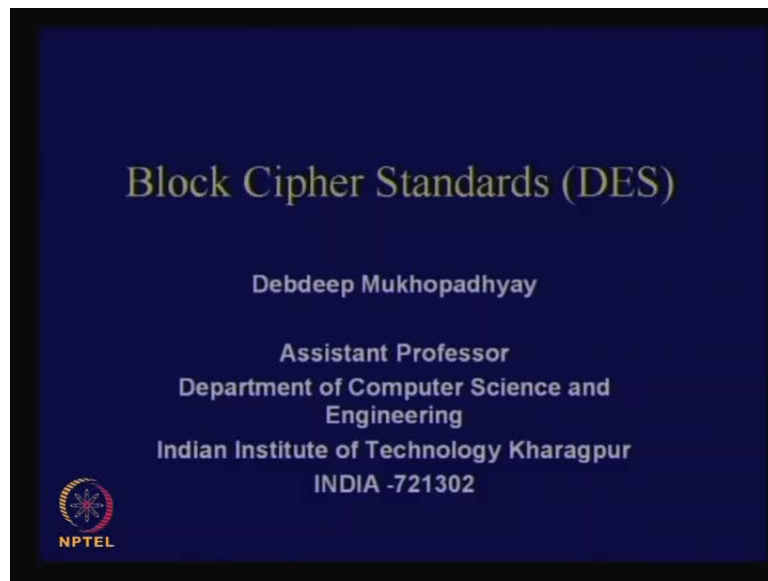


**Cryptography and Network Security**  
**Prof. D. Mukhopadhyay**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture No. # 11**  
**Block Cipher Standards (DES)**

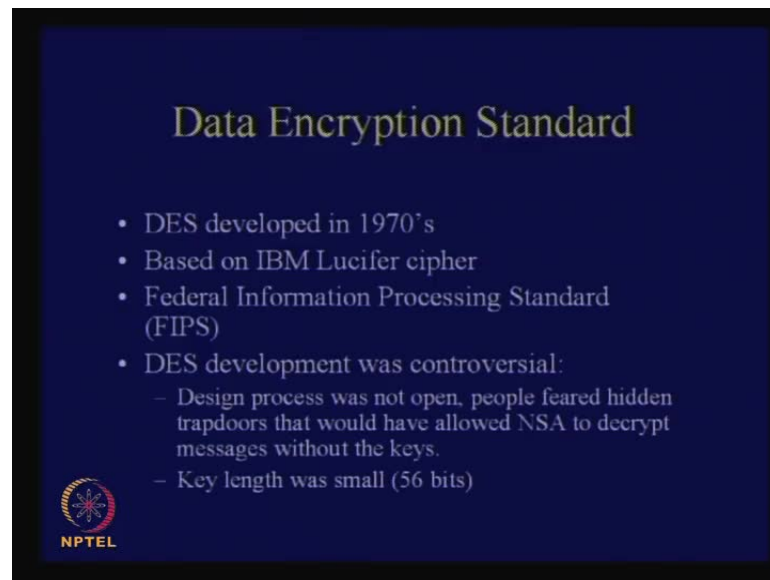
(Refer Slide Time: 00:28)



So, today, we will study about block cipher standards. So, as we have seen in our last day's class, we have understood, what are a feistel cipher and the underlying principles behind the construction of feistel ciphers. So, today, we will see one example of such kind of cipher, that, so therefore, this class of ciphers that we have been talking about is called feistel ciphers.

So, it was designed by a person called Horst Feistel and we have seen little bit introducing about what are the underlying principles, but today we will see one real life cipher, which is known as a DES or data encryption standard, and we will try to understand some of the principles behind it.

(Refer Slide Time: 01:04)



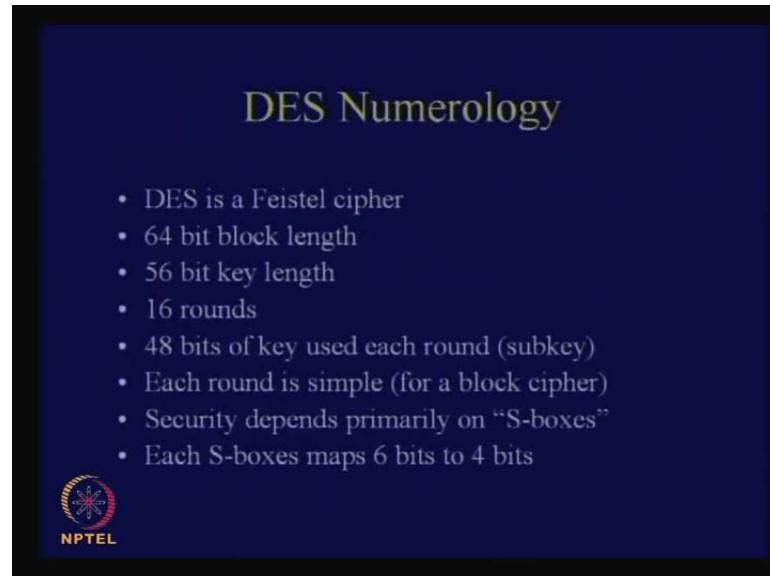
So, brief history is as follows, that DES was developed around 1970's, 1975 around. So, it was based on IBM's Lucifer cipher; so, there was a Lucifer cipher, which was designed by IBM and IBM actually made certain modifications to that and proposed cipher, which was named as DES. So, it was, immediately after that it was actually adopted as a standard by FIPS, so it is federal information processing standard. So, they adopted it as a standard.

So, DES development was quite controversial actually, because there were lot of, sort of, mystery behind the design of DES because people did not actually really (( )) about what are the internal principles for which DES had certain structures. Therefore, design process was not open and people fear, that actually hidden trapdoors are existing inside, so which enabled NSA, for example, that is, in NSA2 essentially, decrypt messages without the actual optimum of the key.

So, there was quite alarming kind of fear, but people have evaluated DES over the years now and since, even today they have not been able to find out a relevant trap door. They are now, more or less, convinced that probably, no such trap door existed. And also, the key length was also fairly small. A 56 bit key even during 1970's was considered to be quite small and today is actually small. So, today, you can actually break that in, may be, less than a day. So, of course, I mean there are actually not only places outside our country, but also inside our country, there are laboratory facilities where people can

actually use large number of parallel computations and actually does a brute force attack on DES. So, these are some points, I mean, some points to be noted about DES.

(Refer Slide Time: 02:51)



But DES is a not only a very good and strong design of a cipher, so what essentially the designers of DES say is, that this is a feistel cipher, which works or operates on 64-bit block lengths, so, and the key length is 56 bit key.

So, which means, that the designers of DES gave a guarantee of a security margin of 2 power 56. So, if there is any attack, which essentially is better than 2 power of 56 search, then that is considered to be an attack.

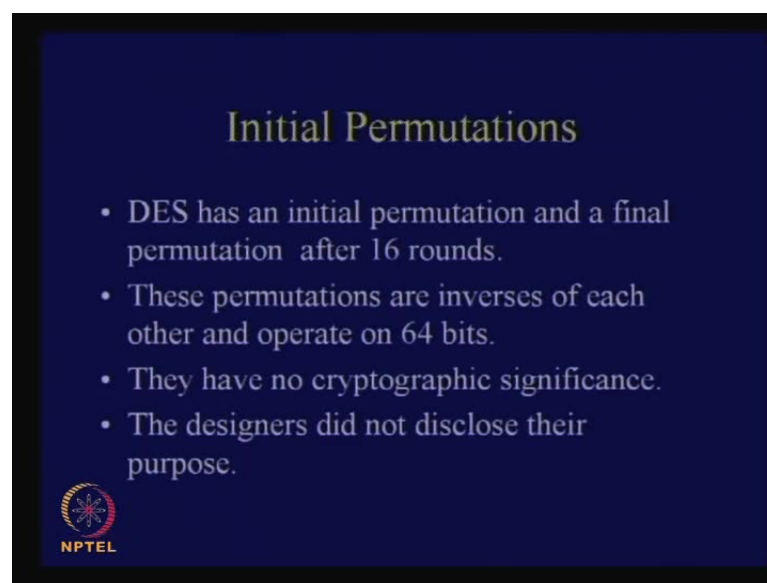
So, actually, there is one kind of attack or there are some attacks, which is essentially reduces to be 2 to power of 55 or so, but that is not so much essential. Even then, I mean, smallest, I mean, what I, what I want to say is that the designers of DES, the promise that they have made, the promise had not been broken. People, they have tried various techniques to attack this cipher, you understand 1970 and today it is 2000, around 2010, so it is almost 40 years of people trying to break a cipher. So, if they have not been able to do certain, I mean, something very significant, it really speaks about the design quality of DES.

So, we will see, that there are 16 rounds in DES, that is, as I told you, all these are build up on round, so there are 16 rounds of DES and each round has got a round key or a

subkey and the total size of the round key is a, is 48 bits. So, each round keys of dimension of 48 bits and each round is simple for a block cipher, so therefore, we will see, what is the structure of each round? And security depends, as I told you, primarily on the S-boxes or the substitution boxes and each substitution box maps a 6 bit value to a 4 bit value.

So, that is the mapping of DES, so therefore, DES takes 6 bits of input and converts that into 4 bits of output. So, each round has essentially 8 S-boxes.

(Refer Slide Time: 04:52)

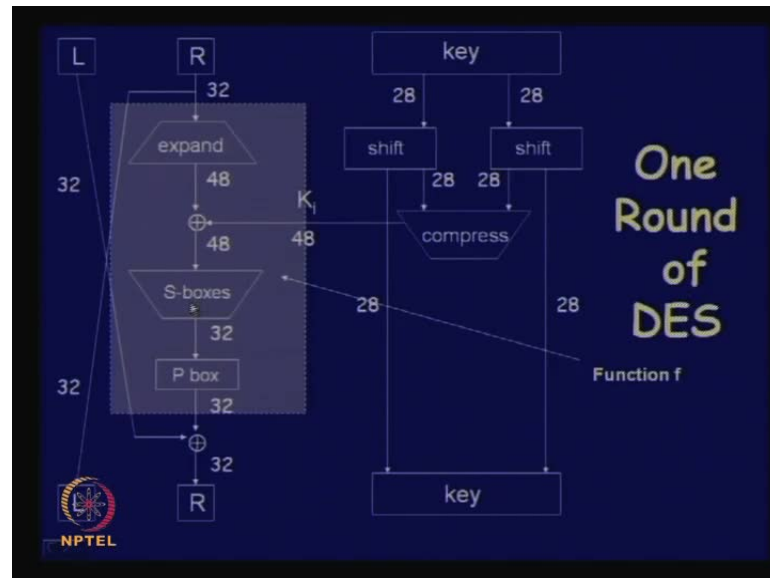


So, if it is just, I mean, I mean, before the start of DES and end of DES, there are 2 permutations actually. So, that is an initial permutation and there is an inverse permutation at the end of DES.

