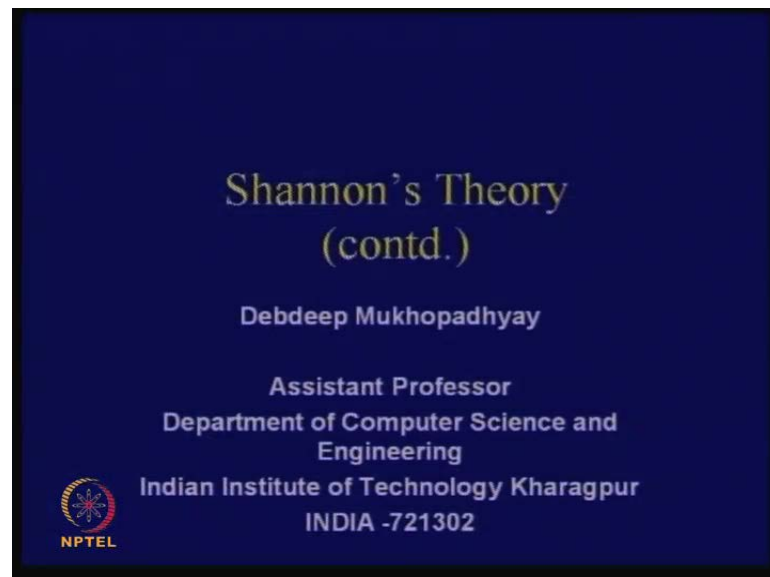


Cryptography and Network Security
Prof. D. Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

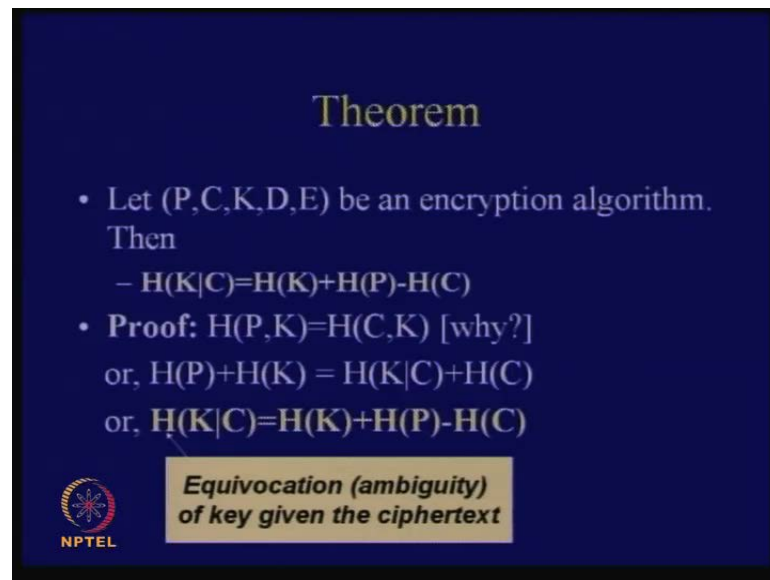
Lecture No. # 09
Shannon's Theory
(Contd.)

(Refer Slide Time: 00:27)



So, in today's talk, we will conclude the topic of Shannon's theory. So, in the previous class, if you remember, that we have concluded with the idea of equivocation of keys. So, we have defined briefly, that what, is the idea behind spurious keys. So, today we will try to understand, whether we can compute a lower bound of the spurious keys.


(Refer Slide Time: 00:47)



Theorem

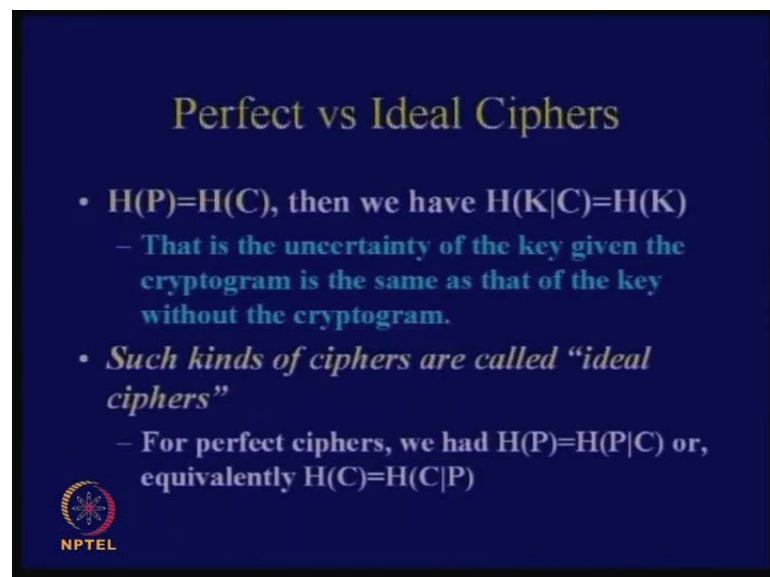
- Let (P,C,K,D,E) be an encryption algorithm.
Then
 - $H(K|C)=H(K)+H(P)-H(C)$
- **Proof:** $H(P,K)=H(C,K)$ [why?]
or, $H(P)+H(K) = H(K|C)+H(C)$
or, $H(K|C)=H(K)+H(P)-H(C)$

**Equivocation (ambiguity)
of key given the ciphertext**




So, if I just would like to recap, that we essentially proved this particular formula yesterday, that is, $H(K|C)$ given C is equal to $H(K)$ plus $H(P)$ minus $H(C)$. So, therefore, the ambiguity of a key given the ciphertext is described as follows, that it is addition of the ambiguity of the key plus the ambiguity of the plaintext subtracted with the ambiguity of the ciphertext.

(Refer Slide Time: 01:12)



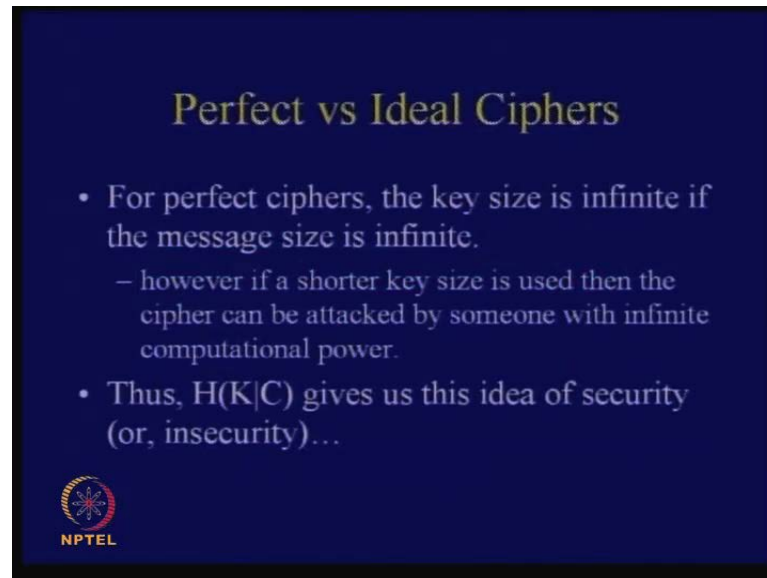
Perfect vs Ideal Ciphers

- $H(P)=H(C)$, then we have $H(K|C)=H(K)$
 - That is the uncertainty of the key given the cryptogram is the same as that of the key without the cryptogram.
- *Such kinds of ciphers are called “ideal ciphers”*
 - For perfect ciphers, we had $H(P)=H(P|C)$ or, equivalently $H(C)=H(C|P)$



So, we also discussed about what is the meaning of ideal ciphers and told, that in case of an ideal cipher, $H(K|C)$ given C is equal to the value of $H(K)$. So, that means, that the ciphertext does not leak any additional information about the key.

(Refer Slide Time: 01:31)



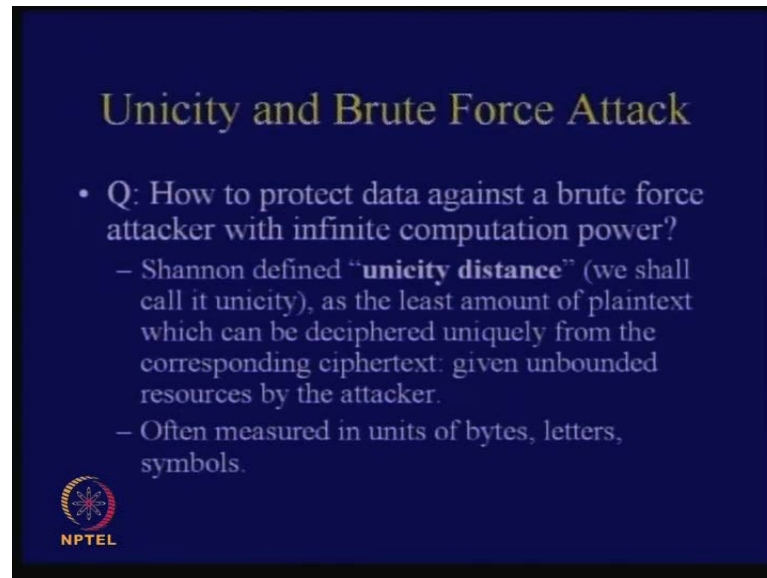
The slide has a dark blue background with yellow and white text. The title 'Perfect vs Ideal Ciphers' is at the top in yellow. Below it are two bullet points in white. The first bullet point says 'For perfect ciphers, the key size is infinite if the message size is infinite.' followed by a sub-bullet '– however if a shorter key size is used then the cipher can be attacked by someone with infinite computational power.' The second bullet point says 'Thus, $H(K|C)$ gives us this idea of security (or, insecurity)...'. In the bottom left corner is the NPTEL logo, which consists of a circular emblem with a star-like pattern and the text 'NPTEL' below it.

So, these are the things that we described yesterday and concluded with it. And so, therefore, $H(K|C)$ given C gives us the idea of security or insecurity. So, therefore, what we discussed is that even for perfect ciphers, the key size is infinite if the message size is infinite. So, that was the problem with perfect ciphers, that is, they are not practical.

So, then we defined another kind of ciphers called the ideal ciphers, where $H(K|C)$ is equal to $H(K)$. And, as I, as I described yesterday, that the main objective of our, what we study in this course, would be to find ciphers, which are secured against a bounded adversary. That means, we are essentially striving to achieve computational security; that means, security considering what is today's computational power.


So, for example, if you can prove that a given cipher has got a security, which requires an adversary to do, say 2^{80} computations, then we are fairly happy; we say, that the cipher has achieved computational security.

(Refer Slide Time: 02:37)



Unicity and Brute Force Attack

- Q: How to protect data against a brute force attacker with infinite computation power?
 - Shannon defined “**unicity distance**” (we shall call it unicity), as the least amount of plaintext which can be deciphered uniquely from the corresponding ciphertext: given unbounded resources by the attacker.
 - Often measured in units of bytes, letters, symbols.

 NPTEL

So, the question is how to protect? So, therefore, but still, I mean, in today's class we will be, essentially, still considering an unbounded adversary and essentially address this question, that is, how do I protect a data against a brute force attacker with an infinite computational power?

