-box. And we will try to see what, how do I measure this amount of non-linearity also, but this is just to indicate, that here actual S-boxes, if represent them by Boolean mappings, would look like this or this; it would look like this, it would look like this transformations and not like this.

If we find that there is an S-box, there is an S-box given to you, which you can represent this, I mean, which you can represent in this form, it is considered to be a bad design; a good design would also give you a good amount of non-linearity. Is this clear?

(Refer Slide Time: 32:08)



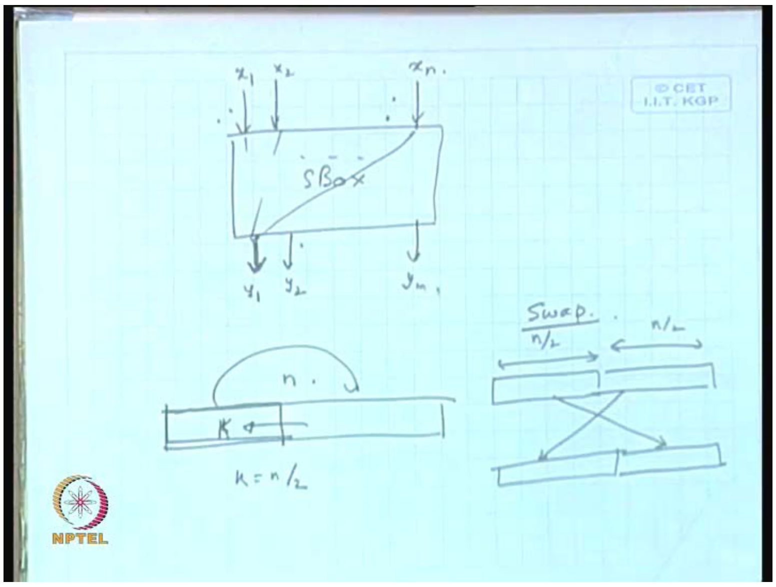
Other Components

- Circular Shift:
  - It shifts each bit in an n-bit word k positions to the left. The leftmost k bits become the rightmost bits.
  - Invertible Transformation
- Swap:
  - A special type of shift operation where k=n/2
- Other operations involve split and combine.
- An important component is exclusive-or operation

So, others, some other components are as follows, see, consider a circular shift, so in a circular shift, each, so essentially, as the name suggests, what we take, do is that you take some bits and you do a circular shift. So, it is very simple.

It is actually an invertible transformation also. So, therefore, you know, that if you shift left, the reverse operation would be, shift right. So, the other operations that we use is the swap operation, it is a special type of shift operation or a circular shift operation, where k is equal to, n-bits, n by 2-bits. So, therefore, in a circular shift, what we do is that you shift each bit in an n-bit word k positions to the left and the leftmost k bits become the rightmost bits.

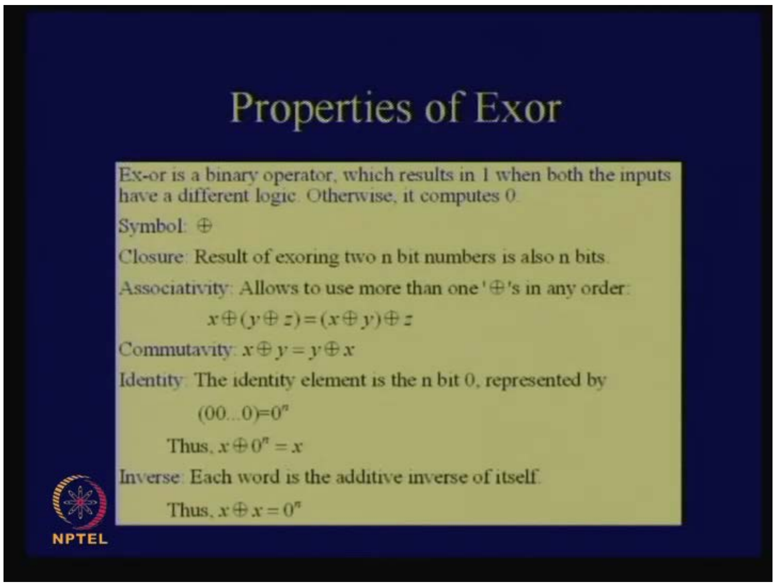
(Refer Slide Time: 32:52)



Therefore, in a circular shift what you do? You take n bits and the first k bits essentially, come here and the other ones get shifted left, so therefore, you shift them. So, therefore, if the value of k is equal to n by 2, then you get a swap operation.

So, in a swap operation, typically you have got n by 2 bits and you take this and you place this here, and you take this and you place this here. Therefore, this is a typical example of a swap operation. So, some other operations are split and combine, which is very simple, like you take a block of data and you split that into two components and the reverse operation would be to combine.

(Refer Slide Time: 33:46)



Properties of Exor

Ex-or is a binary operator, which results in 1 when both the inputs have a different logic. Otherwise, it computes 0

Symbol: $\oplus$

Closure: Result of exoring two n bit numbers is also n bits.

Associativity: Allows to use more than one '$\oplus$'s in any order:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z$$

Commutavity: $x \oplus y = y \oplus x$

Identity: The identity element is the n bit 0, represented by

$(00...0) = 0^n$

Thus, $x \oplus 0^n = x$

Inverse: Each word is the additive inverse of itself.

Thus, $x \oplus x = 0^n$

But we will be discussing in details about an important concept, which is the exclusive-or operation or the Ex-or operation. So, let us see some properties of the Ex-or operation. So, the properties of the Ex-or, I mean, Ex-or is a binary operator. We know it is a binary operator, which results in 1 when both the inputs have got a different logic, otherwise it computes to 0.