So, DES has an initial permutation and a final permutation after 16 rounds. So, these permutations are inverses of each other and operate on 64 bits, that is, it operates on the entire data. So, therefore, it operates on the entire 64 bit block of plain text. But they have got no cryptographic significance, so why they are there? People essentially do not know, therefore, the designers did not disclose the purpose of keeping those initial permutations. The only property, that we know is that the initial permutation, that is, the permutation, which is there at the beginning, is exactly the inverse of the permutation, which is there at the end.

So, if you see in your permutation table, if you see, that the 1st bit is, I mean, say the 58 bit of your initial permutation goes to the 1st bit of the output, we will find in the inverse permutation, just the opposite thing happens. So, they are exactly inverses of each other, but they essentially do not have any cryptographic significance, why? Because they do not add on to the security; we can imagine why.

(Refer Slide Time: 06:08)



This is how, typically, 1 round of DES looks like. So, you see, that DES is essentially a feistel cipher, so therefore, this diagram we have seen in the last day's class also, that is we have got 2 sub blocks, that is, the L and R, that is, the left and the right sub block. So, that means, the 64 bit block is been divided into 2 sub blocks of each of dimension 32 bits.

So, what is the idea of feistel cipher? You take R and you pass it to the left block of the next round and in order to obtain the right block, what you do is that you XOR the left block with the output of a function F, which operates on the right block and on the round key. So, therefore, this shaded region is the function F. So, you see, that there are certain things inside that function F; what are those things? It takes in a 32 bit R and it also takes a 48 bit K, so what it does? In order to do this XOR with the key, we have to expand these 32 bits to a 48 bit, so that is an expansion key box, expansion box, expansion diffusion box, all we have seen.

So, therefore, these sub blocks, essentially, expands 32 to 48 bits, which means, that there is some amount of redundancy of R inside out here and you take this and you XOR this with the key, and you obtain the 48 bit output, that you again pass through an S-box, so, which S-box is in this case a **compressor** also. It basically, what it does is, there are 8 S-boxes each of dimension 6 bit input and the output is 4 bits. So, 4 into 8 is 32, you, you obtain a 32 bit output from the S-boxes.

So, these S boxes, primarily, the non-linear layer of DES that is the only non-linear layer of DES, and its design is very important to the security of the entire cipher. So, therefore, what you do is, that you obtain the 32 bit output and then you pass through a straight key box, that is, it converts this into another 32 bit output and you XOR with the 32 bit left block and you obtain the 32 bit right block.

Similarly, you see, that there is a key scheduling algorithm also sketched out here, which takes, which details that initially you will find, that from the specifications, DES essentially has 64 bits. By out of them 8 bits are kept for parity, so if you drop those 8 parity bits, you are remained with 56 bits and they are essentially used for a cipher. So, therefore, DES, although it has 64 bit of key, the security at DES produces or provides is  $2^{56}$ . That is the plain from the designer's point of view.

Therefore, you divide a key into 2 parts, that is, 28 parts each and then you shift them, that is, shift means, you do is a circular left shift and then, what you do is, that you obtain another 28 bit output and 28 output here and you pass that through a complex function and you obtain a 48 bit output. So, from 56 bits you obtain 48 bits, which means you drop some bits. How many bits do you drop? You drop 8 bits and you obtain a round key and that essentially, you, serve as the round key for the corresponding feistel round. And similarly, you see, that the 28 and the 28, 56 bit key is passed to the next round and a similar transformation takes place in the next round also, to obtain the next round key.

So, this is essentially the how, how the key scheduling algorithm of DES works, but there are some problems with the DES key scheduling and that we will see in our class. So, there is something called weak keys and sub semi-weak keys, which essentially leads to certain kind of properties in the DES cipher, which can be exploited for attacks.

(Refer Slide Time: 09:52)

**DES Expansion**

- **Input 32 bits**  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
- **Output 48 bits**  
31 0 1 2 3 4 3 4 5 6 7 8  
7 8 9 10 11 12 11 12 13 14 15 16  
15 16 17 18 19 20 19 20 21 22 23 24  
23 24 25 26 27 28 27 28 29 30 31 0

 NPTEL

So, therefore, this is just to mention or for completeness I mention, that this is how the DES expansion looks like. So, you see that we have already seen how to read these tables. Therefore, so you see, that input is 32 bits and output is 48 bits. So, this you can see from here, that the input is 32 bits and your output is 48 bits.

So, how does the expansion, they will look like? You see, that here, if you number your bits from 0 to 31, that is, our input bits from 0 to 31, this is how your output table looks like, so, which means, that your 1st bit of the output is derived from the 31st bit of the input. The 2nd bit of the output is derived from the 0th bit of the input and so on. You can read this table, so this is how the table looks like. So, therefore, since there is an expansion, what you essentially expect is that there should be some repetitions.

For example, you see, there is a 31 here and 31 here. Similarly, you will find other repetitions also. So, therefore, that is an, in order to do an expansion table, you must have repetitions.

(Refer Slide Time: 10:56)

### DES S-box (Substitution Box)


- 8 “substitution boxes” or S-boxes
- Each S-box maps 6 bits to 4 bits
- S-box number 1

input bits (0,5)  
↓

input bits (1,2,3,4)

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
00	1111	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0111	0010	1011	1110	1001	0111	0011	1010	0101	0000	0111
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

For other tables refer to Stinson's Book



This is how the substitution box or S-box looks like. So, as I told you, that there are 8 S-boxes in DES round, so which means, if I numbered them from S-box and all of them are distinct, so if I number them from S-box 1 to S-box 8, I have got 8 distinct S-boxes and this is how the, so 1 table of DES S-box looks like. So, I do not think it is very much readable, but just, just 1 point to be noted out here, these values, essentially, vary beyond mean match at this point.

So, essentially, what you see is that each S-box maps 6 bits to 4 bit output. So, therefore, if, when you are taking these 6 bits, these 6 bits, essentially, serve as a look up or serve as an index to look up in this table. So, therefore, what you do is that if you number your input, 6 box, 6 bits from 0 to 5, then what you do is that the 1st and the last bit, that is, the 0th bit and the 5th bit serve as the row index and the remaining 4 bits serve as your corresponding column index, and the corresponding output serves as the S-box output. So, therefore, for other tables you can refer to **Stinson's** book.



(Refer Slide Time: 12:11)


**S-Box with Table entries in decimal**

Output=13

$S_1$															
14	4	13	1	11	8	3	10	6	12	5	9	0	7		
0	15	7	4	2	13	1	10	6	12	11	9	5	3	8	
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

What is the output if input is 101000?

Row=10=2      Column=0100=4



So, this is essentially, when you have encoded the  $S_1$  inputs as decimal values. So, you see, that what we can observe here, that what I told you in the last slide, that what is the output if the input is 1 0 1 triple 0? So, what is the corresponding output if this is the input? So, you see, that your row is 1 0, why? Because I am taking this bit and I am taking the last bit, so you have got 1 0 here, so that is, that is, the 2nd and the row 2 and if your column is 0 1 0 0 because that is the remaining part that works out to 4.

So, therefore, you will observe this bit, so that means 0, 1, 2 and 3 so that is a 2nd row, row is 2.

Now, I am just observing the corresponding row, so this is 0, 1, 2 and 3. So, I have taken the 2nd row, so I am numbering my rows from 0, so this is the 0th row, this is the 1st row, this is the 2nd row and this is the 3rd row, so I take the 2nd row because my row number is 2 and my corresponding column number is 4th, so I take.


No audio 13:24 – 13:30

This is correct, so I take the 4th row, so that is 0, 1, 2, 3 and 4 and therefore, my corresponding number is 13. So, that is the way how all other look up works.

(Refer Slide Time: 13:48)

### Properties of the S-Box

- There are several properties
- We highlight some:
  - The rows are permutations
  - The inputs are a non-linear combination of the inputs
  - Change one bit of the input, and half of the output bits change (**Avalanche Effect**)
  - Each output bit is dependent on all the input bits



So, there are certain properties actually, how the S-box, on which the S-box design is based. So you can see certain properties.

(Refer Slide Time: 14:00)


### S-Box with Table entries in decimal

Output=13

$S_1$															
14	4	13	1			11	8	3	10	6	12	5	9	0	7
0	15	7	4	1		2	13	1	10	6	12	11	9	5	3
4	1	14	8	13		6	2	11	15	12	9	7	3	10	5
15	12	8	2	4		9	1	7	5	11	3	14	10	0	6

What is the output if input is 101000?

Row=10=2      Column=0100=4



See, for example, since you know, that this is your compression function, that is, S-box is the compression function, it automatically implies, that there are some amount, there are some repetition of value here. Also, it is not a permutation, so you can see that if you see the entries these are not permutations, so there are some repetitions. For example, if you see a 14 here, you will find 14 here also. So, there are certain properties, which we

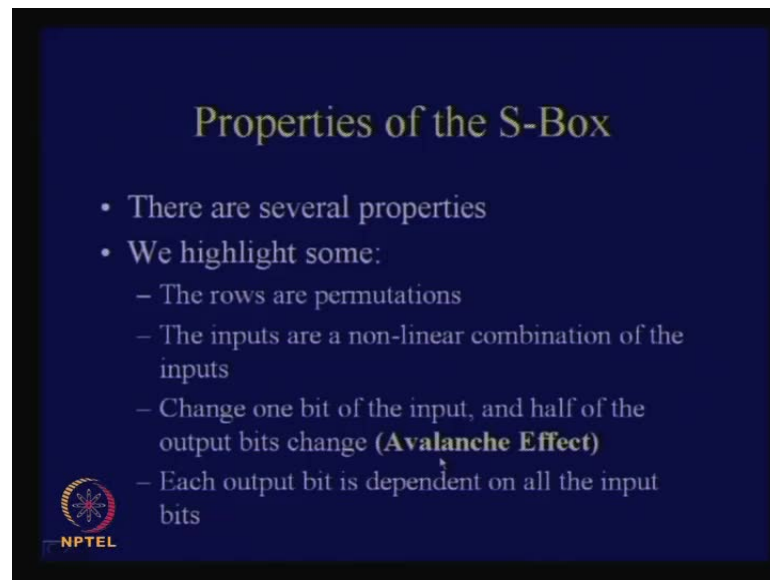
can observe if you observed these S-box elements quite minutely, closely. So, for example, if we observe 1 thing, that in each row, each row is essentially a permutation of values, that is, if you take the numbers from 0 to 15, then you will see that each of the rows is nothing but the permutation of the numbers from 0 to 15. So, there are no repetitions in each row, do you see that? Each row has got essentially nothing but repetition of, so therefore, there are other no repetitions, each row is a permutation of the numbers from 0 to 15.