So, therefore, there is an attacker who has got infinite computational power and still, can I protect the cipher? So, the idea is, that is, Shannon defined a certain parameter, which is called unicity distance and we say, that it is the least amount of ciphertext, which I would like to make it available to the adversary who is an unbounded adversary, so that he does not find out the unique value of the key.

So, the strategy of the, of the adversary is as follows. What he does is that he takes the ciphertext; he assumes a key, decrypts it back and finds out the plaintext. If the plaintext is meaningful, then he notes it down. But the point is, that because of the redundancy in the English language or rather any other language, any language for that matter, the adversary will not actually find out the main, I mean, key uniquely. So, what he will essentially have is a set of keys. So, apart from the actual key, the other keys are called the spurious keys.

Sir, how will the attackers enable it, like his iterations, how will he actually decide or how will the computer decide, that the text is meaningful. It cannot check based on its own keys.

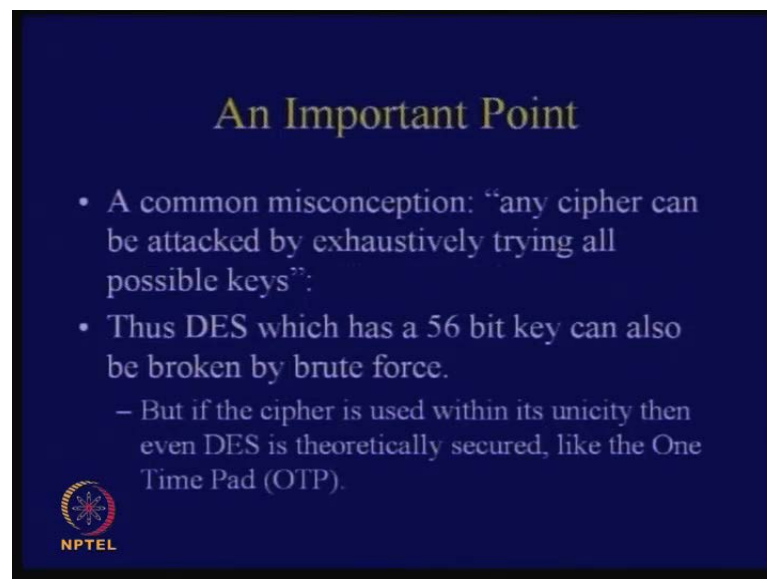
No, no, the idea is as follows, that is, you take the ciphertext; consider a shift cipher, so you have got a unique value of the key. So, you find out the value of the key. So, so what you do is, that you decrypt it back and then you check, that whether the plaintext make sense or not. So, what you are asking is that you have got a lot of work to do, but what I told you at the beginning is, that I am considering what, I am considering an unbounded adversary.

So, the adversary has got infinite computational power; he is, he is supreme. So, the idea is, that whether, the question is, that whether even given such a kind of, such a kind of adversary, how many minimum number of ciphertext will I make it available to the adversary, so that he cannot still guess the actual value of the key? So, therefore, the set of the spurious keys should be null.

Minimum number or maximum number?


Minimum number, because if I give you more information, then you can do more things, so I would like you to give the adversary minimum number of ciphertexts. Is it clear?

(Refer Slide Time: 05:09)



An Important Point

- A common misconception: “any cipher can be attacked by exhaustively trying all possible keys”;
- Thus DES which has a 56 bit key can also be broken by brute force.
 - But if the cipher is used within its unicity then even DES is theoretically secured, like the One Time Pad (OTP).

 NPTEL

So, therefore, so therefore, a common misconception is that any cipher can be attacked by exhaustively trying all possible keys, this is the very common misconception. So, what would you like to say is that for example, even if DES, it has got a 56 bit key, so this can also be broken by brute force. But the idea is, that so suppose, let us consider an

adversary who can do 2^{56} computations, so he should be able to break it. So, the idea is that, but if the cipher is used within its unicity distance, then even an all power adversary cannot break the cipher because, because of the strategy. The strategy is, therefore you take the ciphertext, you decrypt it back, check the plaintext, whether it makes sense or not. If it makes sense, you note the key and keep the key, register the key as a meaningful key or rather a possible key, but that is only one key, which is the actual key, the rest are spurious keys.

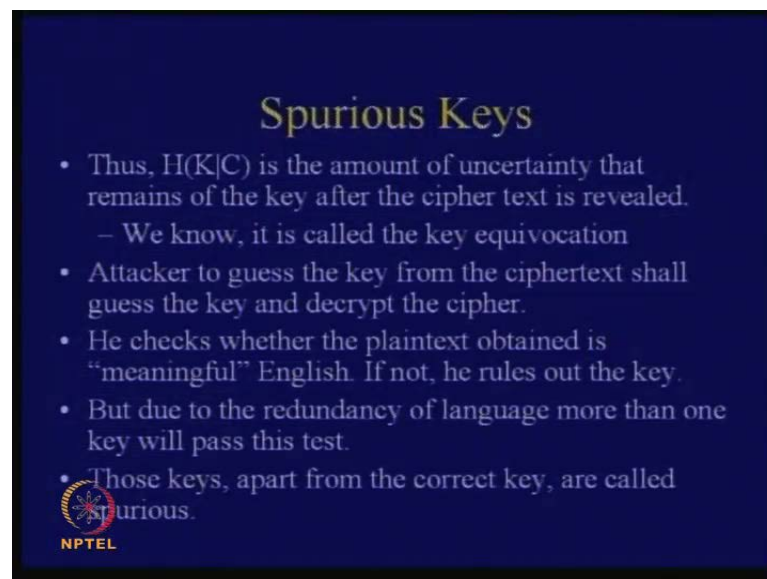
Sir what is the significance of unicity distance

Yes

the significance of unicity distance


The significance of unicity distance is that for example, if I can compute the unicity distance of say DES or any other cipher, then I would like to use the same key for so many times. After that, I have to change the key if I am considering an unbounded adversary. So, for example, if I am considering an unbounded advisory, then I would like to use the key only for, say 2 times, after that I would like to change the key if its unicity distance is 2.

(Refer Slide Time: 06:36)



Spurious Keys

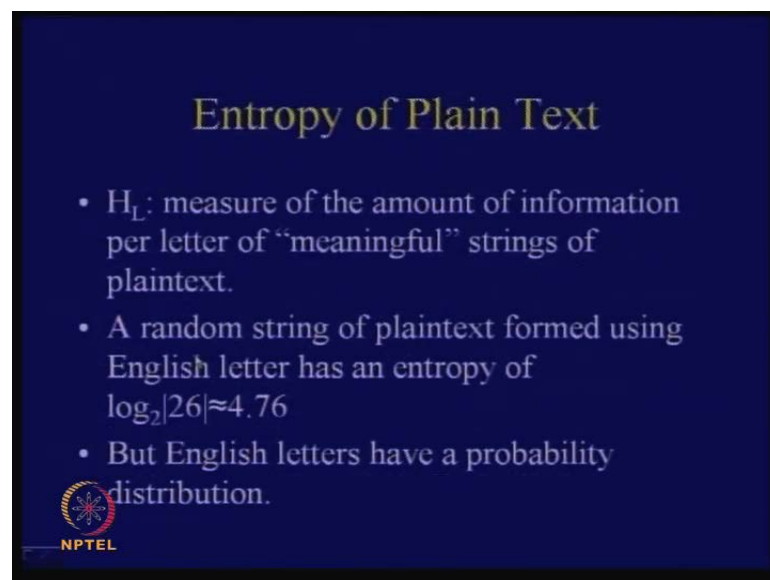
- Thus, $H(K|C)$ is the amount of uncertainty that remains of the key after the cipher text is revealed.
 - We know, it is called the key equivocation
- Attacker to guess the key from the ciphertext shall guess the key and decrypt the cipher.
- He checks whether the plaintext obtained is “meaningful” English. If not, he rules out the key.
- But due to the redundancy of language more than one key will pass this test.
- Those keys, apart from the correct key, are called spurious.

 NPTEL

So, therefore, this is what I have just described. So, H_K given C is the amount of uncertainty that remains of the key after the ciphertext is revealed. So, we know that, we know, it is called the key equivocation and have already defined that.

So, what the attacker does is that he guesses the key from the ciphertext and we shall guess the key and decrypt the cipher. So, what he does next is, he checks, whether the plaintext obtained is meaningful English or not. If not, he rules out the key, but due to the redundancy of language, more than one key will actually pass this test; those keys, apart from the correct key are called spurious.

(Refer Slide Time: 07:16)



The slide has a dark blue background with yellow text. The title 'Entropy of Plain Text' is centered at the top. Below it are three bullet points. At the bottom left is the NPTEL logo, which consists of a circular emblem with a star and the text 'NPTEL' underneath.

Entropy of Plain Text

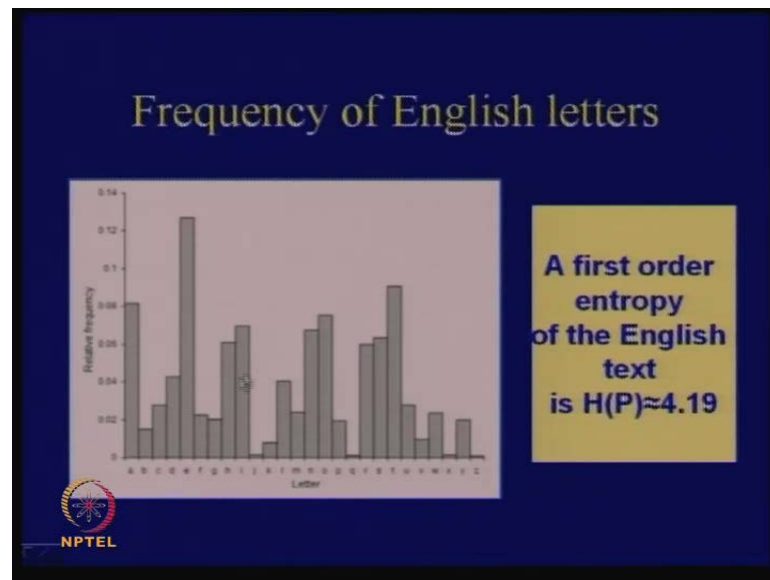
- H_L : measure of the amount of information per letter of “meaningful” strings of plaintext.
- A random string of plaintext formed using English letter has an entropy of $\log_2|26|\approx 4.76$
- But English letters have a probability distribution.

NPTEL

So, then, we come to something, which is called, we have already discussed about, what is entropy. So, we will be using this or rather we will be using this entropy to find out or find out a minimum bound of the spurious keys. So, we will try to find out a minimum bound or the lower bound of the number of spurious keys.

So, so, therefore, so, therefore, consider H_L , so therefore H_L is used to measure the amount of information per letter of meaningful strings of plaintext. So, that is the definition of H_L . It measures the amount of information per letter of meaningful strings of plaintext. So, therefore, consider random string of alphabets. So, there are 26 letters and if all of them are equally likely, then what is the entropy, is equal to $\log_2 26$ base 2 and so that works to, computes to 4.76. But English language you know, have a probability distribution; it is not 1 by 26 for all the letters.