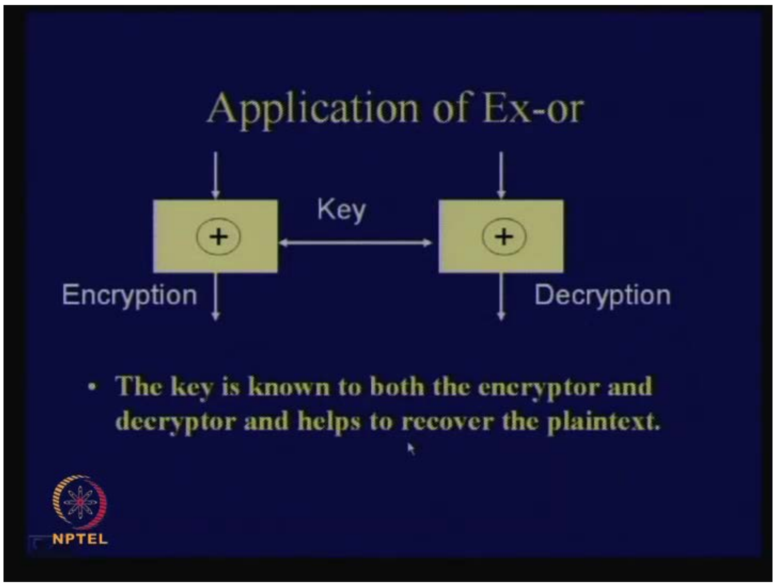
So, we know, that of an Ex-or, the symbol that we will be using is this and it satisfied certain properties, like: closure, associativity, commutavity, identity and inverse. So, therefore, can you tell me that what does it form?

Not only a group, but it forms an abelian group, it forms a commutative group. Therefore, you see, that it satisfies closure because the result of Ex-ORing 2 n-bit numbers, you can encode that by another n-bit number, it follows associativity because it allows to use more than one Ex-ors in any order.

So, if I want to do x Ex-or y Ex-or z and this is the associativity, I can also do it like doing an Ex-or of x and y first and then doing an Ex-or with that; commutavity, because it do, if you do x Ex-or y, it is same as doing y Ex-or x; identity, why? Because the all 0 or I denote them by 0 power n is an identity in the group because if I do x Ex-ORed with 0 power n, then I obtain back x.

It also has an inverse because each word is its self-inverse. So, if we take x and we Ex-or that with x, then you get back 0 n. therefore, this is it, this proves that it also has got inverse and it acts actually as self-inverse.

(Refer Slide Time: 35:23)



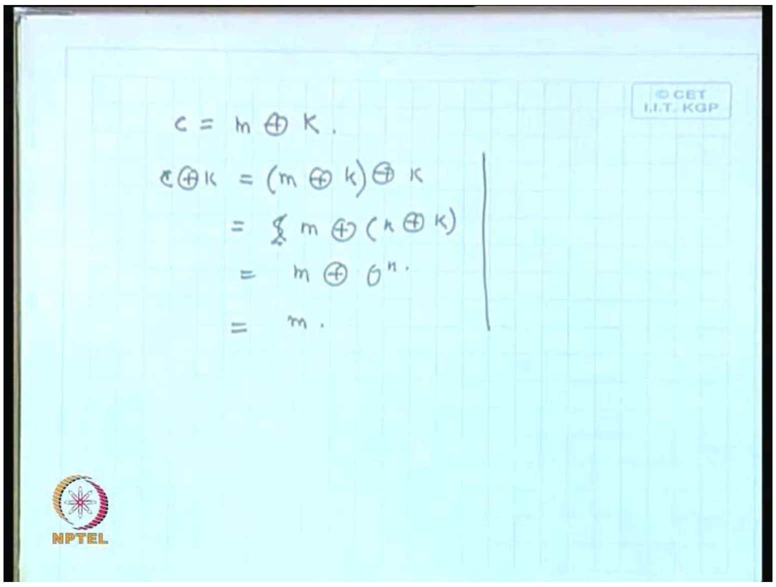
Application of Ex-or

- The key is known to both the encryptor and decryptor and helps to recover the plaintext.

So, an application of Ex-or would be like this. So, when you want to do an encryption, then what we do is that you take the plaintext and we Ex-or that with the key and you obtain the ciphertext.

And if you want to do the opposite, then seems we have, also have the key-in material, what you do is that you take this ciphertext, Ex-or it with the key again and you obtain back this. So, the key is known to both, the encryptor and the decryptor, and this helps to recover the plaintext.

(Refer Slide Time: 35:52)



$c = m \oplus K.$

$c \oplus k = (m \oplus k) \oplus k$

$= m \oplus (k \oplus k)$

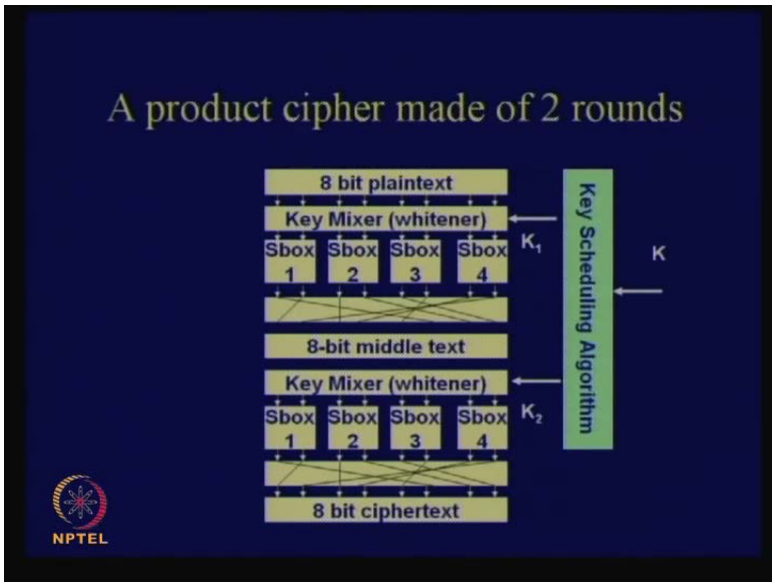
$= m \oplus 0^n.$

$= m.$

So, this is quite self-explanatory from the previous things that we proved because if I denote here ciphertext to the m Ex-or Ex-or of your message and your key, then if you can easily show, that if you do C an Ex-or that with k, then that means, what you do an m Ex-or with k and Ex-or that with k. So, because of the associativity, you can actually write this as m Ex-or with k Ex-or k and since k is a self-inverse, then you obtain back m Ex-or with 0 n, and you know that 0 m is an identity in the group and therefore, you obtain back m. So, therefore, you can prove from the previous results that we had. So, therefore, this is a typical example.