So, another thing, which you observe is that if you find, that there are some repetitions in the entire S-box table, you will find, that the number of repetitions is essentially also regular. That means, each number occurs for the same number of time, that is, there is no element, which occurs for 3 times; another element, which occurs for 4 times; another element, which occurs for say, 1 time, so on, like that.

So, therefore, you will find, that each element occurs actually for  $2^6$  divided by  $2^4$ , that is,  $2^2$ , which means, it is 4. Therefore, each element, you will find, occurs for 4 times, that is a, there is a 4 here, there is a 4 here, there is a 4 here and there is a 4 here, why in a, because each row is a just a permutation.


So, therefore, you see that if you maintain this property, that is, each row is essentially a number or permutation of the numbers from 0 to 15, then there is certain amount of regularities in the entire S-box's structure. So, therefore, this is just one glance at the fact or glimpse at the fact, that if, that the S-box essentially, should maintain certain kind of properties through which your mapping becomes as regular as possible. There are other properties as well, so we will see some of them in our future classes.

(Refer Slide Time: 16:08)



**Properties of the S-Box**

- There are several properties
- We highlight some:
  - The rows are permutations
  - The inputs are a non-linear combination of the inputs
  - Change one bit of the input, and half of the output bits change (**Avalanche Effect**)
  - Each output bit is dependent on all the input bits

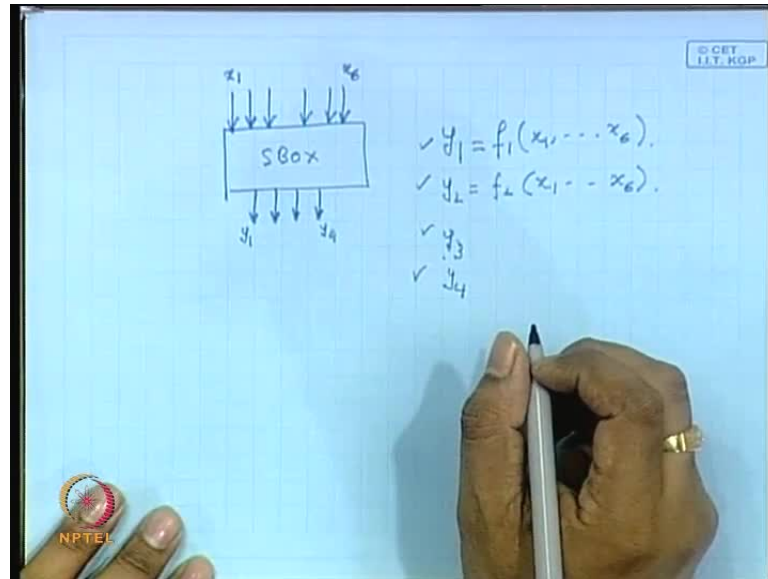
 NPTEL

So, inputs are non-linear combination of the, so the outputs actually are a non-linear combination of the inputs and if you change 1 bit of the input and in half of the output, bit changes, so what is the property known as, is called as Avalanche Effect. So, we have seen some amount of Avalanche in our last day's construction.

And each output bit is dependent on all the input bits, so that is also very important, that each output bit, you will find is actually dependent on all the input bits. So, if you observe this point, let us I mean, just I would like to explain this, so as I told you in your last day's class, that what I want is that each of the S-box output bits, essentially, should not be linear functions. So, they should be non-linear function of the input bits.

Another point, which I am emphasizing here, is that not only each of the output bit should be a non-linear function, but also the linear combinations of the output bits also should be non-linear. So, what I am telling is that suppose, you have got 4 output bits of your DES, so each of the bits are, for example, non-linear, but if I take an XOR of some of, any of the 2 bits, if you get a linear expression in terms of the input, then that is also not desirable.

(Refer Slide Time: 17:35)




So, which means, what I am saying is that if you take an S-box like this and if you observe the corresponding inputs, so there are 6 inputs here and there are 4 output bits. So, therefore, if you numbered them from  $x_1$  to say  $x_6$  and you have got  $y_1$  to  $y_4$ , then as I told you in the last day's class, that  $y_1$ , essentially, should be a non-linear function of terms from  $x_1$  to  $x_6$ . Similarly,  $y_2$  also should be a non-linear term in terms of non-linear function, in terms of  $x_1$  to  $x_6$ ; similarly,  $y_3$  and  $y_4$  also. That means, each of these outputs are non-linear, but another point to be understood from here is that, or rather observed is that not only  $y_1$  and  $y_2$  are non-linear, but also, if you XOR  $y_1$  and  $y_2$ , even then, they should be non-linear.

So, if you, for example, you take any 2 bits and you will find that when you are taking an XOR and if it works out to a linear function, then that is not desirable. So, that means, not only the each of the output bits, but also, their linear combination also should be a non-linear function.

(Refer Slide Time: 19:03)

### Properties of the S-Box

- There are several properties
- We highlight some:
  - The rows are permutations
  - The inputs are a non-linear combination of the inputs
  - Change one bit of the input, and half of the output bits change (**Avalanche Effect**)
  - Each output bit is dependent on all the input bits




So, that explains this point, that is, the row that is an input or non-linear combination, that is, the output, actually this should be, this should be the output, therefore the outputs are actually a, now, so just replace this by output, the output are the non-linear combinations of the input. So, therefore, they essentially should be non-linear. But not only the output, but also if you take linear combination of the output, they should be also non-linear, that is what I would like to highlight some of the properties of the S-box.

(Refer Slide Time: 19:32)

### DES P-box (Permutation Box)

- Input 32 bits  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
- Output 32 bits  
15 6 19 20 28 11 27 16 0 14 22 25 4 17 30 9  
1 7 23 13 31 26 2 8 18 12 29 5 21 10 3 24



So, another point, which we observe, so therefore, the next thing, that we have seen is that DES P-box or the permutation box. So, this is the straight box that we have seen; so, this is also quite easy to read. If you just numbered your input from 0 to 31, your output are just reorganization of this number, it is just rearrangement of these terms; therefore, you obtain your 32 bit output series.

(Refer Slide Time: 19:57)

### DES Subkey

- Input key size: 64 bits, of which 8 are parity bits.
- 56 bit DES key, 0,1,2,...,55
- Left half key bits, LK
 

49	42	35	28	21	14	7
0	50	43	36	29	22	15
8	1	51	44	37	30	23
16	9	2	52	45	38	31
- Right half key bits, RK
 

55	48	41	34	27	20	13
6	54	47	40	33	26	19
12	5	53	46	39	32	25
18	11	4	24	17	10	3

NPTEL

Now, let us just observe the DES subkey of the DES key scheduling also closely. So, you will see, that your input key size, which is of 64 bit keys, as I told you, but out of them 8 bit are actually parity bits, so that is a methodology through which you drop those parity bits. That means, you take a 64 bit key and from there you produce a 56 bit key, which means, that you also have parity, parity drop table kind of them.

So, there is a technique or methodology through which you drop those parity bits. It means, that if you take 64 bits and from there you drop 8 corresponding parity bits and then what you obtain is a 56 bit output or a 56 bit DES key.

So, if you number this from 0 to 55 and this is the way how you obtain them, so therefore, as I told you, that there is a left half at the key bit and there is a right half of the key bit as well. So, what you see is that in initial part, in an input or the left part, you have got a left block and there is a right block, so if you go back to the original diagram, this is how the diagram look like, that there is a left part and there is right part to a key, that is, the printed part 28 bit here and there is a 28 bit here.

So, what you do is initially, that is, therefore, you have to take a 56 bit key and you have to split that into 28 bits block and this is the way how you split that. Therefore, coming to this slide, you see, that what you do is that you do not take the left 28 bits and the right 28 bits, but there is actually a certain order through which you take them. Like for example, I take the 49 bit, I take the 40 second bit, I take the 35th bit and so I basically take, if you see the 1 2 3 4 5 6 and 7 into 4, I take 28 bits from my original key, I mean, the right part. I also take 28, the other 28 bit keys and put them into the right block, that is, my left half and this is my right half.