(Refer Slide Time: 08:17)



So, this was, this is the sort of the graph, which was how the English language characters in general vary.


So, therefore, if you just 1st order entropy, that means, you just take one particular alphabet among and consider its probability and then you feed into the formula of $P \times I$, $\log P \times I$ negative and sigma, then your 1st order entropy of the English text works to 4.19. But you can do better than that. So, you know that in English language, consecutive letters essentially are not uncorrelated. For example, if I take q, then next letter is u, so they are not uncorrelated.

So, that means, that if I take 2 grams for example, this entropy or uncertainty should reduce. So, therefore, if the 1st order entropy is not enough, so what you do is that I do a 2nd order approximation.

(Refer Slide Time: 09:09)

Higher Order Approximations

- A large number of digrams are tabulated and $H(P^2)$ is computed.
- The value is divided by 2 to obtain a second order approximation, $H(P^2)/2 \approx 3.90$
- One could continue obtain trigrams, etc and compute higher order approximations for the entropy.




So, in 2nd order, what I do is that I compute all possible digrams. I find out their probabilities and subsequently their entropy and then I divide, that $H(P^2)$ by 2, so it works to 3.9. So, you see that the entropy has reduced. Similarly, you can do trigrams and again find out the $H(P^3)$ and then, again divide that by 3. And similarly, you can do higher order approximations of the entropy.

(Refer Slide Time: 09:35)

In general...

- Successive letters have correlation, which reduces the entropy.
- Define P_n to be the random variable that has a probability distribution of n-grams of plaintext
- Define H_L as the **entropy of a natural language L**:

$$H_L = \lim_{n \rightarrow \infty} \frac{H(P_n)}{n}$$


So, the idea is that in general, the successive letters have got correlation, which reduces the entropy. So, therefore, for example, define that higher order entropy as follows, that

define H_L as the entropy of a natural language, L as H_L is equal to limit n tends to infinity, H_L is equal to $\lim_{n \rightarrow \infty} \frac{H(P^n)}{n}$.

So, you find out the value of H_L , means, you find out all n grams, their probability distribution, find, compute the corresponding entropy and then, divide this by n and your limit tends, n tends to infinity. That means, this, this is an approximation for very high values of n and it has been formed experimentally, that the value of H_L , if you consider very high values of n , in general ranges between 1 and 1.5.

So, you will find that the H_L falls within, within 1 and 1.5.

(Refer Slide Time: 10:29)

The slide is titled "Redundancy" in yellow text on a dark blue background. It features a central formula in a light-colored box:
$$R_L = 1 - \frac{H_L}{\log_2 |P|}$$
 To the left of the formula, it says "Fraction of 'excess letters'". To the right, it says "Entropy of the language" and "Entropy of the random language". Below the formula, it states: "For English Language, $1 \leq H_L \leq 1.5$. Considering $H_L = 1.25$, and $|P| = 26$, $R_L \approx 0.75$." At the bottom, it says "English Language is 75% redundant." with an NPTEL logo.

So, so from there, what do we understand? We understand that there is some amount of redundancy in the language. So, in order to understand that better, let us quantize the redundancy by using these parameters R_L . R_L is defined as 1 minus H_L divided by log cardinality of the plaintext base 2.

So, you can immediately understand certain terms from this formula, for example, will find that R_L is equal to 1 minus H_L . So, therefore, if you just consider a random language, that is, if you just, if English language would have been random, then what would have been entropy of H_L ? It would have been $\log_2 26$. So, in that case, what would have been the redundancy? It would have been 0.

So, therefore, you see, that for any other value of H L, like when I am considering, say 2 gram, 3 gram, 4 gram, 5 gram and say n grams, then this value of H L was gradually reducing. So, that means, the redundancy was increasing. So, therefore, this formula is able to capture the redundancy of the language.

(Refer Slide Time: 11:44)

$$R_L = 1 - \frac{H_L}{\log_2(P)}$$

$$\frac{H_L}{\log_2(P)} = 1 - R_L$$

$$H_L = (1 - R_L) \log_2(P)$$

if $H_L \downarrow \Rightarrow R_L \uparrow$

So, for example, if for typical values, if you say, that H L lies between 1 and 1.5, so you take a value of, you take a value of, so for example, let us consider this, that is, 1 minus H L divided by log of P base 2. So, in that case, I can rearrange this as follows because I will be using this later on, so I can write, for example H L divided by log P base 2 is equal to 1 minus r l. So, therefore, H L is equal to 1 minus R L into log of cardinality of P base 2.

So, we will just note this equivalent relation because we will be requiring this result. Subsequently, so you note one fact, that if this value of H L, that is, if H L reduces, then the corresponding, this implies, that the corresponding value of R L increase. So, therefore, if the entropy reduces of the corresponding language, that is equivalent to saying, that the redundancy of the language has increased. So, therefore, this formula is able to capture this intuitive result.

So, let us therefore, so let us consider what is the corresponding redundancy of an English language? So, you will like to quantize that. So, if you find out, you will find out, that H L lies between 1 and 1.5. So, consider, that H L is equal to 1.25 and you

know, that the number of plaintext characters is 26. So, therefore, if you feed into this formula, your R L works to 0.75.


So, what does it mean? It means that English language is 75 percent redundant, so whatever you speak, out of that 75 percent is actually redundant. So, does it mean that out of 4 characters you talk, I can throw away 3 characters? No, it is not exactly so. So, what it means only is, that if you do for example a Hoffmann coding, then essentially you are expected to get such a kind of compression.

So, so, therefore, that is the idea of redundancy of a language and that is precisely the reason why cryptanalysis is favored. If you are the very random kind of language, then cryptanalysis would have been harder, but you know, that there is a redundancy in English language and that solves as extra information to the attacker.

(Refer Slide Time: 14:07)

A lower Bound of equivocation of key

- P^n : r.v representing n-gram plaintext
- C^n : r.v representing n-gram ciphertext
- $H(K|C^n) = H(K) + H(P^n) - H(C^n)$
 - $H(P^n) \approx nI_L$ (assuming large n)
 - $= n(1 - R_L) \log_2 |P|$
 - $H(C^n) \leq n \log_2 |C|$
- If $|P| = |C|$,
 - $H(K|C^n) \geq H(K) - nR_L \log_2 |P|$

 NPTEL

So, let us try to calculate the lower bound of equivocation of the key. So, that is the objective of today's class. The first part of today's class, so for example, already defined n grams, so consider P^n and R^n and P^n and C^n to be two random variables defined to represent n grams of the plaintext and n grams of the ciphertext.

So, all of us know already this formula, so it is, what this says, $H(K|C^n)$ is equal to $H(K) + H(P^n) - H(C^n)$. So, this formula we have already proved in the last day's class, yes.

So, what is $H(P^n)$ equal to? So, $H(P^n)$ we have defined from the definition of H_L . If the value of n is quite large, you can approximate that by $n H_L$ and you know that H_L by my previous result was equal to $1 - R_L \log_2$ cardinality of P base 2.

(Refer Slide Time: 15:12)

$$R_L = 1 - \frac{H_L}{\log_2 |P|}$$

$$\frac{H_L}{\log_2 |P|} = 1 - R_L$$

$$H_L = (1 - R_L) \log_2 |P|$$

if $H_L \downarrow \Rightarrow R_L \uparrow$

$$n H_L = n (1 - R_L) \log_2 |P|$$

$$\therefore \frac{H(P^n) \approx n H_L = n (1 - R_L) \log_2 |P|}{H(C^n) \leq n \log_2 |C|} \Rightarrow \frac{H(C^n)}{n} \leq \log_2 |C|$$

So, therefore, if I would like to calculate the value of H , $n H_L$, then I just need to multiply this particular thing, **by n in in**, by n . So, I get n into $1 - R_L$ into \log of cardinality of P base 2. So, that is the value of $n H_L$.

So, therefore, I can say from here, that $H(P^n)$, which is approximately equal to $n H_L$, I can write that as equivalent, equal to n into $1 - R_L$ into \log of cardinality of P base 2 and the next thing that we want is H of C^n .

So, I write, that H of C^n is equal to or rather is less than equal to n of \log of cardinality of C base 2, why? Because whatever be the entropy, so I am considering n grams, so this is equivalent, is saying, that H of C^n by n is less than equal to \log of cardinality of C base 2. So, all of us know this fact because this essentially capture the entropy of a random ciphertxt.

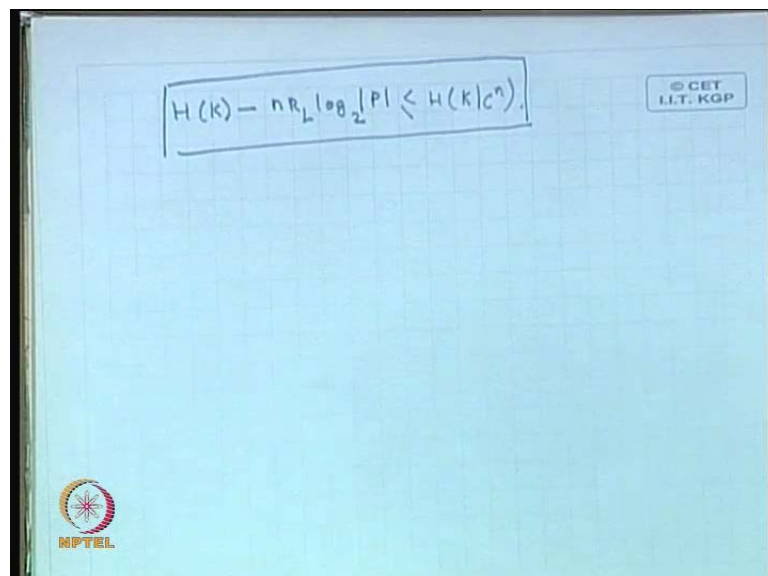
So, whatever be it, the entropy that is divided by n should obviously be less than a random key. So, this, this has got the, you know, the maximum entropy most uncertainty. So, therefore, this value is obviously is lesser than this, so you will be using these two bounds in our calculation.

So, one bound is given by $H(K|C^n)$ is less than equal to $n \log$ cardinality of C base 2 and the other approximation is $H(K|P^n)$ is approximately equal to $n(1 - R/L) \log$ cardinality of P base 2. So, we will plug this equation, these 2 things into our, into our equivocation formula, that we had.

So, the formula that we had was $H(K|C^n)$ is equal to $H(K) + H(K|P^n) - H(K|C^n)$. So, we have a fair amount of the estimate of the $H(K|P^n)$ and $H(K|C^n)$, so if you plug that, we get this value, that is, $H(K|C^n)$ is greater than equal to $H(K) - n(R/L) \log$ cardinality of P base 2. So, do you see that? So, you see, that if I subtract $H(K|P^n) - H(K|C^n)$, then these particular terms, that is, $n \log$ cardinality of P base 2 minus $n \log$ cardinality of C base 2 gets cancelled if the cardinality of P and C are same.