Since, we have known the components we can actually try to think of how modern block cipher would have looked like? Therefore, this is an example of a product cipher.

(Refer Slide Time: 36:42)



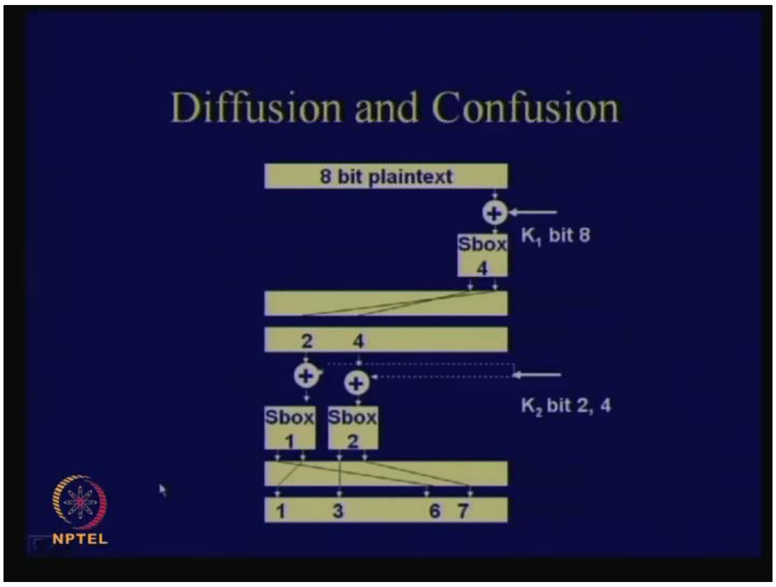
A product cipher made of 2 rounds

We discussed, what is a product cipher in last day’s class and it is made of two rounds. So, you see, that here, there is an 8-bit plaintext and there is some key mixer operation, which is a whitener like, it is also called key whitener. So, it is typically an Ex-or operation, Exclusive-OR operation with the key.

So, what you do is that there are four S-boxes, four simple S-boxes being placed here and then there is a permutation; so, therefore, it is nothing but a P-box or a permutation box. So, you see, that is a transposition of bits, it is a wiring in terms, if you would like to implement as a hardware, it is just a rewiring, it is a fixed mapping; it is a fixed transposition mapping.

So, then you obtain something, which we have called as 8-bit middle texts and then you take the output, again Ex-or that with the next round key, and then something which is called the key scheduling algorithm. What it does? It takes the input key and produces all the round keys. So, you take k, you produce k 1, you take actually k 1 may be, and produce k 2 and you just try to make all the round keys, one after the other.

(Refer Slide Time: 37:58)



Diffusion and Confusion

So, this is a typical example of a product cipher, which is made of two rounds. In order to appreciate the confusion and diffusion property of this cipher, we can observe through this particular, sort of, flow of data. See, consider this last bit, of the output, of the input. So, you see, that this is essentially getting Ex-or with 8-bit of the key because that the 8-bit, and I have considered 8-bit of the plaintext is getting Ex-or with the 8-bit of the key and then, you put that, that output goes to the S-box, the 4th S-box. So, how many bits does it affect in the 4th, I mean, how many bits of the output does it get, gets affected? 2 bits.

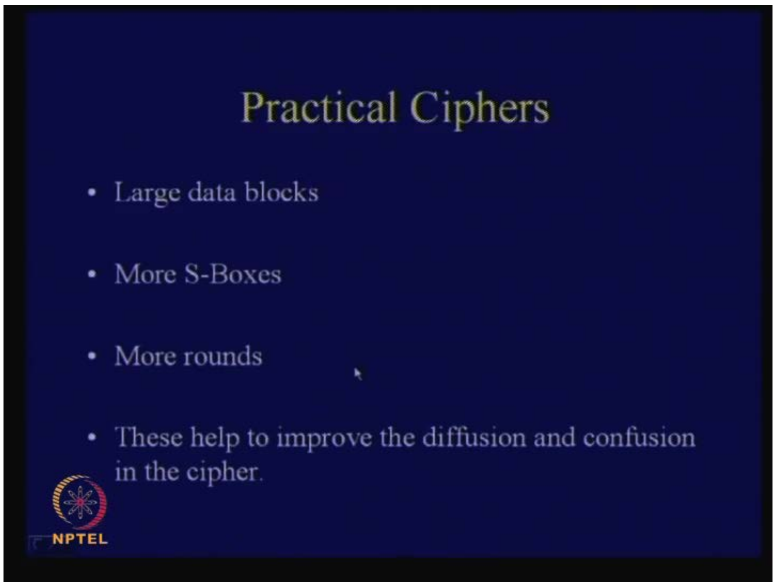
These 2 bits essentially, goes to also 2 another bits here. So, you see, here it goes to 2 and 4. So, now, you see, that this bit affects this S-box and this bit affects this S-box, so which means, that you have got more number of disturbed S-boxes.

And then this particular bit is getting Ex-or with second bit of the key and this 4th bit is getting Ex-or with the 4th bit of the key and they go to the these S-box 1 and S-box two of the second round. And similarly, it is also affecting 2 bits here and 2 bits here and both of them are, both the S-box outputs are getting transposed into 4 output bits. So, the point you observe is that if I have disturbed only 1 bit in the input, now we have got 4 bits in the output, which have got disturbed.

We also know n, that 4 is actually n by 2, approximately. So, therefore, in a random mapping also, if I would have changed only 1 bit in the input, then half of the output bits would have been affected approximately, so that is something which is called avalenge affect or something which is called an avalenge criteria. Therefore, this is an example of an avalenge criteria and also an example of diffusion. So, you see, that this is the essentially, you see that there is an obscurity of the number of bits of the output between the number of bits of the output and the bits in the input, and the confusion, you can also observe, through this phenomenon, that the 1st, 3rd, 6th and 7th bit of the output, essentially a dependent on the 8-bit of the key, second bit of the key, and the 4th bit of the key.