(Refer Slide Time: 22:05)

**DES Subkey**


- For rounds  $i=1, 2, \dots, n$ 
  - Let  $LK = (LK \text{ circular shift left by } r_i)$
  - Let  $RK = (RK \text{ circular shift left by } r_i)$
  - Left half of subkey  $K_i$  is of 24 bits
 

```

13 16 10 23 0 4 2 27 14 5 20 9
22 18 11 3 25 7 15 6 26 19 12 1
          
```
  - Right half of subkey  $K_i$  is 24 bits
 

```

12 23 2 8 18 26 1 11 22 16 4 19
15 20 10 27 5 24 17 13 21 7 0 3
          
```

 NPTEL

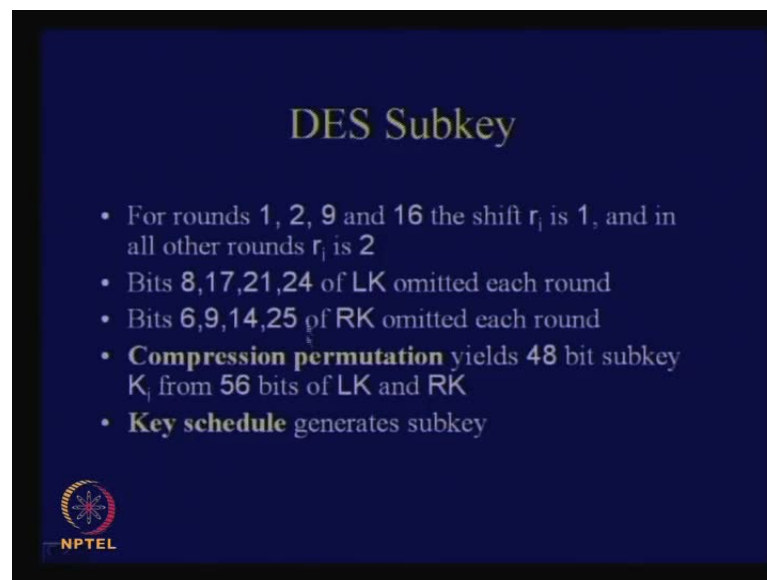
And then, what you have to do, you have to do a shift, that is, you have to do a circular left shift. So, what you do is that for rounds, I equal to running from 1 to n, you do a left circular shift by, say,  $r_i$  steps of the left block. And similarly, you do a circular shift or circular left shift of your right block as well and that is also by  $r_i$  steps. For the time being, let us ignore  $r_i$ , but  $r_i$  is actually either 1 or 2, that is, you either shift them by 1 step or you shift them by 2 steps.

So, then, you obtain essentially, what you obtain in the input is also 28 bits and also in left block is also 28 bits and right block is also 28 bits, and then, what you do is that you take the left half of the subkey  $K_i$ , which is of 24 bits. So, because you know, that from there you choose and similarly, from the right half and the subkey also, you have to choose 24 bits.



So, that means, you have to drop some bits, so therefore you drop some bits of the left part and you drop some bits of the right part and therefore, you obtain 24 bit plus 24 that is 48 bits. So, therefore, you obtain a 48 bits subkey through this technique from a 56 bit input key.

(Refer Slide Time: 23:13)



So, for rounds 1 2 9 and 16, I will tell you, that the shift  $r_i$  is 1 and for all other cases, the value of  $r_i$  is actually 2. And when you are shifting or when you are dropping those bits, that is, from 28 bits you are basically taking only 24 bits that means, you are dropping how many bits, 4 bits. So, in the left part, 8, 17, 21 and 24 bits are dropped and in the right part 6, 9, 14, 25 bits are dropped. So, and the compression function yields 48 bits subkey  $K_i$  from 56 bits of LK and RK. And therefore, this is how the key schedule works and it generates the corresponding subkey.

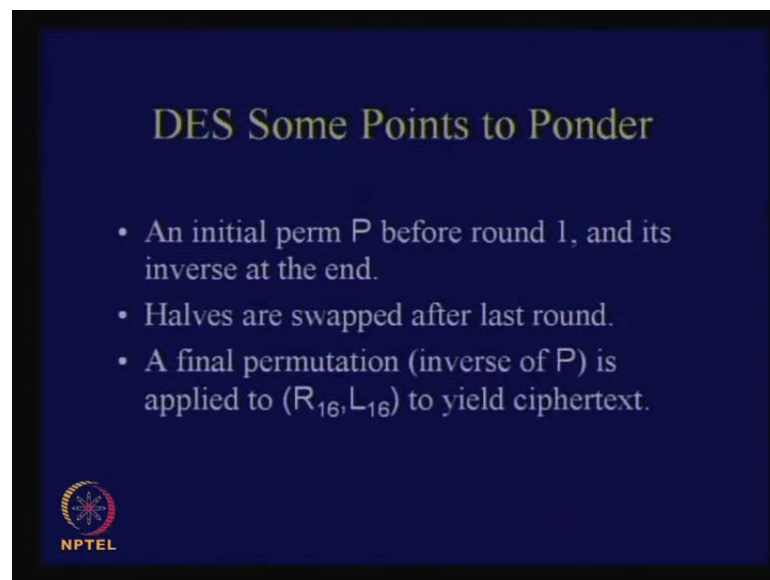
So, you see that there is some amount of history, how the design specification of DES was provided. So, it is kind of, sort of, monotonous, that is, you are basically told to do certain thing, drops of bits, but exactly the rationale behind design is not provided.

So, this was the time essentially, when cipher designing was essentially close to job. It was more of, kind of, a classified thing and it was not so much open to the academic world, like now we evaluate ciphers. So, those times, essentially, only a few people essentially used to understand, what is the, how a cipher should be designed, what should be the specifications behind the cipher. So, things have changed right now, so therefore

we know much better essentially, and why certain steps are there. And therefore, DES or the design of DES is probably not a very good design to motivate a student to work in this field.

So, therefore, but the point is, that what happens is, this is actually a very good cipher and so, once you get actually a little bit of into the cipher and start to understand the principle there, actually these, these things or if not the entire thing, but at least a significant part of this starts to make sense. So, therefore, you have to bear at this point and just believe, that this is the way a cipher may be, should be constructed and then go ahead and may be, you can look back at this point.

(Refer Slide Time: 25:14)



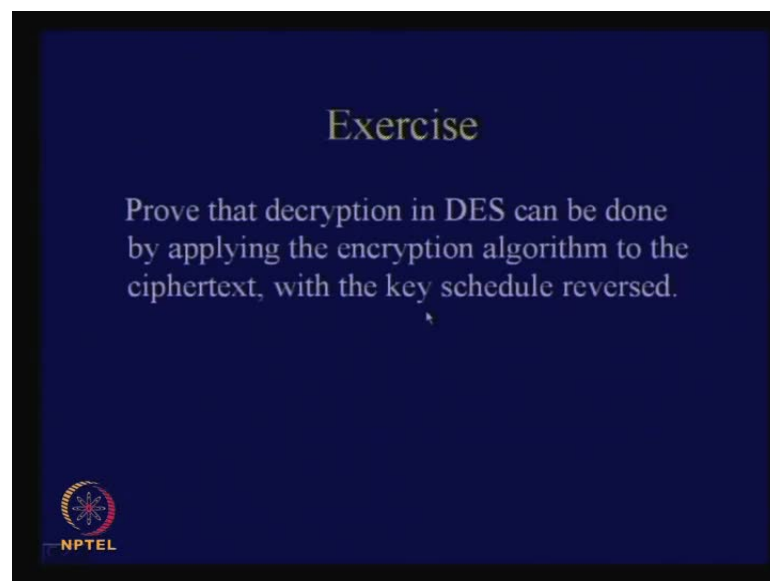
So, but certain points we can actually sort of reflect at this point, like why is the initial permutation P before round I and its inverse at the end. So, why is that provided? It does not add anything to the security, you can understand, that even a plaintext, I can obtain the permutation; even a ciphertext, I can obtain the inverse permutation; so, why is it provided?

Another point is that you will find, that in the last, may be I have not told you, that all the 16 rounds are identical. But before you do the final inverse of the permutation, you do another swap and then do an initial permutation. So, why is that initial swap provided, that is also another kind of, sort of, question that, that can arise in our minds.

The 3rd point is that a final permutation or inverse of P is applied to R 16 and L 16, so that, I already told you, that is the initial permutation and there is also final permutation. So, you see that it is not L 16 comma R 16 but it is R 16 comma L 16, why? Because there is a final swap, so where is the swap provided, that is also another question.

So, if you think of these questions, I believe, that the 1st point or the last point really does not have any significance from the point of cryptography. It may be, probably implementation become harder and may be the designers of DES did not want a very easy implementation at that point, so that is I just guessed. And the 2nd point actually is, you can actually understand, why it was given, so that the halves are swapped after last round, may be there is a reason for that and if you understand why it so, then you should be able to solve this exercise.

(Refer Slide Time: 26:50)

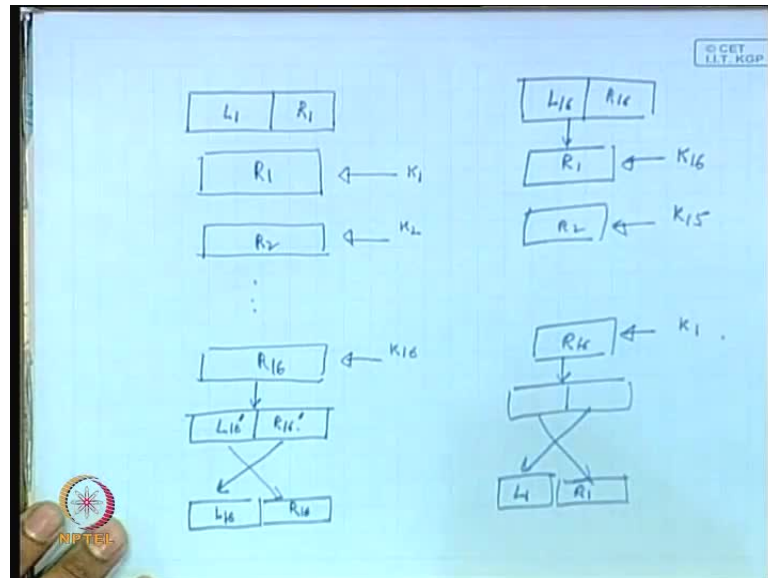


That is, prove that decryption in DES can be done by applying the encryption algorithm to the ciphertext with the key schedule reversed. So, that means, the, if you read the question carefully it says, that you have to prove, that decryption in DES can be done.

So, therefore, it can do, in order to do a decryption in DES, what you have to do is that you have to apply the encryption algorithm only. So, you can use the same algorithm, which you use for encryption, only thing is that the key schedule has to be reversed. That means that if you are generating  $K_1$  to  $K_{16}$  to do the encryption, in order to do the

decryption you have to do K 16 into K 1, that is, just invert the key scheduling. So, why is that? So, that is actually not so trivial, but may be you can just think on that.

(Refer Slide Time: 27:40)



So, therefore, if I take the input of DES, then what you can do is that you can take the input, means, essentially, you know, that there is L 1 part and there is an R 1 part. So, there is a left block and the right block and then, you do certain operations, that is, there are some rounds. So, therefore, for example, consider there is a round 1, what is the input to that key, it is K 1. Similarly, there is a round 2 and the input to that is K 2. And similarly, there are some steps and you obtain K 16 here and you obtain the corresponding ciphertext, which I denote them to be, say, L 16 comma R 16. So, you know that as I told you, that let us mark this by L 16 dash and R 16 dash, why, and because this is how all the rounds are essentially symmetrical, but what you do is that after that you do a swap. So, this means, that you essentially mark this as L 16 and this as R 16, if I denote this in the fashion.