So, if I just consider both of them are English letters for example, then the cardinality of P and cardinality of C is the same thing. So, they are the same value. So, therefore, they cancel each other and we have got a lower bound of $H(K|C^n)$, which says, that $H(K|C^n)$ is definitely greater than $H(K) - n(R/L) \log$ of cardinality of P base 2. So, therefore, let us remember this formula, so that, because we will be needing this later on.

(Refer Slide Time: 18:12)



It says that $H(K)$, if I write it in other way, minus n of R/L log of cardinality of P base 2 is lesser than equal to $H(K|C^n)$. So, now, we will try to prove an upper bound of $H(K|C^n)$, that is, $H(K|C^n)$ should be lesser than equal to some term. So, from

there we will try to find out the, quantize the values of the spurious keys. So, till this part is clear.

(C)


Yeah, we are not assuming anything of the key, so for example we have just assumed that the ciphertext cardinality and the plaintext cardinalities are the same, but the key is, we have not assumed the size of the key. This calculation has got is independent of that information.

(Refer Slide Time: 19:04)

Possible Keys

- Define, $K(y) = \{\text{possible keys given that } y \text{ is the ciphertext}\}$
 - that is $K(y)$ is the set of those keys for which y is the ciphertext for meaningful plaintexts
- When y is the ciphertext, number of keys is $|K(y)|$
- Out of them, only one is correct. Rest are spurious.

So, number of spurious keys = $|K(y)| - 1$

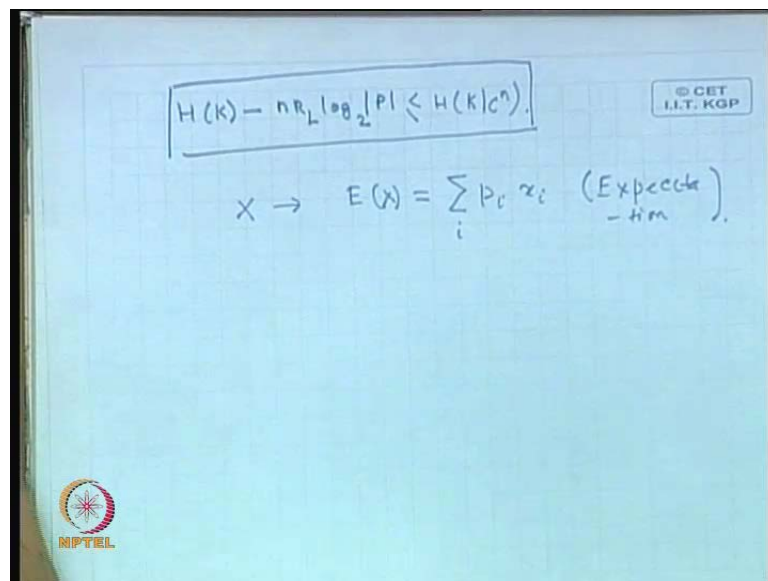
 NPTEL

So, therefore, consider possible keys. So, therefore, K_y , just define K_y to be the possible keys given that y is the ciphertext. So, define this set. So, what does it mean? It is, that possible keys given that y is the ciphertext, so I already define. What it means? It means, that K_y is the set of those keys for which y is the ciphertext for meaningful plaintexts.

So, therefore, as I told you that, that is a cryptanalyst take or an attacker takes the ciphertext, assumes the value of the key and finds out those keys or registers those keys for which the plaintext is meaningful and that is denoted by the set K_y . So, therefore, the K_y set holds those keys for which the corresponding plaintext is meaningful. So, out of these keys, how many is, how many are spurious keys? Cardinality of K_y minus 1 because only one key is the actual key, rest is spurious.

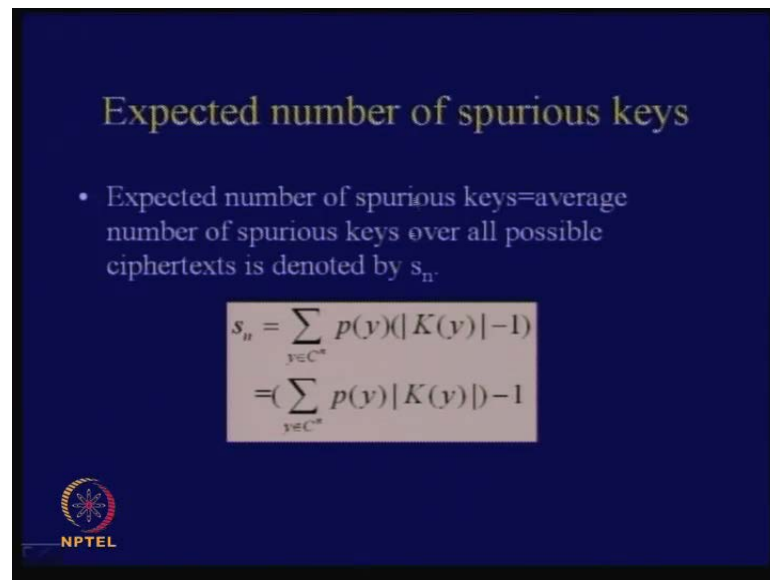
So, therefore, we know that when y is the ciphertext, number of keys is modulo of $K y$. So, out of them only one is correct, so rest of them are spurious. So, the number of spurious keys can be found out by cardinality of $K y$ minus 1. So, what is the expected size of cardinality? So, you know that this is actually a distribution. So, therefore, in order to calculate the expected value of a random variable, what do we do? We multiply the corresponding value with its probability and do a sigma.

(Refer Slide Time: 20:35)


$$H(K) = nR_L \log_2 |P| < H(K|C^n)$$
$$X \rightarrow E(X) = \sum_i p_i x_i \text{ (Expectation)}$$

So, I guess we know this result, that if there is a random variable x , then its expected value $E x$ is computed by sigma of its corresponding probability into the x_i , where i runs over all possibilities. So, that is the way how we calculate the expectation of a random variable.


(Refer Slide Time: 20:58)



Expected number of spurious keys

- Expected number of spurious keys=average number of spurious keys over all possible ciphertexts is denoted by s_n .

$$s_n = \sum_{y \in C^n} p(y)(|K(y)| - 1)$$
$$= \left(\sum_{y \in C^n} p(y) |K(y)| \right) - 1$$

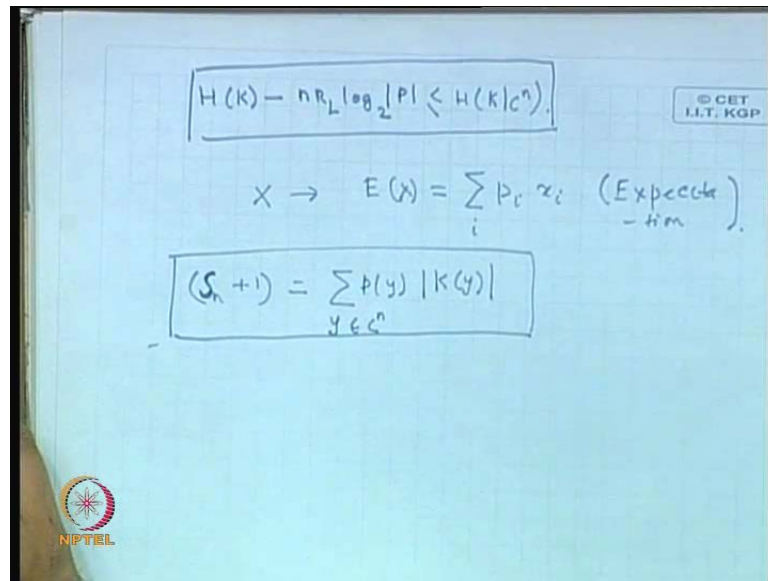
 NPTEL

So, we apply this and we find out the expected number of spurious keys. So, what is the expected number of spurious keys? Here, it is the average number of spurious keys over all possible ciphertext and this is denoted by the variable S_n . So, S_n is nothing but sigma of cardinality of K_y . We are just multiplying, K , cardinality of K_y minus 1 because this is the number of spurious keys and we are multiplying that by the probability of this event.

So, what is the probability of this event, that the ciphertext y is chosen? So, that is P_y and that is valid for, done, this calculation is done for all possible ciphertexts. So, you know, if I just simplify this formula, then I can actually multiply this sigma, this P_y with K_y and I obtain this, distribute this P_y over 1, then I obtain sigma of P_y . So, what is sigma of P_y overall possible ciphertext? It is unity, 1, so I get sigma of P_y multiplied with the cardinality of K_y and that, from there I subtract the value of 1. So, this gives me what? This gives me the expected number of spurious keys.

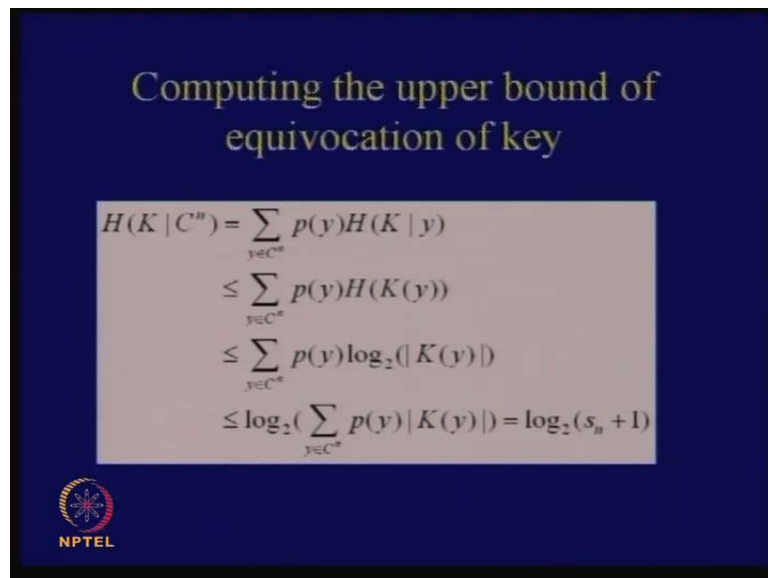
So, therefore, from here I can write, that S_n plus 1, I can, I can just reorganize this equation, I can write S_n plus 1 is equal to this particular thing.

(Refer Slide Time: 22:22)



So, therefore, I can write like, S_{n+1} is equal to sigma of $P(y)$ cardinality of $K(y)$, where y varies overall possible ciphertext. So, this is actually, also we have this and this is also we can put down from the definition of S_n .

(Refer Slide Time: 22:47)



So, so therefore, the, if we need to calculate the upper bound of the equivocation of key, we do further calculation from the definition of $H(K|C^n)$. So, what is the $H(K|C^n)$? So, as I told you, that there are 2 random variables here, K and C^n , what we do is, that let us vary one random variable and keep the other one constant.

So, therefore, we vary in this case y and we keep the value of K constant. So, therefore, from our definition of condition or entropy, we can write that is equal to σ of P_y multiplied by H of K given y . So, this we had actually written in last day's class, you can follow that. Therefore, this is equal to, actually this is this equal to, should be actually an equal to, so this is equal to σ of P_y , I just write this as K_y . Therefore, what does it mean? It means K given y . So, that is exactly the definition of h , that is, that is exactly the definition of K_y .