So, therefore, there is a problem of, or rather there is a hiding of, hiding of, hiding of mapping, hiding of how the outputs bit have been generated or rather, hiding of the relation of the output bits with the plaintext and hiding of the relation of the cipher text with the key bits. Also, the first one is the example of diffusion and the second one is an example of confusion.

(Refer Slide Time: 40:25)



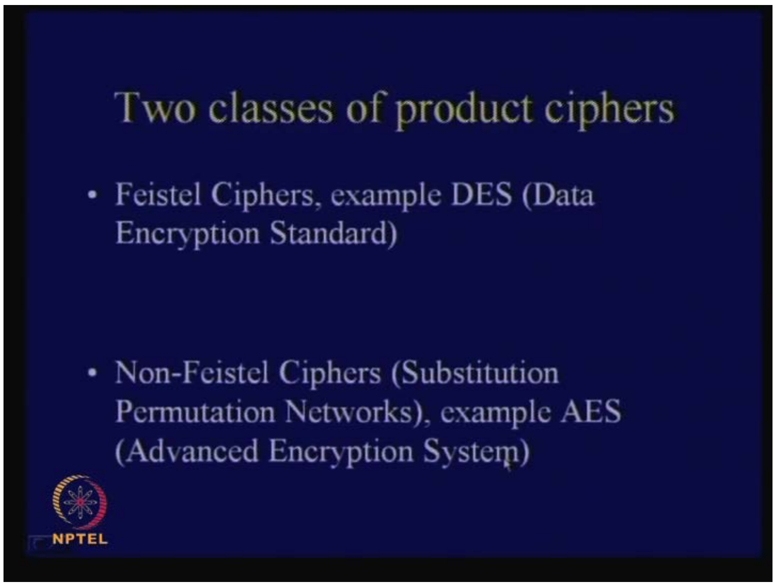
Practical Ciphers

- Large data blocks
- More S-Boxes
- More rounds
- These help to improve the diffusion and confusion in the cipher.

No, no, in those cases we cannot do decryption, but those things we can actually apply in a certain technique, which is known as Feistel ciphers; we will come to that, how we can apply them? Now, what you say is correct, that if these boxes would have been essentially something like a compression function, then we would not have been able to recover, obviously; but we can apply them, using, in another scenario.

So, in practical ciphers, actually you require large data blocks than 8-bits and you require more S-boxes to get disturbed, and you require more number of rounds, and this helps to improve the diffusion and confusion properties in the cipher. So, that is the final objective that I want to enhance or improve the diffusion and confusion properties in the cipher.

(Refer Slide Time: 41:28)

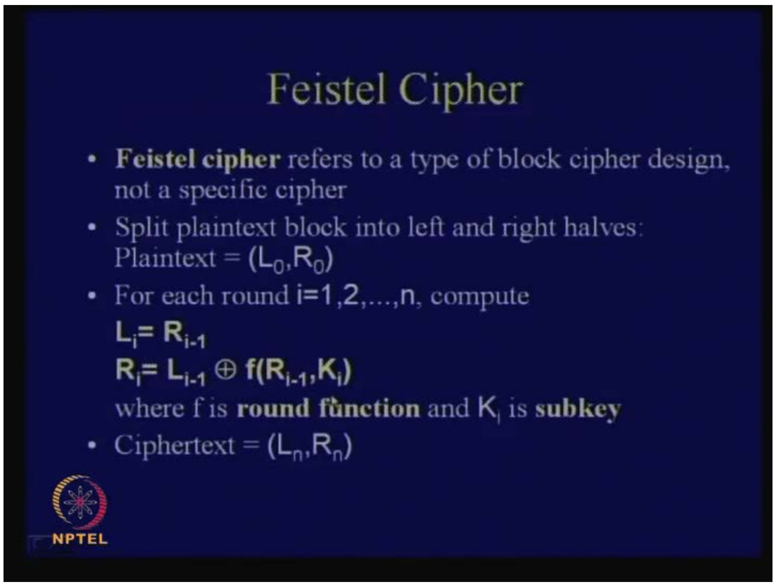


Two classes of product ciphers

- Feistel Ciphers, example DES (Data Encryption Standard)
- Non-Feistel Ciphers (Substitution Permutation Networks), example AES (Advanced Encryption System)

So, we will just discuss about two classes of product ciphers, one is called a Feistel ciphers, example of that – DES, it is called Data Encryption Standard and also non-Feistel ciphers, and example of that is AES. So, non-Feistel ciphers are also called Substitution Permutation Networks or SPN ciphers. So, we will try to understand these two things, may be in our next day’s class, but just in today’s class, we will try to understand the concept of Feistel ciphers without going into DES.

(Refer Slide Time: 41:55)

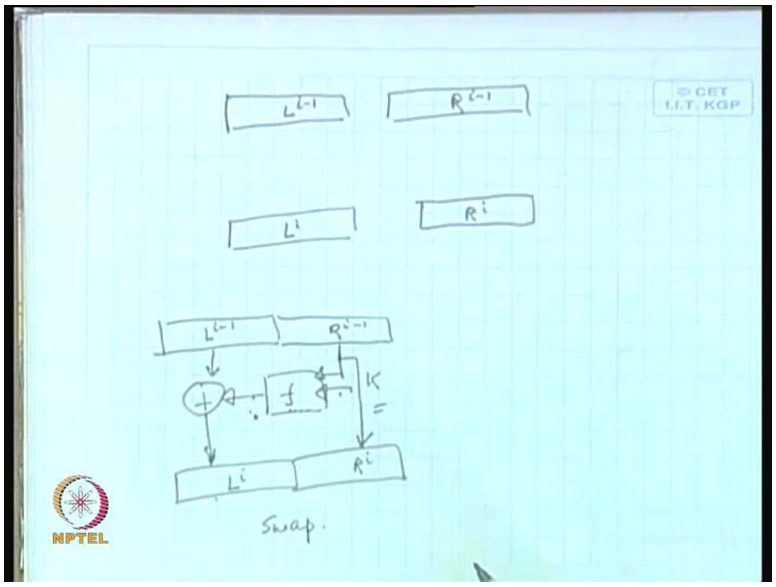


Feistel Cipher

- **Feistel cipher** refers to a type of block cipher design, not a specific cipher
- Split plaintext block into left and right halves: Plaintext = (L_0,R_0)
- For each round $i=1,2,\ldots,n$, compute
  $L_i= R_{i-1}$
  $R_i= L_{i-1} \oplus f(R_{i-1},K_i)$
  where f is **round function** and K_i is **subkey**
- Ciphertext = (L_n,R_n)