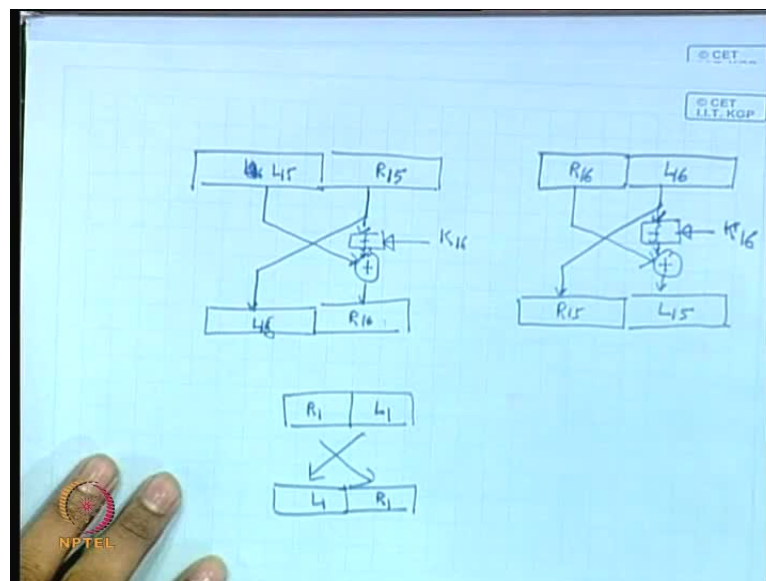
So, now, in order to do the decryption, what you can do is that you can use the same encryption rounds from R 1 to R 16 and still do the decryption. So, what is the advantage if you can, if you are able to do that, if you have a hardware, which can do the encryption? You can use the same hardware to do the decryption; also, you do not require the redesign of your decryption hardware.

So, that is quite an interesting and sort of, convenient feature in a DES algorithm. So, what you do here is that **you take...** Now this L 16 and R 16 and you start feeding to the same round, that is, start feeding to the R 1, R 2 and so on until R 16 and again you obtain certain output and do a corresponding swap there.

So, you note, that I have actually dropped out the initial permutation and the final permutation because they are not so significant, they are just transposition, I can drop them and still my discussions holds. So, now, the only thing, which should be here is that these keys, you inverse them. So, we start with K 16, K 15 and end up with K 1.

So, why do you think, that you even get back L 1 and R 1? So, that is the question, which I am asking so that you can follow if you observe the feistel block. I just work 1 block and leave the others to you as an exercise.

(Refer Slide Time: 30:20)



So, for example, you can take this, that is, L 16 and R 16. So, you note, that since I have taken here L 16 and R 16 and there is a swap actually, so, which means, that this is R 16 and this is L 16. So, therefore, what you have feeding when you are doing a decryption, is actually you are feeding R 16 and L 16. Therefore, observe first, the 1st, the encryption thing, so therefore, you have got L 15 and R 15 and what you do is that **you have got...**

So, if you have normal feistel block, what you will obtain? You obtain here, these as L 16 and this is your R 16, so what, how do you obtain R 16? You take this and XOR that with the output of the function F, which operates on R 15 and the corresponding key, which is K 16.

So, when you are doing a decryption, what you are doing is you are just taking R 16 and L 16 and let us do the same procedure. So, what you obtain here, this comes out as the left block and what about the right block, you XOR that, you XOR R 16 with the output of the function F, which works on L 16 and the corresponding K 16. So, what do you observe here is that do I obtain R 15 and L 15?

So, you see that R 15 and L 16 are same from this equation, and what about L 15? L 15 was the XOR of function F when it operated upon R 15, and what is R 15 equal to? R 15 is equal to L 16. So, here also, you see that the function F is operating on L 16. So, therefore, you see that you still obtain R 15 and L 15. Do you all of us see that?

No audio 32:27 – 32:32.

So, therefore, your L 15, therefore you see, that if you give R 16 and L 16 as an input and you obtain R 15 and L 15 as an output, so therefore, this property is an invariant show all the rounds, that means, if you give R 15 and L 15 to the next round and provide K 15 as the next key, then what happens? You obtain R 14 and L 15. So, if you keep on arguing the same fashion, the final block, which comes out of your cipher is R 1 and L 1, before the swap.

So, that means, after that we do a swap operation and you obtain back L 1 and R 1. So, I just sketch the proof and leave to you the details of an exercise; is this clear? So, that probably explains, why the final swap was presented although it does not have any security implication? It has got efficiency, **security efficiency implication**, as I told you, that efficiency is very important and that is actually one prime reason, why AES evolved.

**(C)**

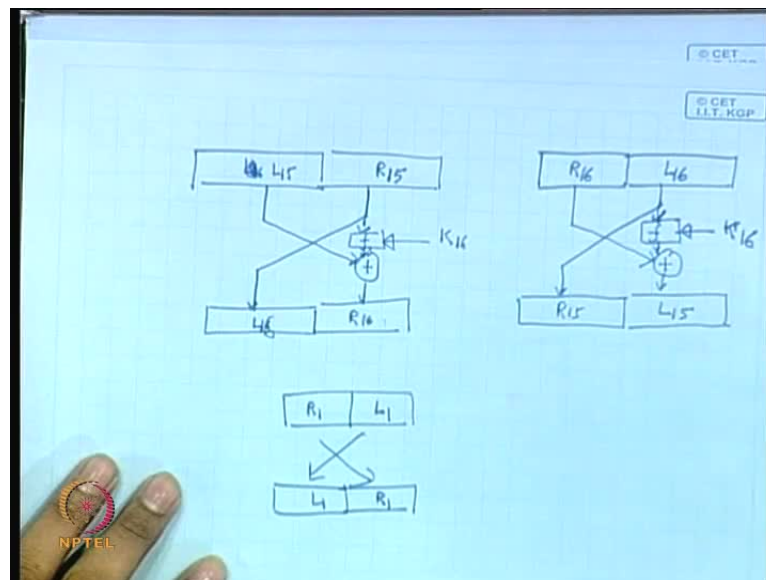
That is a very sort of pertinent question, that is, if you, the question is why, why essentially 16 rounds of DES was taken? Why not 15, why not 14? So, actually, there are certain kinds of analysis, which have been done, which are called cryptanalysis. So, there

is class of certain analysis, like differentiate cryptanalysis, linear cryptanalysis, there are many advanced cryptanalysis also through this cryptanalysis, those cryptanalysis people find, evaluate the security of the given cipher.

I will try to brief to you certain some amount of cryptanalytic techniques, but like differential and linear, for example, so, but you see, that actually 8 rounds of DES was considered to be quite secured, but another 8 rounds were essentially just kept as a, sort of, margin, security margin, that tomorrow something happens, still if something happens for 8 round today, then may be tomorrow it can happen for 9, 10 round.

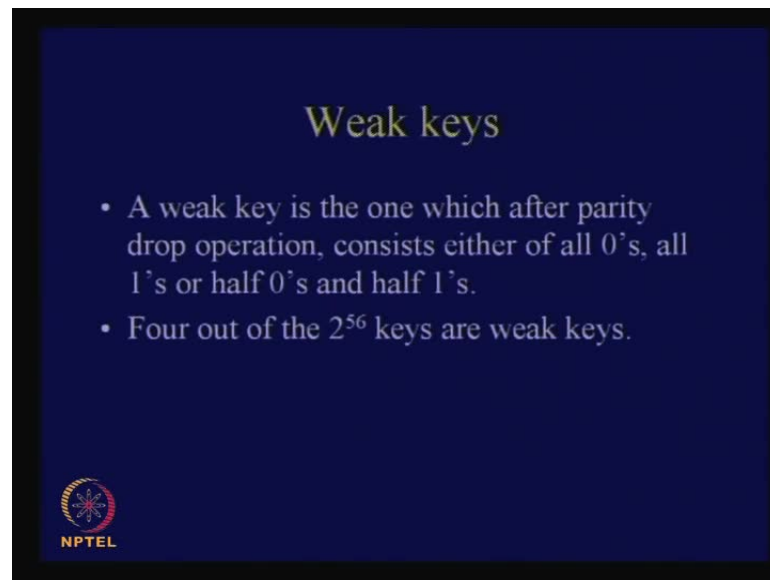
But so, therefore, in order to increase the margin, it was kept twice what we find is required. So, therefore, the thumb rule is that for linear and in order to attack your linear and differential cryptanalysis or simple cryptanalytic techniques, will be known if odd number of rounds are required, then the cipher that you proposed is essentially, typically, has got twice or number of times. So, this is just some thumb rule, so this is at least more than 2 hour essentially. So, is this part clear to us?

(Refer Slide Time: 30:20)



So, then we see some inherent weaknesses in the DES key scheduling.

(Refer Slide Time: 35:27)



So, for example, one weakness in DES key scheduling is called something, which is called weak key. So, what is the weak key? So, weak key is one in which after the parity drop operation, the key consists either all of 0s, all 1s or half 0s or half 1s.

So, therefore, you see, that it is either the entire round key is either 0 or the entire key, that you generate after dropping the parity bits, if you obtain all of them as all 0s or all of them as all 1s or half of them are 0s and half of them are 1s.


Then, essentially, there are certain properties, which appear there in the key scheduling algorithm. See, for example, 4 out of the 2 to the power of 56 keys are actually weak keys. So, therefore, out of 2 power of 56 possible keys, actually 4 keys out of them are weak keys.



(Refer Slide Time: 36:15)

### Examples of weak keys

Keys before parity drop (64 bits)	Actual key (56 bits)
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFFF FFFFFFFF




So, therefore, these are example of weak keys. Then, you see, that if you drop the parity bits, that is, 64 bits, then you obtain the actual key, that the actual 56 bit key, that you obtain is actually all 0 or half 0s, or half, and all 1s and all F's and all 0s or all 1s.

So, what is the problem with these kinds of keys? See, if we observe your key scheduling algorithm, then what you do is, essentially, just a shift operation and just a permutation. So, doing a shift over all 0s gives you only all 0s; similarly, doing a permutation also, there was a change in the value of those things. So, there is no non-linearity in the key scheduling algorithm, it is just transposition.