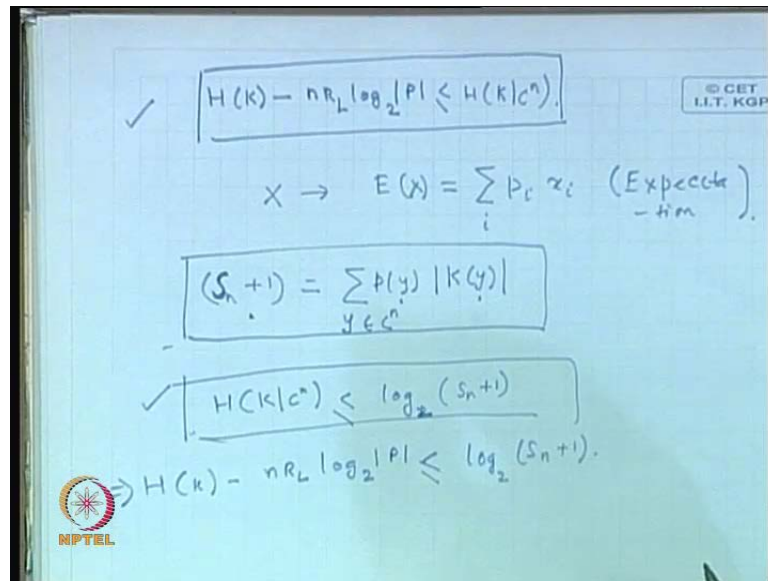
So, I obtain the σ , I obtain the, multiply P_y with $H K_y$ and I take the corresponding σ . Now, this is less than equal to P_y , multiply with logarithm of cardinality of K_y base 2 and taken a σ and this follows from what I already told you, that if this K_y would have been a, so I mean, if for example this H of K , this K_y would have been a random distribution. In that case, this would have been an upper bounded, this would have been equal to logarithm of cardinality of K_y base 2 for any other distribution; this uncertainty is lesser than a random distribution.

So, therefore, I can actually find out an upper bound and this is the upper bound, then I apply a sudden result from mathematics called genesis inequality, but it is applicable for the logarithm series because it is a monotonically increasing function. But let us just believe this fact, that I can actually find out an upper bound of this, which is called, so I can upper bound this particular thing by this expression. So, I can take the log out and I can take this, this as follows.

So, what is this particular σ equal to? This σ is equal to S_{n+1} . From this result you see this and σ of P_y cardinality of K_y was equal to S_{n+1} . So, we use this result and plug into that equation, we obtain this. So, you see that this is equal to σ of P_y cardinality of K_y and instead of this I can write S_{n+1} .

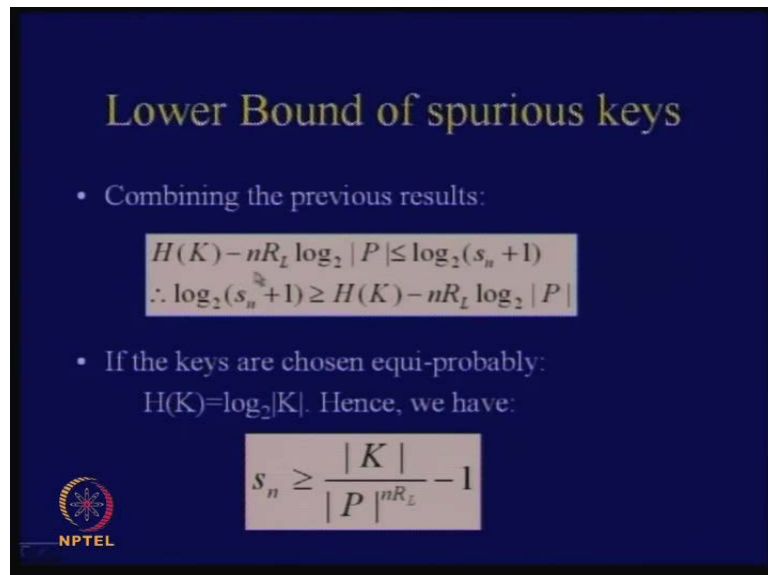
So, what we have proved is that $H K$ given C_n is less than equal to logarithm of S_{n+1} base 2.

(Refer Slide Time: 25:28)



So, what we are proved is $H(K|C^n)$ is less than equal to logarithm of $S_n + 1$ base 2. So, now, we can actually take this result and we can combine this fact and this fact and obtain that, rather write that, $H(K) - nR_L \log_2 |P|$ is less than equal to logarithm of $S_n + 1$ base 2. Is it ok?

(Refer Slide Time: 26:19)



So, that is precisely written, what here, so I, I write, that $H(K) - nR_L \log_2 |P|$ is less than equal to logarithm of $S_n + 1$ base 2. So, therefore, I can

actually really reorganize this and write as, rearrange this and write as logarithm of S^{n+1} base 2 is upper bounded by H_K minus $n R L \log$ cardinality of P base 2.

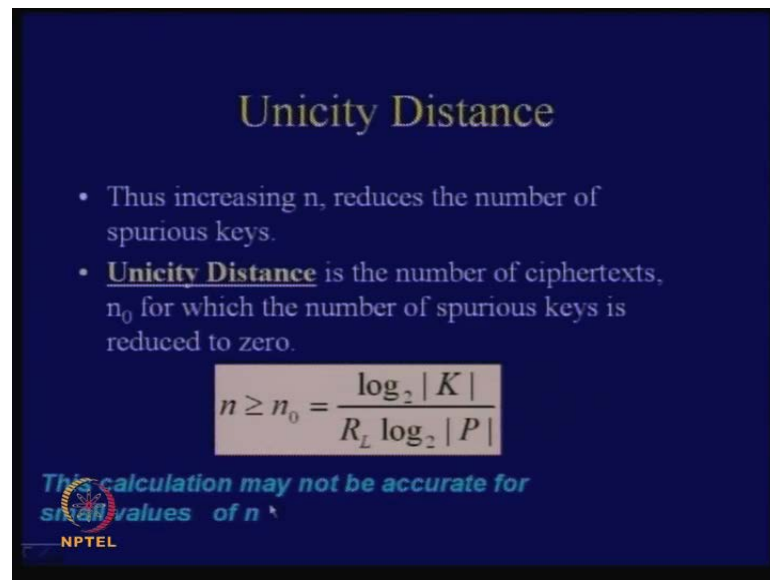
So, now, if the keys are chosen equi-probably, that is, if all the keys are equally likely, then I can actually write, that this as an equality, I can write, that H_K is equal to logarithm of K base 2. So, in that case I can plug this into this previous equation and this works out to this equation. So, this, this and this inequality, so I can obtain, that S^{n+1} , I mean, rather S^{n+1} greater than equal to cardinality of K divided by cardinality of P to the power of $n R L$. So, this follows from this equation.

If you plug-in the value of H_K and make it equal to logarithm of cardinality of K base 2, so you just see, that if I take this and if I plug here, then I, I, I can actually write this as logarithm of cardinality of P base 2 to the power of $n R L$ and from there I obtain a log here and this is also a log. So, the subtraction of log would be log a by b. So, therefore, I can write that as logarithm of cardinality of K divided by cardinality of P to the power $n R L$ base 2 and, and on the left-hand side, I have got S^{n+1} .

So, therefore, since I have got two logs on both sides and we have got an increasing function, I can actually write the S^{n+1} is greater than equal to, **cardinality of P divide,** cardinality of K divided by the cardinality of P to the power of $n R L$.

So, what do you obtain from here? I mean, what is the objective, the objective is, what is, you note, what is the value of n , what was n ? n was the number of ciphertexts, that I had provided. So, therefore, now I would like to make the number of spurious keys equal to 0 that is the object; that was the objective finally.

(Refer Slide Time: 28:25)



Unicity Distance

- Thus increasing n , reduces the number of spurious keys.
- **Unicity Distance** is the number of ciphertexts, n_0 for which the number of spurious keys is reduced to zero.

$$n \geq n_0 = \frac{\log_2 |K|}{R_L \log_2 |P|}$$

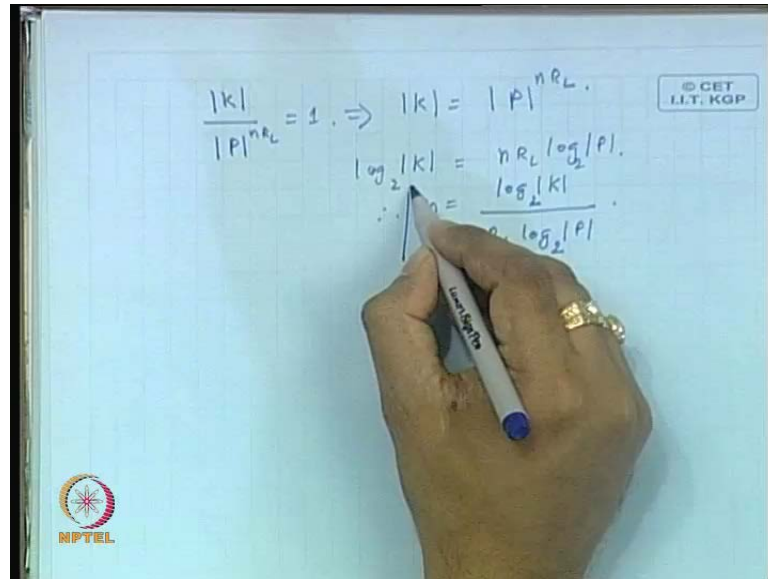
This calculation may not be accurate for small values of n .

NPTEL

So, unicity distance says, that the, thus increasing n , we obtain, obtain from here, that if I increase the value of n , then that reduces the number of spurious keys. That means what? If I provide an attacker more and more information, then the number of spurious keys gets reduced.

So, unicity distance is that particular number of ciphertexts, so I call it n_0 for which the number of ciphertexts, rather the number of spurious keys is actually reduced to 0. So, if I, if I, for example in the previous equation, if I had made the value of n , so I can actually write, that if I just plug-in 0 to that previous value, I would have obtained this bound.

(Refer Slide Time: 29:12)



The image shows a hand writing on a whiteboard. The equations written are:

$$\frac{|K|}{|P|^{n_{RL}}} = 1 \Rightarrow |K| = |P|^{n_{RL}}$$
$$\log_2 |K| = n_{RL} \log_2 |P|$$
$$\therefore n = \frac{\log_2 |K|}{n_{RL} \log_2 |P|}$$

There is a small logo in the bottom left corner that says "NPTEL" and a copyright notice in the top right corner that says "© CET I.I.T. KGP".

You see this, that is, if this was equal to 0, then I would have obtained, would have obtained, what would have obtained? That cardinality of K divided by cardinality of P to the power of n R L is equal to 1. So, that means, that cardinality of K is equal to cardinality of P to the power of n R L. So, therefore, that is exactly this particular equation you see, that n is equal to logarithm of K base 2 divided by R L log P base 2.