So, a Feistel cipher refers to a type of block cipher design and not a specific cipher. So, it is not a fixed cipher that we will discussing, but it is a general class of ciphers. So, for, what we do is that the first thing that we do is that we take a plaintext and we divide that into two parts, one of the parts is called the, left mode, left block and the other part is called the right block.

(Refer Slide Time: 42:33)



And then, you do certain transformations, may be, you do n rounds. So, first thing what you do is that you consider this, take a left block and you would essentially one, you also have right block here, so I call that by L i minus 1 and R i minus 1 to denote, that this is the…, What is the output of the i minus 1 in step, and I would like to obtain L i and R i, so question is how? So, we want to understand this internal mapping.

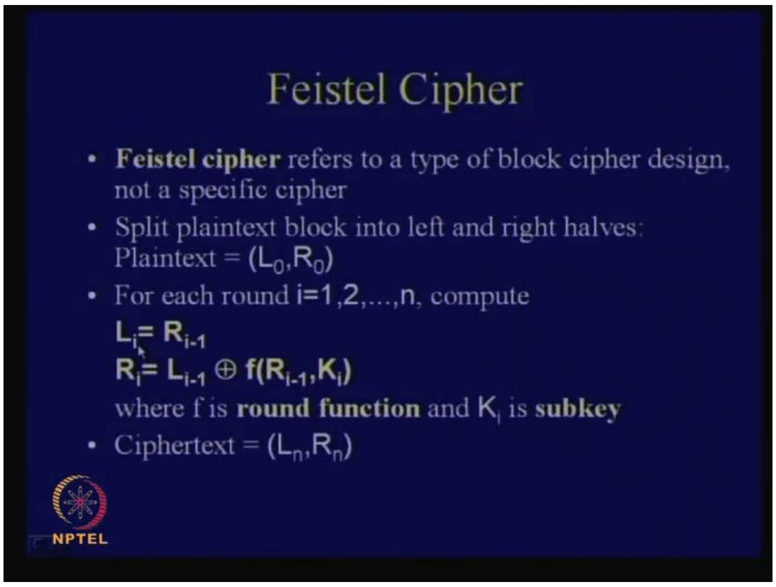
So, you see, that if I had taken this for example, this input, I call that by say, L i minus 1 and would have obtained L i. One possible way would have been to take this and Ex-OR this with the output of a function f, whose input is k and obtain the output. So, you see, that this function f could actually had been a compression function and still, we would have been able to recover the input, why? Because we know the k value and therefore, we would have felt, fading this value of k and again, the decrypter also would have obtained the value of f k. And then, you would have simply Ex-OR with this output and obtain back the plaintext. Therefore, you see that actually a compression function can be applied for cryptography also, for block cipher designs also. So, this answers probably, little bit of your questions.

But the other thing that we would also want to do is that we are not very comfortable with having the input only as key. So, we would also like a part of the input to go to this function f also. So, what we do is that we extend this and observe R i minus 1, and this output of R i minus 1, we feed to the other part of the function f.

And now, you see, that in order to obtain, or rather in order to decrypt, you actually, require this value of R i minus 1 also. So, one simple thing, that we could have done is that we could have just passed this R i minus 1 to the output as R i.

But what is the problem in this case? If you had just iterated this block again and again, what would have been the problem?

(Refer Slide Time: 45:09)

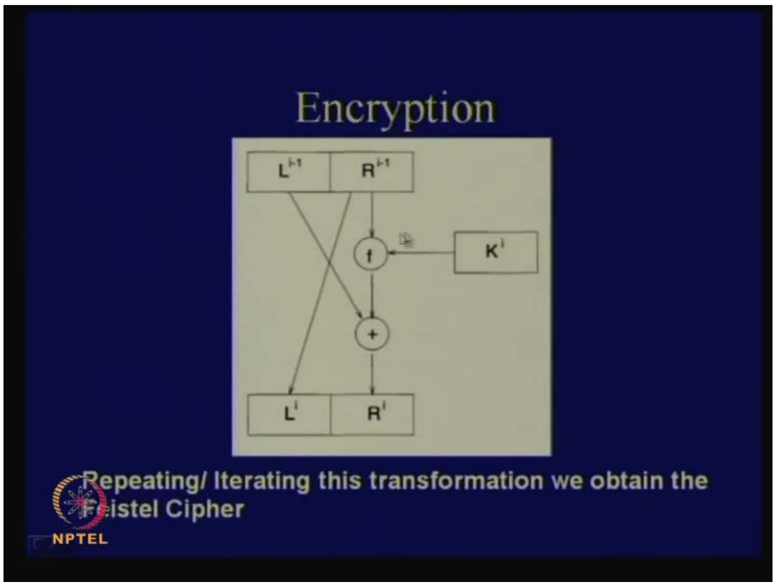


Feistel Cipher

- **Feistel cipher** refers to a type of block cipher design, not a specific cipher
- Split plaintext block into left and right halves: Plaintext = (L_0,R_0)
- For each round i=1,2,…,n, compute
  $L_i = R_{i-1}$
  $R_i = L_{i-1} \oplus f(R_{i-1},K_i)$
  where f is **round function** and K_i is **subkey**
- Ciphertext = (L_n,R_n)

The problem would have been that right hand just would have remained the same. So, what we do is that after this, we do a swap operation. So, you follow that with the swap operations, you see, that actually, this property also gets disturbed. So, now you see, the how essentially, that how the Feistel cipher mapping look like. It was just like this, that is, L i has been attributed to R i minus 1 is equal to R i minus 1.