(Refer Slide Time: 37:02)

### Consequence of weak keys

- The round keys created from any of these weak keys are the same.
  - For example, for the first weak key, all the round keys are 0.
  - The second key leads to half 0s, and half 1s.
- If we encrypt a block with a weak key and subsequently encrypt the result with the same weak key, we get the original block.




So, therefore, if you take these keys, then you will find that there are certain consequences. The consequences are as follows, the round keys created from any of these weak keys are the same, that is, all the round keys will be the same. See, for example, for the 1st weak key, all the round keys will be 0.

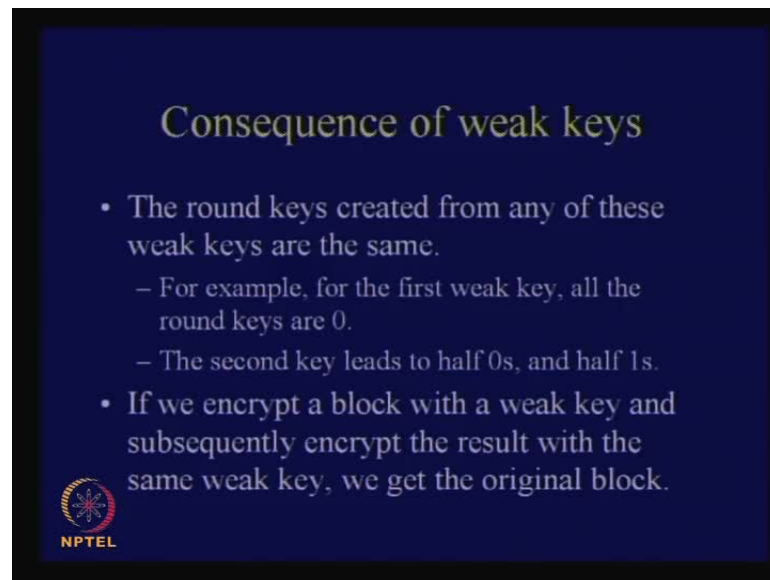
(Refer Slide Time: 37:16)

### Examples of weak keys

Keys before parity drop (64 bits)	Actual key (56 bits)
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFFF




(Refer Slide Time: 37:29)



The slide has a dark blue background with yellow text. The title 'Consequence of weak keys' is at the top. Below it are three bullet points. The first bullet point has two sub-points. The NPTEL logo is in the bottom left corner.

### Consequence of weak keys

- The round keys created from any of these weak keys are the same.
  - For example, for the first weak key, all the round keys are 0.
  - The second key leads to half 0s, and half 1s.
- If we encrypt a block with a weak key and subsequently encrypt the result with the same weak key, we get the original block.

 NPTEL

So, if you take, for example, this corresponding weak key and then run the key scheduling algorithm, then all the round keys will be 0 and if all the round keys are 0, then what is the, what is the other problem? The problem is that if we encrypt a block with a weak key and subsequently, encrypt the result with the same weak key, you get back the original block. Why? Because of the previous result. Do so you see that the inverse key scheduling gives you the same, same, same, same keys, same set of keys. Therefore, you obtain, therefore, if an attacker takes a plaintext and it does an encryption without knowing it, so you know, the known plaintext attack module, therefore we have discussed that.


You take the plaintext, you obtain the cipher text, you again encrypt that same cipher text to obtain back your original plaintext, it means, your, that the key is actually a weak key. So, that means, out of  $2^{56}$  possibilities, you are now just left with 4 possibilities.

So, that is, the problem with the DES ciphers. Is that clear?

(Refer Slide Time: 38:27)

### Semi Weak Keys

- A semi weak key creates only two different round keys and each of them is repeated eight times.
- There are six key pairs that are called semi-weak keys.
- The round keys created from each pair are the same in different order.



There is something, which is called semi weak keys also. Semi weak key, essentially, creates only 2 different round keys and each of them is repeated 8 times. So, therefore, what it does is that it creates only 2 different round keys and each of them is repeated 8 times.

So, you can see, I mean, I will show you one example, so there are 6 key pairs that are semi weak keys. Therefore, there are 6 semi weak keys occurs and the round keys created for each pair are the same, but in different order.

(Refer Slide Time: 38:56)

### Semi weak keys

First key in the pair	Second key in the pair
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E0 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1



So, therefore, what you, what I am saying is this, that is, there are 6 pairs, if you see 1, 2, 3, 4, 5 and 6, so there are 6 pairs of semi weak keys and if you take any of, see let us consider the 1st pair, like, that this key and this corresponding key and just round them through a key scheduling algorithm. Let us see, how the round keys are developed if you feed this key and if you feed this key as an input to the key scheduling algorithm. So, this is how the thing, that the propagation of the keys look like.

(Refer Slide Time: 39:23)

### A Sample round key generation


1	9153E54319BD	6EAC1ABCE642
2	6EAC1ABCE642	9153E54319BD
3	6EAC1ABCE642	9153E54319BD
4	6EAC1ABCE642	9153E54319BD
5	6EAC1ABCE642	9153E54319BD
6	6EAC1ABCE642	9153E54319BD
7	6EAC1ABCE642	9153E54319BD
8	6EAC1ABCE642	9153E54319BD
9	9153E54319BD	6EAC1ABCE642
10	9153E54319BD	6EAC1ABCE642
11	9153E54319BD	6EAC1ABCE642
12	9153E54319BD	6EAC1ABCE642
13	9153E54319BD	6EAC1ABCE642
14	9153E54319BD	6EAC1ABCE642
15	9153E54319BD	6EAC1ABCE642
16	6EAC1ABCE642	9153E54319BD

There are 8 equal round keys in each semi-weak keys.

Also, the round key in the first set is the same as the 16<sup>th</sup> key in the second set.

This means that the keys are inverses of each other.

Thus,  
 $E_{k_2}(E_{k_1}(P))=P$



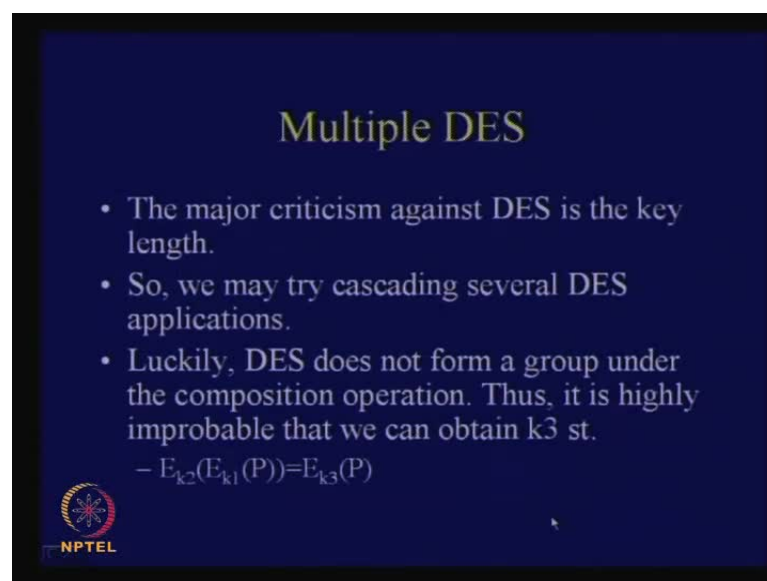
So, we will see, that for example, there is a definite ordering of the, there is a definite property in the, how the round keys are developed. You see, that there are lots of repetitions, like for example, this key was distinct, but all of them are essentially the same keys. So, there are 8 same keys and then, and, and this also generates the same keys, but only the ordering is different. So, you see that this is corresponding. Whenever this occurs here, that is, the other one occurs here and there are also 2 keys, 2 round keys, like this is one pattern and this is the other pattern, there are no other patterns.

And also, another thing, you observe is that this key scheduling is actually the inverse of this key scheduling, do you see that? There are the same keys, which has been generated here, like 9, 1, 5, something. So, we will find, that this appears here, these, so these things appear here and this is just the inverse of this corresponding key scheduling, so, which means, again you have this property, which helps.

So, therefore, if you take this key, for example, as  $E_{K1}$  and if you encrypt a plaintext and if you encrypt again using this corresponding key, you are left back bit key because of your previous property. So, you see that first of all, how all the attacker, how the attacks are getting developed? The 1st thing is that you observe the property in your cipher, which was essentially an advantage, supposedly.


Same thing you are essentially, you are able to use as the disadvantage because of certain weakness in your key scheduling algorithm. So, therefore, what you say as advantage could be quite tricky because essentially, when you are saying an advantage, it means what? There is a property in the cipher and the moment there is a property in the cipher you can explore, that is a disadvantage also. So, therefore, you have to be careful when you are designing a cipher. So, there are 8 equal round keys in each semi weak keys, so that you can see from this example also, there are round keys in the 1st set is the same as the 16th key in the 2nd set, which means, that this key scheduling and this key scheduling are inverses of each other. This means, that these keys are inverses of each other and therefore, this property holds. So, therefore, if you observe a certain kind of property like this, then that means, this could be a weak key also, and therefore, we are left with very few choices and then, you can do a brute force search (( )) and you can ascertain the actual values of the key.