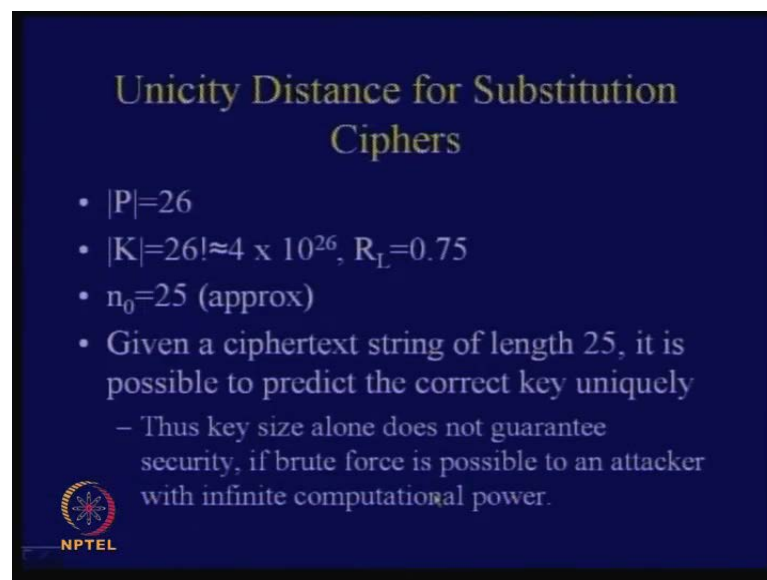
So, actually, in my equation I can take log on both sides and I can write log K base 2 is equal to n R L log cardinality of P base 2. So, what is the value of n here? It is equal to log of cardinality of K base 2 divided by R L logarithm of P base 2. So, this is the value of n for which my number of spurious keys just becomes equal to 0. So, for unicity distance would be greater than that, so therefore if I had provided you more and more, more ciphertexts, then actually, your number of keys would have been 0.

So, therefore, if I use the cipher within this unicity distance, then the number of spurious keys will not be 0. So, that means, the attacker is not able to exactly find out the unique value of the key and we have, we have, so throughout our calculation we actually consider an unbounded advisory. So, even an unbounded advisory would not actually, actually find out the actual value of the key, not the unique value of the key, but you will have a set of possible keys. And unicity distance is actually that number of ciphertexts for which this number of spurious keys just becomes equal to 0.

So, therefore, we will obtain this particular lower bound. So, therefore, this is the lower bound of the number of spurious keys, number of, for the, for the lower bound, for the unicity distance. So, beyond that is actually the attacker is able to find out the unique value of the key. So, note, that this calculation may not be accurate for small values of n , why and because my original H_n definition relied upon the fact, that limit n tends to infinity, so that was an assumption that I made.

So, therefore, this may not be very much true for n equal to 1, 2 or so on. It should be, it should be, should be fairly ok for large values of n .

(Refer Slide Time: 31:23)



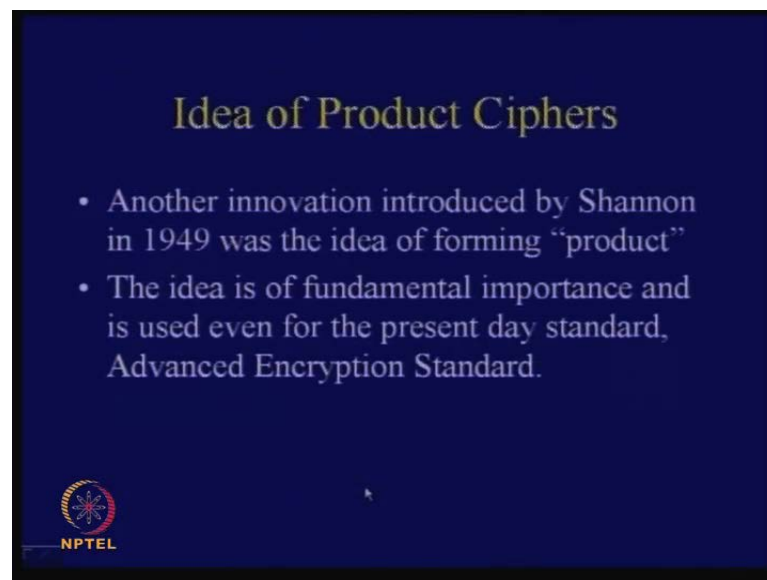
The slide has a dark blue background with yellow and white text. The title 'Unicity Distance for Substitution Ciphers' is centered at the top in yellow. Below it, there are four bullet points in white. The first three are: '|P|=26', '|K|=26!≈4 x 10²⁶, R_L=0.75', and 'n₀=25 (approx)'. The fourth bullet point is 'Given a ciphertext string of length 25, it is possible to predict the correct key uniquely', followed by a sub-bullet point: '- Thus key size alone does not guarantee security, if brute force is possible to an attacker with infinite computational power.' In the bottom left corner, there is a circular logo with a red and blue design and the text 'NPTEL' below it.

So, let us do an example calculation of, with the substitution cipher. So, we had number of plaintexts characters equal to 26, so the cardinality of P was 26, cardinality of K was 26 factorial. So, that was around 4 into 10 to the power of 26, fairly large value of the key. So, your R_L , if you assume is equal to 0.75 of English language, then if you plug-in, you will find, that n_0 is approximately equal to 25.

So, that means, that given a ciphertext string of length 25, an unbounded attacker can actually predict the unique value of the key. So, thus, what we observe from here is that a key size alone, such a large key size does not guarantee security if brute force is possible to an attacker with infinite computational power.

So, an attacker who has got an, has got an unbound, I mean, has got an infinite computational power, for him such a big size of key, actually he requires just 25 ciphertexts to actually find out the value of the key. All these majors are of course probabilistic, but it will match with the actual result quite closely.

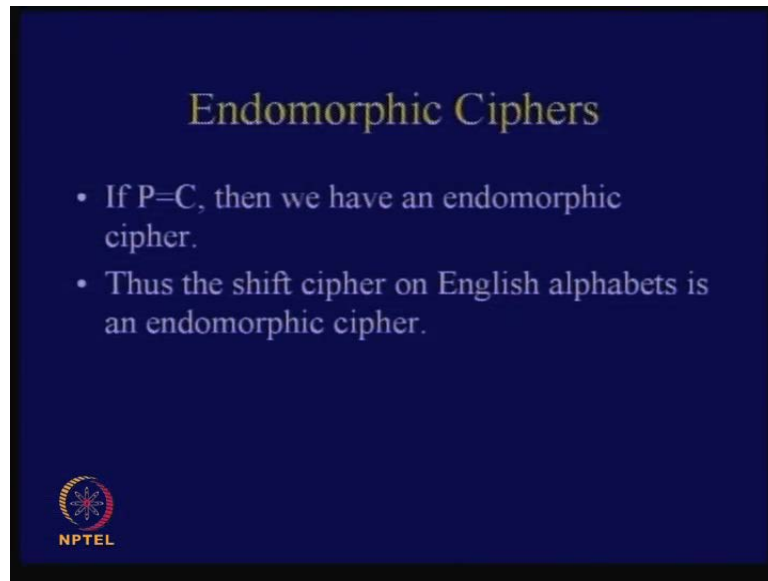
(Refer Slide Time: 32:35)



So, this, with this we essentially conclude this part of unicity distance, but we will conclude the remaining part with an idea of product ciphers. So, actually, if, after this we will actually go, start talking about real ciphers like block ciphers and string ciphers, but before this I would like to mention about the idea of product ciphers.

So, this was actually also mentioned in Shannon’s paper and that is why it is called the seminar paper. It was mentioned as old in 1949 and (()) the idea of forming products. So, the idea is still fundamental because even present, present day ciphers like AES for example, still uses the concepts of product ciphers. So, let us try to understand the concept of product ciphers and actually you will observe, through that lot of things becomes meaningful; lot of things, which we see in our future ciphers will actually becomes meaningful. So, let us try to see.

(Refer Slide Time: 33:27)



So, before that I would just like, in order to simplify our life, let us, I just coin a term called endomorphic ciphers. So, what is an endomorphic cipher? Endomorphic ciphers are those ciphers for which the plaintext and ciphertext are the same sets.

So, for a normal substitution cipher, your plaintext and ciphertext were just English language, I mean, English characters.


So, if P and C are the same, then we have what is called an endomorphic cipher. Therefore, the shift cipher of an, on English language, on English alphabets was an example of an endomorphic cipher.

(Refer Slide Time: 34:00)

What we have learnt from history?

- **Observation:** If we have an endomorphic cipher $C_1=(P,P,K_1,e_1,d_1)$ and a cipher $C_2=(P,P,K_2,e_2,d_2)$.
- We define the product cipher as $C_1 \times C_2$ by the process of first applying C_1 and then C_2
- Thus $C_1 \times C_2=(P,P,K_1 \times K_2,e,d)$
- Any key is of the form: (k_1,k_2)
and $e=e_2(e_1(x,k_1),k_2)$. Likewise d is defined.

Note that the product rule is always associative



So, consider an endomorphic cipher and let us try to understand certain things from history. So, therefore, if we have an endomorphic cipher, so C_1 , you note, that I have written (P, P) because P and C are the same things, that the plaintext set and the ciphertext sets are the same things.

So, I write (P, P) and then followed, follow that with K_1 because K_1 denotes the key set of the encryption function C_1 , that is, the cipher C_1 and you have got an encryption function e_1 and corresponding decryption function d_1 . You also have a cipher, which is called C_2 and I denote that with (P, P, K_2, e_2, d_2) .

So, that means what the key two is? The set key, of the keys for the 2nd cipher e_2 , the corresponding encryption function and d_2 is the corresponding decryption function. So, let us try to understand or define what is mean by product cipher C_1 cross C_2 . So, what does C_1 cross C_2 means? It just means, like as you know, that you apply 2 function after, one after the other. So, therefore, if I say C_1 cross C_2 , it means, that first I will apply C_1 and follow that with the application of C_2 . So, I think, you have seen these in a case of composition of functions in **(())** structures class.

So, therefore, this exactly precisely similar kind of thing, so you see C_1 cross C_2 , I would define as P cross P because **extreme** endomorphic. You see, why it is

endomorphisms? Yeah, because that repeated application also keeps, still keeps, it is endomorphic property. So, it is still endomorphic, but your key set is the condition product of K_1 and K_2 because there is $K_1 \times K_2$ and you have got the corresponding encryption function e and decryption function d .

So, any key I can write in the form of an ordered pair, $K_1 \times K_2$ and form ordered sets like that. So, the encryption function is defined as $e = e_2 \circ e_1$ and, but initially you take the plaintext x , you take the key K_1 and you apply the function e_1 . Subsequently, you choose the key K_2 and you apply the encryption function e_2 . So, what will the corresponding decryption function look like?