(Refer Slide Time: 45:29)

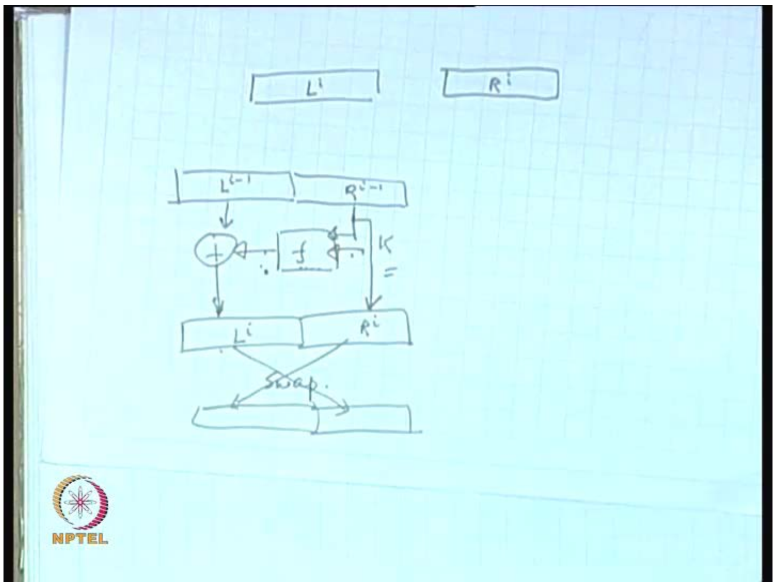


Encryption

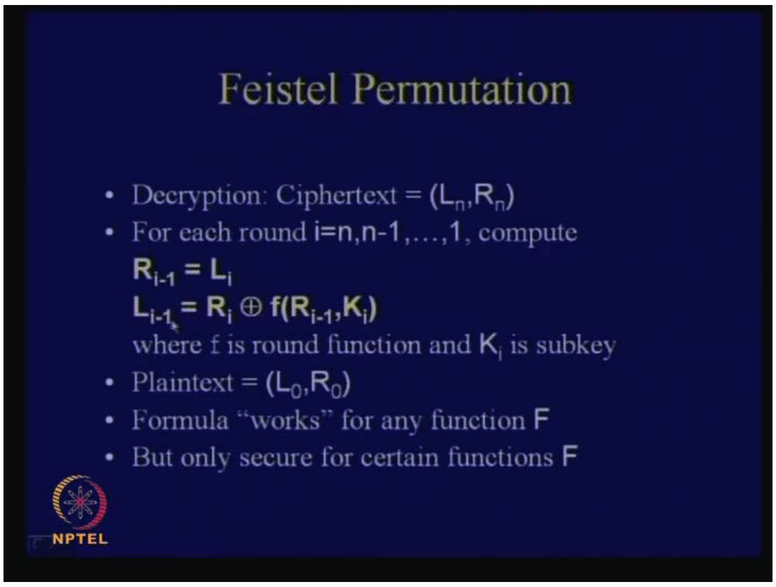
Repeating/ Iterating this transformation we obtain the Feistel Cipher

The R i minus 1 value is assigned to L i and L i minus 1 Ex-OR with f of R i minus 1 and comma K i is been assigned to R i. So, therefore, if cryptographically this would have been look like this, therefore you take L i minus 1 R i minus 1, you put in this function f K i here, you obtain this output, Ex-OR this and you obtain this R i output, and R i minus 1 has been passed to the L i output. So, therefore this is the same thing that we saw here with the extra swap.

(Refer Slide Time: 45:47)



(Refer Slide Time: 46:07)



Feistel Permutation

- Decryption: Ciphertext = (L_n, R_n)
- For each round $i=n,n-1,\ldots,1$, compute
  $R_{i-1} = L_i$
  $L_{i-1} = R_i \oplus f(R_{i-1}, K_i)$
  where f is round function and K_i is subkey
- Plaintext = (L_0, R_0)
- Formula "works" for any function F
- But only secure for certain functions F

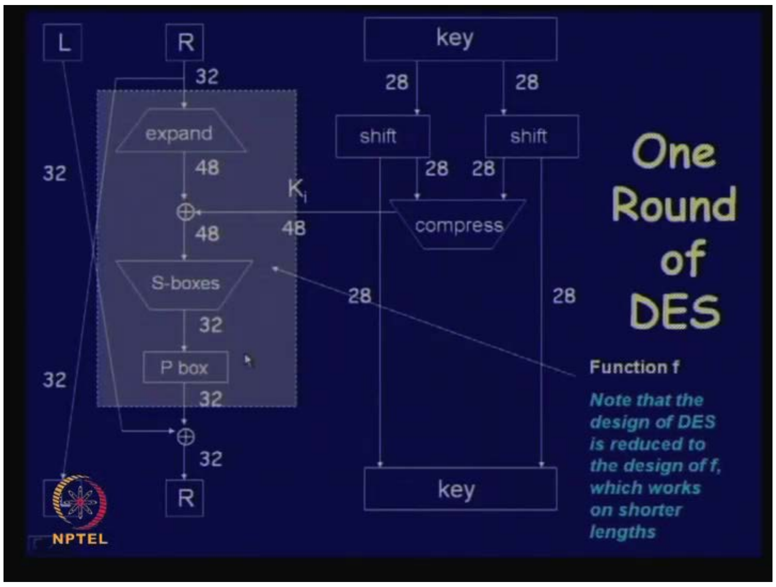
So, therefore, I mean, this would have been the same thing essentially, if you had just swapped this output. So, decryption is also quite easy now, you see the decryption, how decryption would have worked? You just take R i minus 1 and you have, you can obtain

L i and L i minus 1 is equal to R i Ex-OR with f(R i minus 1, K i). So, you, all of us know the value of R i minus 1 because that is exactly equal to the value of L i. So, therefore, the point is, that here this actually, this actually, this function f, any function f, here would have sufficed.