(Refer Slide Time: 41:50)



**Multiple DES**

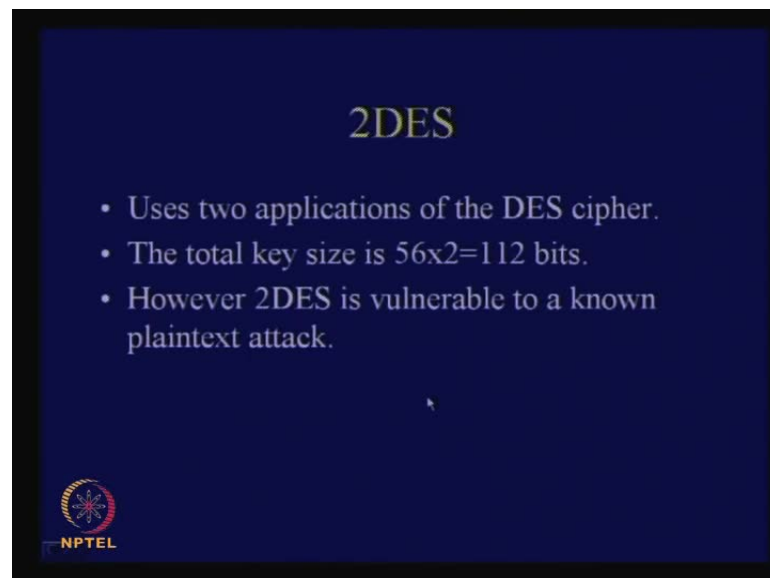
- The major criticism against DES is the key length.
- So, we may try cascading several DES applications.
- Luckily, DES does not form a group under the composition operation. Thus, it is highly improbable that we can obtain  $k_3$  st.  
–  $E_{k_2}(E_{k_1}(P))=E_{k_3}(P)$

 NPTEL

So, this we have discussed, that is, you would try to, what you would try to do is that in order to amplify a particular cipher, you will try to apply it over again and again, that is, we want to do a multiple application of the cipher. So, therefore, I would like to understand, how multiple DES works. That is, if I take 1 DES and because the size of the DES was only 56 bit keys and therefore, and since I know, that this is not an idempotent cipher, you know why? So, therefore, I take this DES cipher and I apply it twice, so therefore, what I would typically expect is my key size should increase.

So, if original DES key was 56 bit key, applying twice the DES cipher should give me 112 bits of security. So, therefore, this, essentially, it should be possible because DES does not form a group on another composition operations, so I told you briefly about that. So, thus, it is highly improbable, that we can obtain a  $K_3$ , which satisfies this equation, that is, you take plaintext  $P$  and you encrypt it by  $K_1$  and you again encrypt this by  $K_2$ , then it is very highly unlikely, that you get the 3rd key  $K_3$ . It is also 56 bit key, it satisfies this equation. So, therefore, the, it is more probable, that you should be able to amplify your security or increase your security.

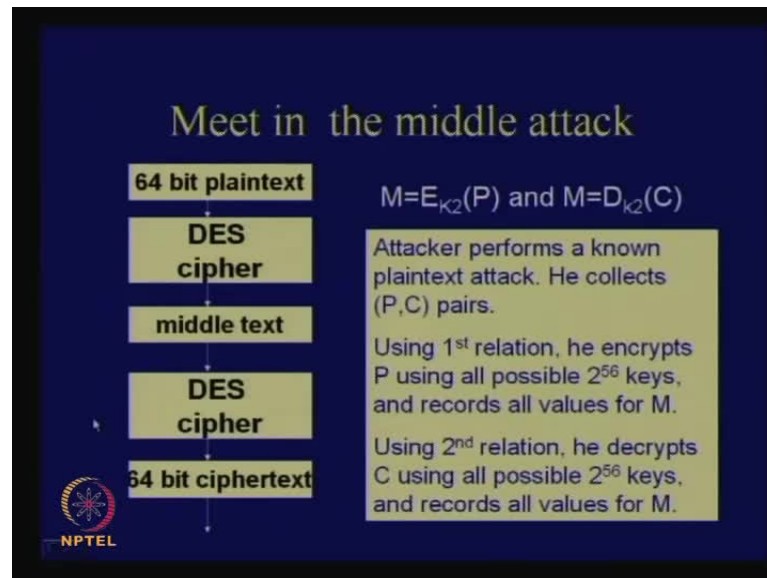
(Refer Slide Time: 43:10)



So, consider 2 DES, this is the obvious thing, that we will try, that takes DES and apply that twice. So, therefore, total key size in this case is 112 bits. However, we will see, the 2 DES is actually vulnerable to a certain kind of known plaintext cipher and this attack is

known, that meet in the middle attack and it is the very common technique for cryptanalysis. So, what is the meet in the middle attack, we will just see this.

(Refer Slide Time: 43:32)

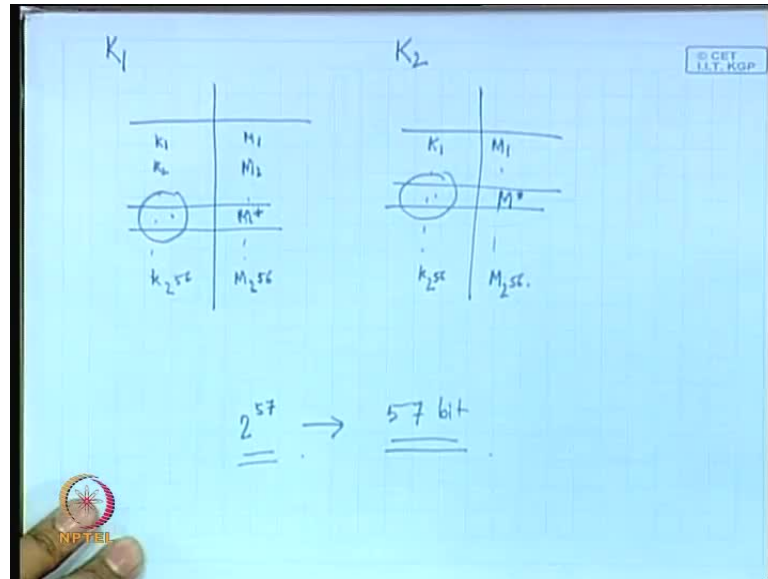


So, what you do is that you take a 64 bit plaintext, you apply the DES cipher, obtain a middle text and you again apply another DES cipher and you obtain the 64 bit cipher text. So, what is the property that is there? Therefore, you imagine that the 1st DES cipher had a key K 1 and the 2nd DES cipher had a key K 2. Then, if you denote the middle text by the letter M, then you know, that M is equal to E K2 P. That means, E K2 is one application of DES and M is also equal to D, which is the corresponding decryption function with K2, operate, operated upon the cipher text C. So, now, what the attacker does is that he does a known plaintext attack, we mean, that he obtains a large number of plaintext ciphertext pairs.

With the property, that is, he knows the values of the plaintext, he will not choose it, therefore, he just knows the value of the plaintext. So, what is there is that he collects the large number of P comma C pairs, that is, plaintext and cipher text pairs and then, what he does is that he uses the 1st relationship.



(Refer Slide Time: 45:03)



Moreover, what he does is that he encrypts the plaintext using all possible keys. So, how many possible keys are there?  $2$  power of  $56$ ; the  $2$  power of  $56$  keys and records all values for  $M$ . so, whatever values for  $M$  are there, he records them in a corresponding table, therefore you guess 1 key. Therefore, you start building a table of, in this fashion, you take a corresponding key, say  $K_1$ ,  $K_2$  and so on till  $K_2$  to the power of  $56$  key bits, and then, therefore, you obtain the corresponding values of  $M_1$ s also in each cases. Therefore,  $M$  also stands for up to  $2$  power of  $56$ . So, this we do for the forward function, similar thing we can do for the reverse thing also, for that is for the value of  $K_1$  and similar thing you can do for  $K_2$  also.

You just start guessing  $K_2$  and you know the cipher text correspondingly and therefore, what you do is that you decrypt them and you obtain them in the text. So, therefore, you again have got values running from  $K_1$  to  $K_2$  to the power of  $2$  to the power of  $56$  and you observe the value of  $M_1$  from  $M_2$  to the power of  $56$ .

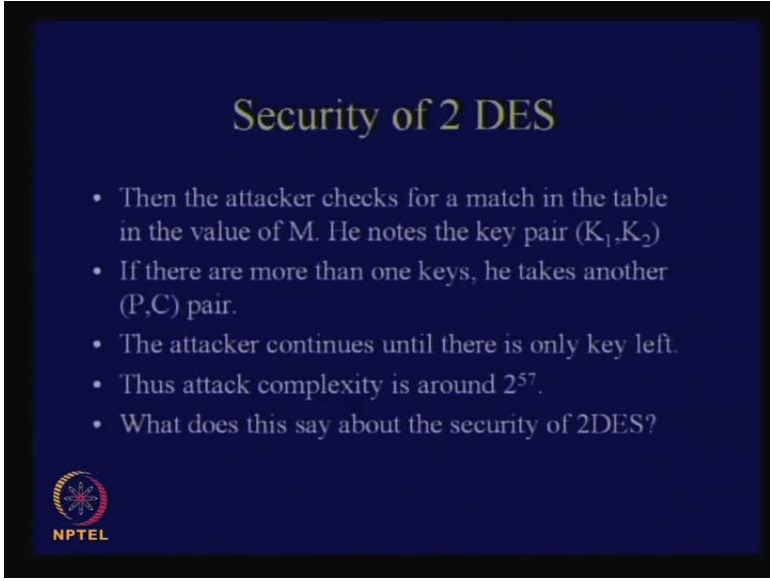
Then what you to do is that you search this table and you also search this table and you try to find out 1 entry, where actually, the  $M$  values are the same. So, you find  $M$  star here, you also observe a search for another  $M$  star here, so that means, that this key and this key could be a possible pair, actual pair.

So, therefore, the moment we find out a unique pair, you start recording that as the actual key, but what will happen often is that you are not able to find out any couple like that.

So, then, what you do is that you again start with another plaintext ciphertext pair and repeat the experiment and it is actually quite probable, that within few trials, we will find the actual values of the key.

So, in this case, what is the attack complexity, that are complexity is 2 power of 56 to maintain this table and 2 power of 56 to create this table. So, what is the total number, total size of the key? 2 to the power of 56 into 2, that is, 2 power of 57. So, roughly, with the attack complexity of 2 power of 57, you should be able to ascertain the value of  $K_1$  comma  $K_2$ . So, what is the guarantee, what is the security that you obtain here? You obtain a 57 bit key security, whereas the designers where looking for a 112 bit of security. So, therefore, 2 applications of DES is not secured, you can frame an attack, which is of a class of known plaintext attack and you can find out, that essentially, the security, that you obtain is quite less compared to what I would have liked.

(Refer Slide Time: 47:40)



The slide has a dark blue background with yellow text. The title 'Security of 2 DES' is centered at the top. Below it is a bulleted list of five points. In the bottom left corner, there is a circular logo with a red and white design, and the text 'NPTEL' below it.

### Security of 2 DES

- Then the attacker checks for a match in the table in the value of M. He notes the key pair  $(K_1, K_2)$
- If there are more than one keys, he takes another  $(P, C)$  pair.
- The attacker continues until there is only key left.
- Thus attack complexity is around  $2^{57}$ .
- What does this say about the security of 2DES?