(Refer Slide Time: 36:11)

The image shows a handwritten derivation on a grid background. At the top right, there is a small stamp that reads "© CET I.I.T. KGP". At the bottom left, there is a logo for NPTEL. The derivation is as follows:

$$d = d_2(d_1(y, k_1), k_2)$$

$$y = e_2(e_1(x, k_1), k_2)$$

$$d(y) = d_1\left[\underbrace{d_2(e_2(e_1(x, k_1), k_2), k_2), k_1)}_{e_1(x, k_1)}, k_1\right]$$

$$= d_1\left[\underbrace{e_1(x, k_1)}_{x}, k_1\right]$$

$$= x$$

The corresponding decryption function will look like this, it will look like $d = d_2 \circ d_1$ (y, K_1) followed that with K_2 like this. So, do you see, that if I apply d and e subsequently, they are actually inverses of each other; that is obvious because of the associativity of the product rule. So, actually you can see that because if I apply d and if I apply d over an application of the e function, then actually I obtain back where I started with. So, you can see this because of this.

(C)

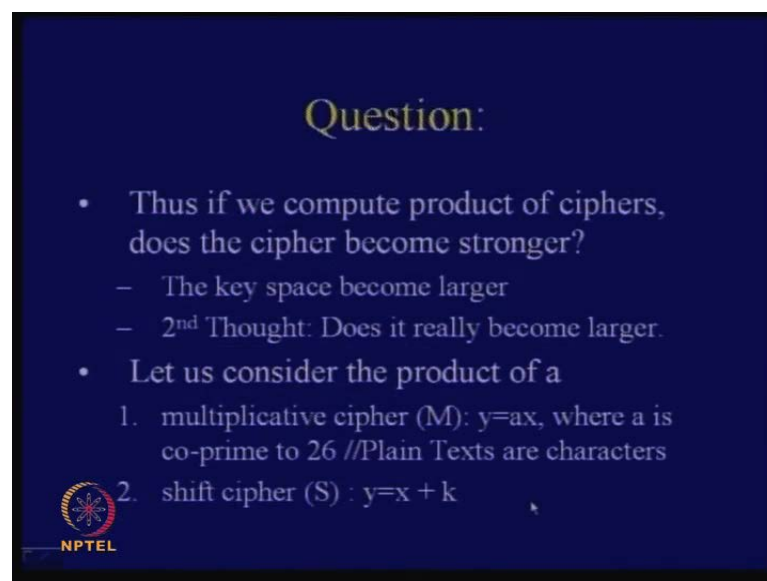
Yeah, it will be first d_1 and follow that with d_2 , yeah, so that you can obtain from the decryption function because what you do is, that you take the corresponding, see you

write this $e_2 e_1(x, K_1, K_2)$. So, that is my y , so I take this as a y and I apply my d over that.

So, therefore, what I do is, that I want to compute this, I want to compute $d y$, so for that I apply $d_1 d_2$ over $e_2 e_1(x, K_1, K_2)$ and then, so this is my, so therefore, this is my e_1 . So, therefore, what I do is, that I have actually encrypted. So, this is my scope of the function e_2 , then I apply d_2 . So, in order to decrypt this I need K_2 and follow that with K_1 . So, what you do is that you see, that in this case your d_2 and e_2 cancels each other. So, I can do that because of the associativity of the product function. So, if I do that, I obtain d_1 and I obtain then $e_1 x$ of K_1, K_1 . So, again, this d_1 and e_1 cancels each other and I finally obtain back the value of x .

So, therefore, you, therefore you see, that d is actually a corresponding decryption function. So, this follows because of this fact, that is, the product rule is always associative.

(Refer Slide Time: 38:37)



Question:

- Thus if we compute product of ciphers, does the cipher become stronger?
 - The key space become larger
 - 2nd Thought: Does it really become larger.
- Let us consider the product of a
 1. multiplicative cipher (M): $y=ax$, where a is co-prime to 26 //Plain Texts are characters
 2. shift cipher (S) : $y=x + k$

NPTEL

So, the question is, that if we can compute product of ciphers, thus the cipher becomes stronger, that is what is most important. So, I take two small ciphers and I compose them I compute the product. Does the key, thus is, does the cipher becomes stronger, that means, does the key space becomes really larger? So, in the initial, on the surface we have actually (K_1, K_2) .

So, the condition product size should increase, but on a second thought, does it really become larger? So, in order to understand that, what is your opinion on that? Will it really become larger; will the key size be become larger? So, actually, not always, so let us try to consider one example, I guess it will be lot clear through that. So, let us consider a simple multiplicative cipher. So, what it does is, that it just takes x and multiplies with a , where a is co-prime to 26. So, you know what is co-prime?


So, a is co-prime means, it is GCD of a and 26 is 1 and next is you consider a shift cipher where you take x and you add that with K . So, therefore, this is my m and this is my S , so this was an example of what? It was an example of a computation cipher and this was an example of a substitution cipher.

(Refer Slide Time: 40:04)

Is $M \times S = S \times M$?

- $M \times S$: $y = ax + k$: key = (a, k) . This is an affine cipher, as total size of key space is 312.
- $S \times M$: $y = a(x + k) = ax + ak$
 - Now, since $\gcd(a, 26) = 1$, this is also an affine cipher.
 - key = (a, ak)
 - As $\gcd(a, 26) = 1$, a^{-1} exists. There is a one-one relation between ak and k . Thus the total size of the key space in $S \times M$ is still 312. Thus this is also the affine cipher

Thus S and M are commutative.

 NPTEL

So, now you consider, that for example, that we have got M and you have got S and what you do is that you just compose this M cross S and you obtain this, that is, y is equal to ax plus K is, you see, you, first of all, have, so you need to first do the multiplication. So, you do a S , I mean, ax and then, you need to do S . So, you add K with ax , so what is the key? In this case the key is $(())$, (a, K) and you know, that this is an example of what? It is an example of affine cipher. So, what was the key size in case of English language? What was the size of affine cipher? It was equal to 312.

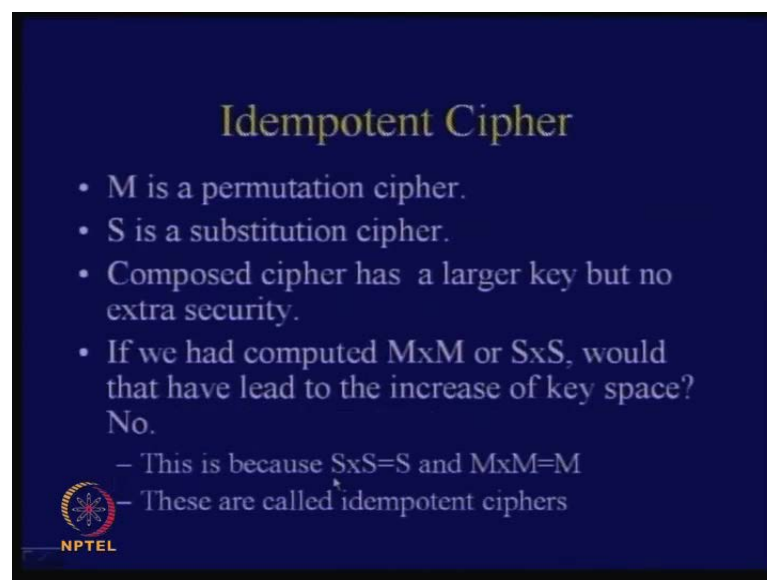
So, now you consider S , from S cross M , so x cross M is equal, y is equal to ax plus K , so that is, you can also write that as ax plus $a K$. Now, you know, that GCD of $(a, 26)$ is

1. So, therefore, this is also an affine cipher and the key would be $(a, a^{-1}K)$, but since the GCD of a and 26 is 1, so an inverse exists. This we discussed and actually there is a one to one relation between $a^{-1}K$ and K . So, therefore, the total size of the key space in S from S cross M is still 312.

So, you see, that here we have got the key $a^{-1}K$ and here we have the key K , but since there is an inverse of a , that is, actually a one to one correspondence between this set and this set. So, if there are 26 possibilities of K , then also $a^{-1}K$, since you are doing a modulo 26 also has got 26 possibilities.


So, that means, this set and this set are essentially the same things. So, that means, in both the cases you do M cross S or you do S cross M , your key size is still 312. So, what you see is that M cross S and S cross M are same. So, that is what is called commutative. So, we have got commutative ciphers. So, it means that M cross S and S cross M , when they are same we call them to be commutative and this is an example of a commutative cipher. It does not matter whether you do first shift and then multiply or you do first multiply and then shift, both are the same things.

(Refer Slide Time: 42:13)



Idempotent Cipher

- M is a permutation cipher.
- S is a substitution cipher.
- Composed cipher has a larger key but no extra security.
- If we had computed $M \times M$ or $S \times S$, would that have led to the increase of key space?
No.
 - This is because $S \times S = S$ and $M \times M = M$
 - These are called idempotent ciphers

 NPTEL

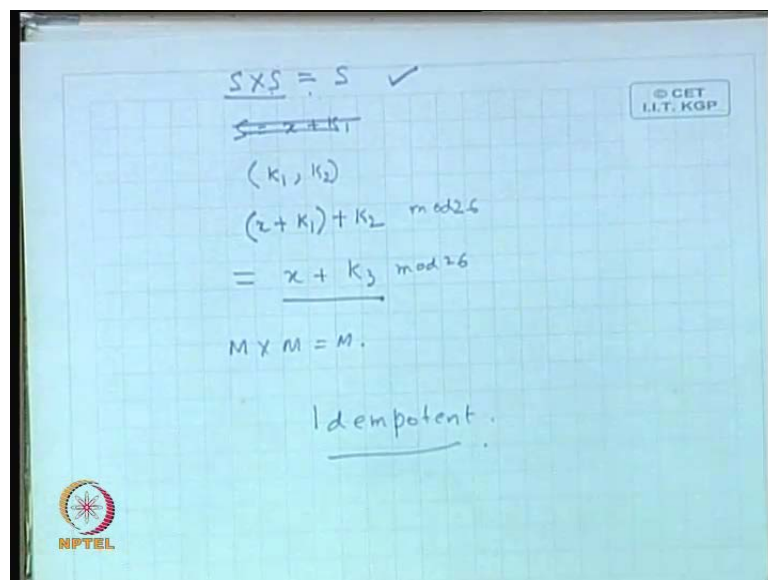
So, then, let us see what an idempotent cipher is? So, therefore, what is an idempotent function? It means, if we apply the same function twice, you obtain back the same function.

So, therefore, M is a permutation cipher, S was a case of a substitution ciphers and both of them were actually idempotent ciphers. So, a composed cipher has a larger key, but no extra security because $M \times M$, if it is equal to M, then even composing M S for more than the, more than once, essentially leaves you with the same kind of transformation. So, essentially, it does not add to your security. So, therefore, for example if you had completed $M \times M$ or $S \times S$, that would not have led to the increase of the key space. So, this is because $S \times S$ and is equal to S and $M \times M$ is also equal to M and these class of ciphers are called idempotent ciphers.

So, you could easily observe from this fact, that is, if I had done an $M \times M$, what would have been the, would have, what would have that meant? I would have done a x, then a a x, but my key size would not have still increased because doing S square essentially does not increase the key space.

Similarly, consider for shift cipher, you do one shift and you do another shift. So, in both the cases you can represent that, you can represent that, that by a 3rd shift, so, do you understand what I am saying?