The decryption would not have been a problem, any function f you take, we can still do the decryption. So, therefore, it works, the formula works for any function f, but security will not obtain for any function f; there is only certain functions f which will give you security.

So, therefore, the, now you see, that the, this capital F is actually a small f; so you can utilize this capital F by small f. Therefore, you see, that the problem of security, now we have actually, sort of, reduced our problem to the design of the small function f. So, where the big problem, like we have to design say, a 128-bit block cipher or a 64-bit block cipher, so we have actually reduced that problem now to the design of a small function called f.

(Refer Slide Time: 47:12)



One Round of DES

Function f

Note that the design of DES is reduced to the design of f, which works on shorter lengths

So, you see that this is the typical round of a DES, without going into the details because we will discuss this in our next class. Just observe one thing; this is nothing but the application of the Feistel cipher. You take L, this is being Ex-ORed with the corresponding output of a function f and the function f has got two inputs - one is the R part of the previous round and the other part is the value of the key.

And this goes to here, right part of the output ciphertext, output of this round and this part, that is, the left part is being assigned from the right part of the previous round. Now, you see, that internally there are lot of things inside this function f. So, you take a 32 bit in case of DES because DES has got totally 64 bits, we divide that into 2 parts – 32 bits, 32 bits.

Then we expand this 32 bits using a P-box expansion diffusion box, expanded to 48 bits, you Ex-OR that with 48 bits of the round keys and then you obtain 48 bits here, then you take this 48 bits and you pass that in through an S-box, which actually takes 48 bits and compresses that to 32 bits.

And then you pass that again through a P-box, this is a straight box. So, therefore, you take 32 bits and you map that into a 32 bits and you Ex-OR that with 32 bits of your left part, and you obtain the output, 32 bits of the round.

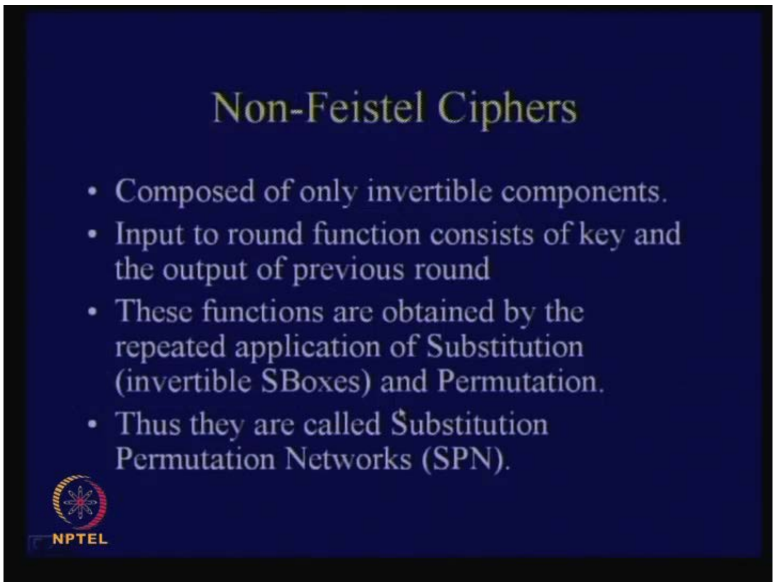
So, here, you, that there are 48 bits of the key and they actually derive by using a function, which is called the key scheduling function. So, you take the input key and there is a certain algorithm through which the 48 bit round key is derived.

And another thing we noted here is that, I will come to this in our next day's class, that is, this S-box, although it shows that it is a 48 to 32 bit mapping, is actually composed of the smaller S-boxes. So, all the smaller S-boxes are 6 to 4 mappings, that is, it takes 6 bits and maps that to 4 bits of the output.

So, how many S-boxes are here? There are 8 S-boxes, S 1 to S 8. So, each of these S-boxes, there is a principle behind the design of this S boxes, there is an application of lot of communitative coding theories and stuff like that. So, that, we will come to that at the end in our class.

What the point is, that note that the design of the DES is reduced to the design of function f, which works on shorter lengths. This is typically what we do as computer scientist is that when we have a bigger problem as the engineers, we reduce that to a small problem and solve that. And what we have done exactly is that we have taken a big problem and we have converted that into a small and much manageable problem.

(Refer Slide Time: 49:48)



Non-Feistel Ciphers

- Composed of only invertible components.
- Input to round function consists of key and the output of previous round
- These functions are obtained by the repeated application of Substitution (invertible SBoxes) and Permutation.
- Thus they are called Substitution Permutation Networks (SPN).

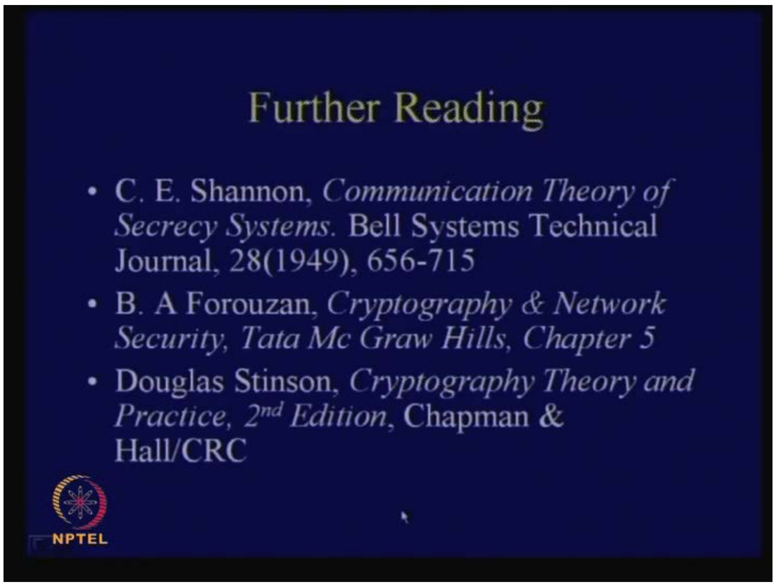
So, the other part of ciphers is called non-Feistel ciphers. It is composed of only invertible components.