NPTEL

So, the security of 2 DES, what you will see is that if I denote this value.

(Refer Slide Time: 47:47)

### Triple DES

- Since 2DES was a bad design, people consider 3 applications of DES.
- The first and third stages use  $K_1$  as key.
- The second stage use  $K_2$  as the key.
- Also, the middle stage uses decryption.
- Thus, setting  $K_1=K_2$  we have simple DES.


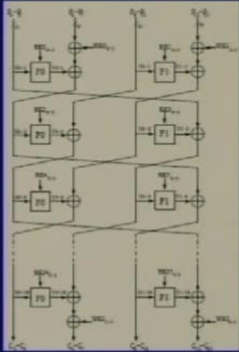


So, what I say about the security of 2 DES is that 2 DES was bad design. You can consider 3 applications of DES, the 1st and the 3rd stages uses  $K_1$  as key and therefore, people went to the triple DES design and in triple DES, what was found out is that the 1st and the 3rd stages uses  $K_1$  as the key and the 2nd stage uses  $K_2$  as the key. Also, the middle stage is actually a decryption stage. The setting  $K_1$  equal to  $K_2$ , we will essentially have the symbol DES, so therefore, people went to a triple DES design.

(Refer Slide Time: 48:20)

### Generalization of the Feistel Cipher

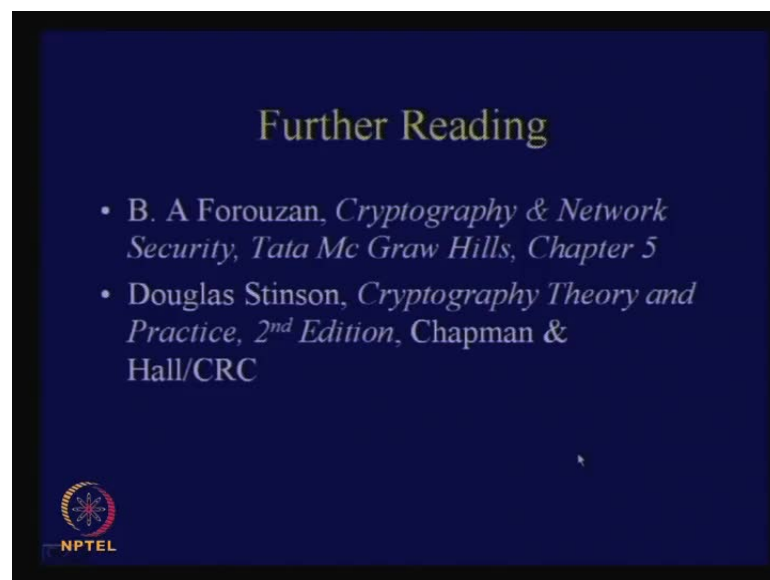
**Clefi**  
128 bit block cipher  
designed by Sony Corporation



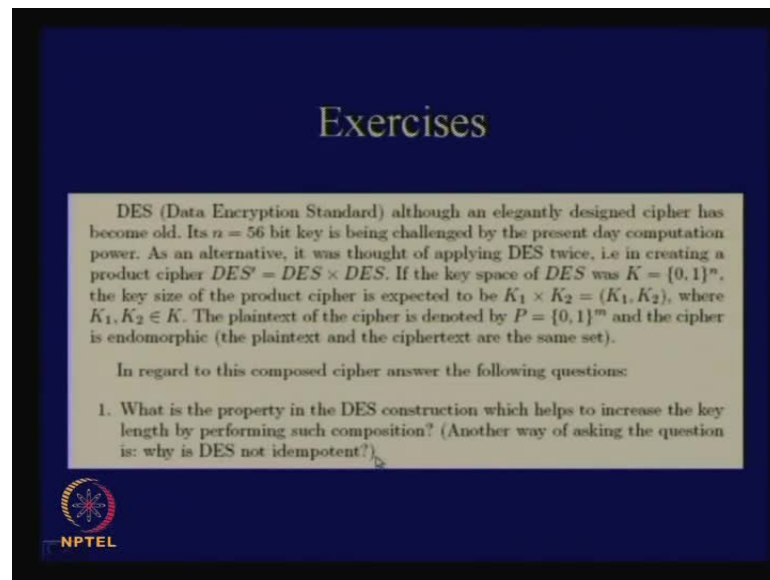
And I will just try to conclude my talk within this day's class with an idea or something, which is called a generalization of the feistel cipher. So, therefore, people have actually generalized the idea of the feistel cipher and I have made something, which is little bit more complicated than a normal feistel cipher. For example, this is an example of a Clefia cipher, which is designed by Sony Corporation. It is around, it is a 128 bit block cipher, but this has internally the same feistel structure, but only the interlining is little bit expanded over 2 branches, 2 parallel branches.

And therefore, you obtain essentially, a 6, 128 bit of blocks, block cipher. Therefore, this is an example of something, which I called, which we called as generalized feistel structure, but the properties of decryption, encryption, more or less the same.

(Refer Slide Time: 49:06)



(Refer Slide Time: 49:12)




**Exercises**

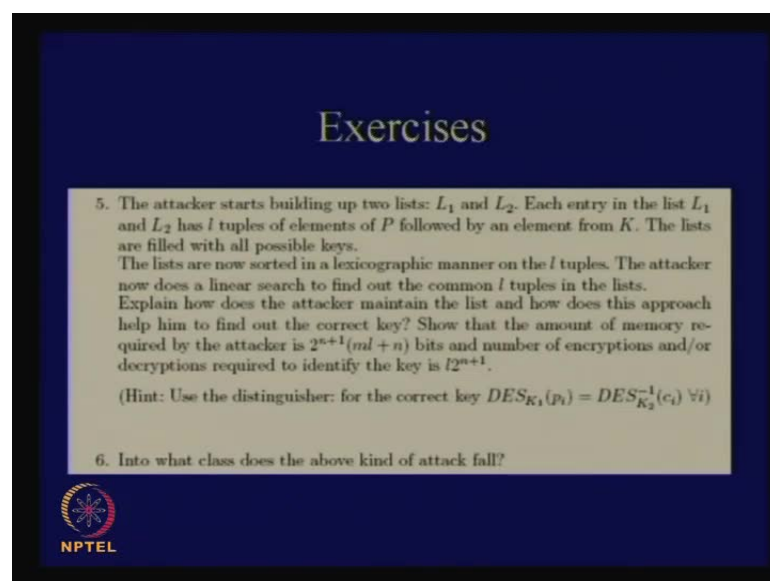
DES (Data Encryption Standard) although an elegantly designed cipher has become old. Its  $n = 56$  bit key is being challenged by the present day computation power. As an alternative, it was thought of applying DES twice, i.e. in creating a product cipher  $DES' = DES \times DES$ . If the key space of DES was  $K = \{0, 1\}^n$ , the key size of the product cipher is expected to be  $K_1 \times K_2 = (K_1, K_2)$ , where  $K_1, K_2 \in K$ . The plaintext of the cipher is denoted by  $P = \{0, 1\}^m$  and the cipher is endomorphic (the plaintext and the ciphertext are the same set).

In regard to this composed cipher answer the following questions:

1. What is the property in the DES construction which helps to increase the key length by performing such composition? (Another way of asking the question is: why is DES not idempotent?)




(Refer Slide Time: 49:38)



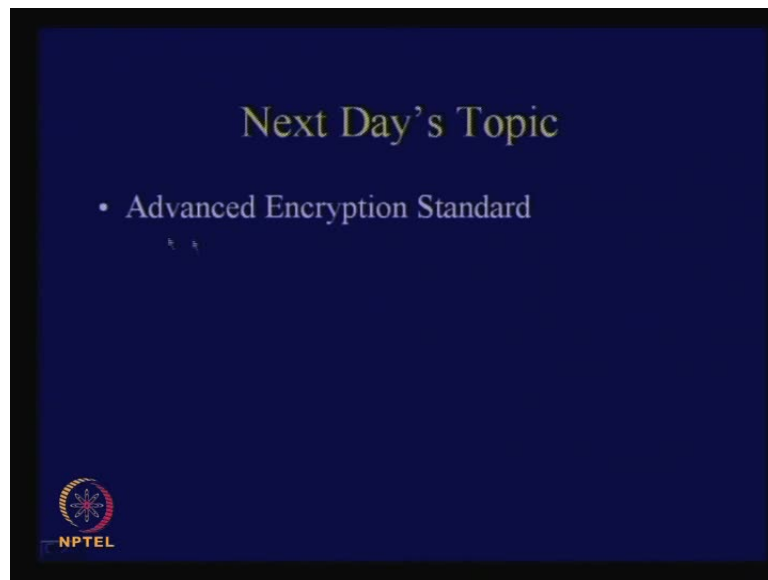
**Exercises**

5. The attacker starts building up two lists:  $L_1$  and  $L_2$ . Each entry in the list  $L_1$  and  $L_2$  has  $l$  tuples of elements of  $P$  followed by an element from  $K$ . The lists are filled with all possible keys. The lists are now sorted in a lexicographic manner on the  $l$  tuples. The attacker now does a linear search to find out the common  $l$  tuples in the lists. Explain how does the attacker maintain the list and how does this approach help him to find out the correct key? Show that the amount of memory required by the attacker is  $2^{n+1}(ml+n)$  bits and number of encryptions and/or decryptions required to identify the key is  $l2^{n+1}$ .  
(Hint: Use the distinguisher: for the correct key  $DES_{K_1}(P_i) = DES_{K_2}^{-1}(c_i) \forall i$ )
6. Into what class does the above kind of attack fall?



So, you can read further thing from the Forouzan's book and Stinson's book, but before I conclude this, as an example, which I shall put them, whether the exercises, which I shall put them online also, you can just try to work through this exercise. This should, if you are able to confidently solve this, we should clear your last day's whatever is based upon the exercise or (( )) are based upon the last day's class and today's class. So, as you can go through these problems and you can try to understand, what are the, you should, you should be able, you should be able to clear, therefore, whatever we have discussed.

(Refer Slide Time: 49:48)



So, these are essentially, this will be posted online, so you can try to solve them and our next day's class will be based on advanced encryption standard.