So, you see that the input to round, put in, the input to round function consists of keys and the output of the previous round. So, in this case actually, we will discuss this when you talk about AES, whatever mappings or whatever transformations that we do here, are all of them are actually, invertible mappings.

So, like in Feistel ciphers, we had used non-invertible mappings also, but in SPN ciphers or in substitution permutation networks are what I classify as non-Feistel ciphers, all of the mappings to be essentially composed of invertible mappings.

The S-box would be invertible, all the permutations will be invertible, and all the transformations are essentially one-to-one.

(Refer Slide Time: 50:36)

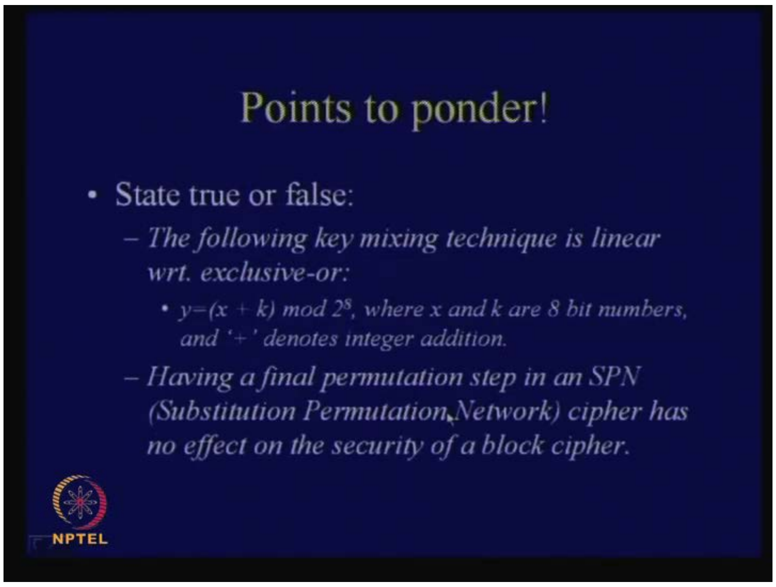


Further Reading

- C. E. Shannon, *Communication Theory of Secrecy Systems*. Bell Systems Technical Journal, 28(1949), 656-715
- B. A Forouzan, *Cryptography & Network Security, Tata Mc Graw Hills, Chapter 5*
- Douglas Stinson, *Cryptography Theory and Practice, 2nd Edition*, Chapman & Hall/CRC

So, these are some of the references that we have used in our class. Obviously, we have used certain books of Shannon's paper, we have used our standard text book of Douglas Stinson and I have also referred to another book, which is written by Forouzan. It is a recent book on Cryptography and Network Security and this chapter is slightly brief in chapter 5. So, you can, so there is an Indian edition of this book also available, so maybe, you can refer to this book also.

(Refer Slide Time: 51:05)



Points to ponder!

- State true or false:
  - *The following key mixing technique is linear wrt. exclusive-or:*
    - *$y=(x+k) \bmod 2^8$, where x and k are 8 bit numbers, and '+' denotes integer addition.*
  - *Having a final permutation step in an SPN (Substitution Permutation Network) cipher has no effect on the security of a block cipher.*

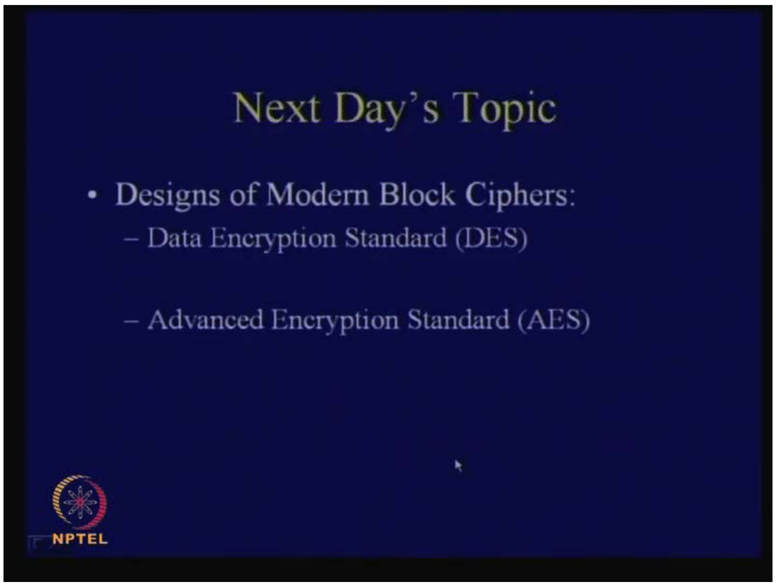
And certain points to ponder, so you can just think of these questions, well, you do not require to submit, but you just think of these questions. It says, that a following key mixing technique, you have to just say whether it is true or false, it is linear with the respective Exclusive-OR.

So, what I have done is that I have taken x, I have added that with the value of the key, that is, the k and here, the plus is essentially an integer addition. So, it is x plus k modulo 2 power of 8, so where x and k are 8-bit numbers and plus denotes integer addition, so what you have to say is that whether each of this bits of the output, each of the output bits are essentially, are linear mapping of the inputs or rather, they are non-linear mappings of the inputs?

And the other thing to be thought is that having, so we are discussing about substitutions and permutations, so let us imagine, that before in the final stage, that is, having a final permutation step in an SPN cipher, whether that would have increased the security of the cipher or not?

So, therefore, to consider that whether having a final permutation state in an SPN cipher has got no effect on the security of a block cipher? So, you have to reflect, whether it is true or false.

(Refer Slide Time: 52:10)



Next Day's Topic

- Designs of Modern Block Ciphers:
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)

So our next day’s topic will be designs of modern block ciphers. We will try to understand, or rather go into the, or rather at least, look into the design of DES and AES, without really going into the design methodologies.