(Refer Slide Time: 43:32)



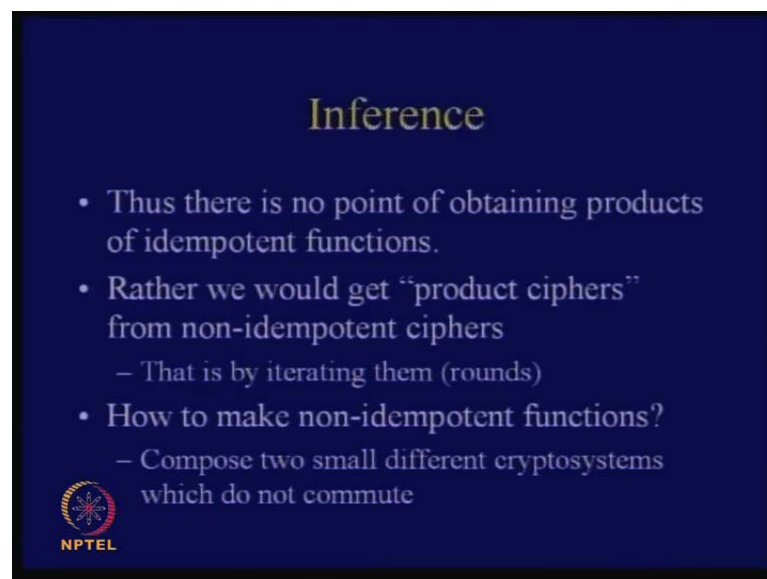
So, what I am saying is that if you consider, say S as x plus K 1 for example, so I am just considering S cross S and I am just trying to argue, that S cross S is actually equal to S. So, what is the idea? So, therefore, imagine that in the 1st phase we have got, we have got, we have got the function as you choose the key as K 1 and in the 2nd case, you

choose the key as K_2 . So, on the 1st application of S , I would have computed x plus K_1 and in the 2nd click application of S , I would have computed as K_2 . Therefore, I would have obtained x plus K_1 plus K_2 so that, since I am doing mod 26, I can always represent that as x plus some $K_3 \text{ mod } 26$, where K_3 is nothing but the summation of K_1 and K_2 .

So, that means, even for S cross S I have got the same size of the key, so it is a same cipher. Therefore, I can conclude, that S cross S is equal to S . Similarly, for M cross M also, you can actually show that M cross M is also equal to M . So, both these cross of ciphers are something which we call as idempotent ciphers.

So, therefore, we have defined what the commutative cipher is and we have defined what an idempotent cipher is, and what we will now consider is what happens if you compute the product of such kind of ciphers, which are commutative as well as idempotent.


(Refer Slide Time: 44:59)



The slide has a dark blue background with yellow text for the title and white text for the bullet points. The NPTEL logo is in the bottom left corner.

Inference

- Thus there is no point of obtaining products of idempotent functions.
- Rather we would get “product ciphers” from non-idempotent ciphers
 - That is by iterating them (rounds)
- How to make non-idempotent functions?
 - Compose two small different cryptosystems which do not commute

 NPTEL

So, actually, that you can observe from this fact, so what we are trying to observe is, that there is no point of obtaining products of idempotent ciphers. So, if you take M cross M , it is a same thing as M . So, that is no point of doing such products. So, rather you would get product ciphers form non-idempotent ciphers, that is, by iterating them.

So, if we have some non-idempotent ciphers, I would have liked to iterate them and therefore, that is essentially the concept of round, which exists into all classes of symmetric ciphers in today's world.


So, the question is how to make non-idempotent ciphers or functions? So, if I, the idea would be, that compose 2 small different cryptosystems, which do not commute.

(Refer Slide Time: 45:40)

Why?

- If there are two cryptosystems which are idempotent and also commute then their product is also idempotent.
- $(S_1 \times S_2) \times (S_1 \times S_2) = S_1 \times (S_2 \times S_1) \times S_2$
 $= S_1 \times (S_1 \times S_2) \times S_2$
 $= (S_1 \times S_1) \times (S_2 \times S_2)$
 $= S_1 \times S_2$

Thus, $M \times S$ is also idempotent. Why?
Thus, composing $M \times S$ does not help.

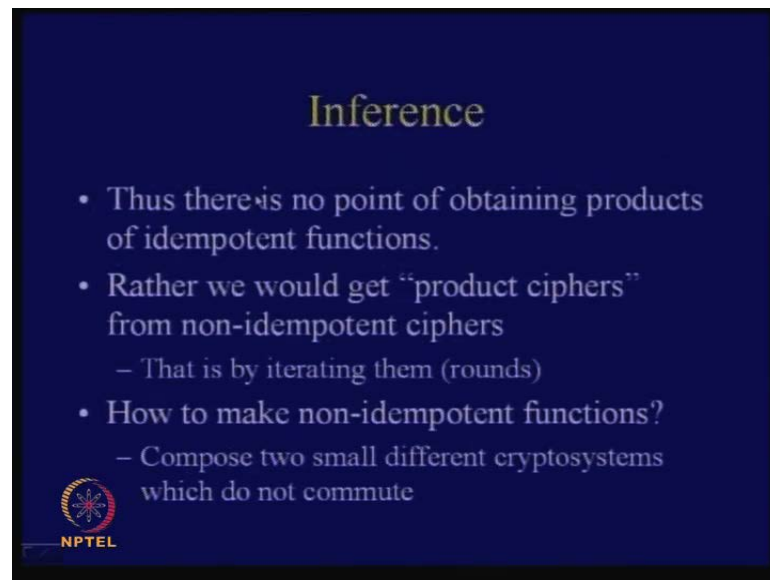
 NPTEL

So, do you follow this? If you do not follow them, then this will become clear because of this calculation. So, what was I said here is, that if there are 2 cryptosystems, which are idempotent and also commute, then the product is also idempotent.

So, if this result is true, what does it mean? It means that if you have got 2 cryptosystems, which are idempotent and also commute, then the product is also idempotent. So, what does it mean? It means, that if we obtain a function of this class, then if you take and if you take them and if you still product or rather compute the product of those kinds of ciphers, the key size does not increase.


So, therefore, considering products does not help. Therefore, from this theorem or rather this result, we know, that we actually require to, I mean, compute the products of ciphers, which even if they are idempotent, they do not commute.

(Refer Slide Time: 44:59)



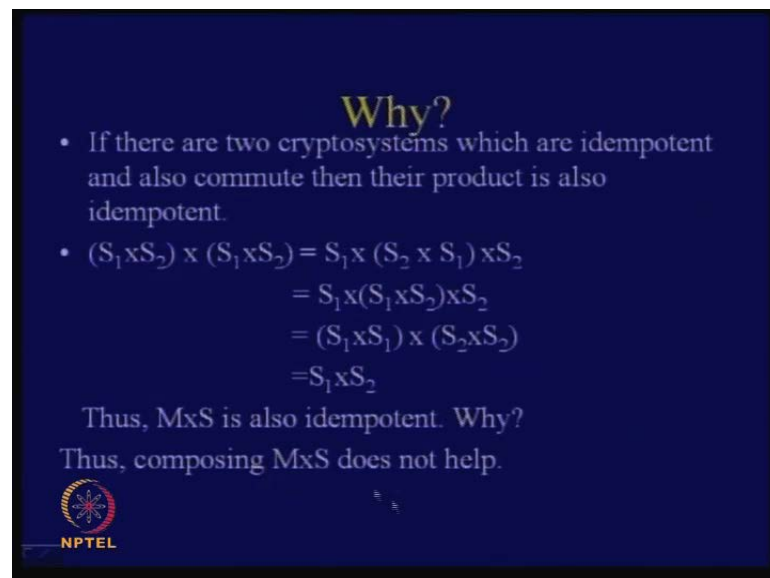
Inference

- Thus there is no point of obtaining products of idempotent functions.
- Rather we would get “product ciphers” from non-idempotent ciphers
 - That is by iterating them (rounds)
- How to make non-idempotent functions?
 - Compose two small different cryptosystems which do not commute

 NPTEL

So, that explains this point, that is, compose two small different cryptosystems, which do not commute. So, those kinds of ciphers, if you iterate them, will actually make sense.

(Refer Slide Time: 46:45)




Why?

- If there are two cryptosystems which are idempotent and also commute then their product is also idempotent.

$$\begin{aligned}(S_1 \times S_2) \times (S_1 \times S_2) &= S_1 \times (S_2 \times S_1) \times S_2 \\ &= S_1 \times (S_1 \times S_2) \times S_2 \\ &= (S_1 \times S_1) \times (S_2 \times S_2) \\ &= S_1 \times S_2\end{aligned}$$

Thus, $M \times S$ is also idempotent. Why?
Thus, composing $M \times S$ does not help.

 NPTEL

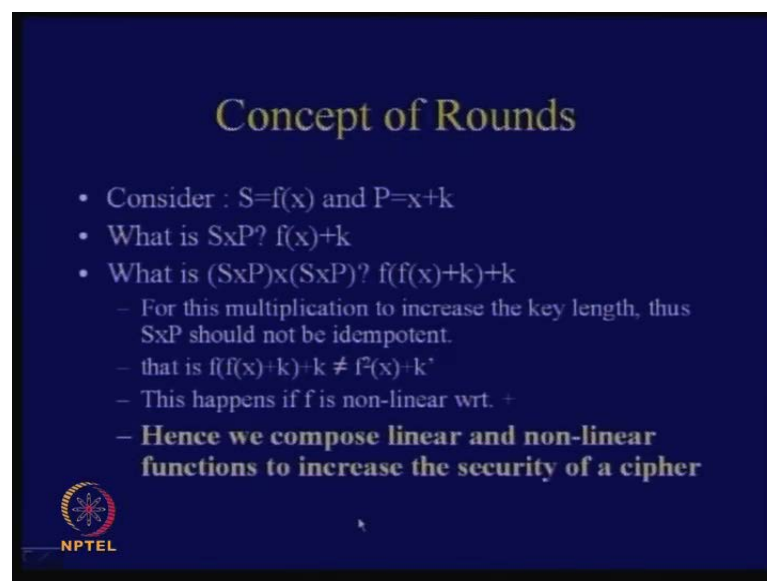
So, therefore, let us see this result, it is quite simple, it says, that S_1 and S_2 are two such cryptosystems, which are idempotent and at the same time they commute. So, $S_1 \times S_2 \times S_1 \times S_2$, I am considering the product of these things, so therefore, so if I observe that, if from the associativity I can write, like $S_1 \times S_2 \times S_1 \times S_2$ and since this commutes, $S_2 \times S_1$ becomes equal to $S_1 \times S_2$.

So, now you know, that $S_1 \times S_1$ is equal to S_1 and $S_2 \times S_2$ is also equal to S_2 , so what you have obtained is, that $S_1 \times S_2$ and product and multiplying that with $S_1 \times S_2$, essentially leaves you with $S_1 \times S_2$.

So, what does it mean? It means that this is an idempotent function. So, in your previous case we have proved, that $M \times S$, M and S were essentially, both of them were idempotent. So, therefore, can you, can you, can you show, can you understand why $M \times S$ is also idempotent? Why? It because we have proved, that $M \times S$ is equal to $S \times M$. So, that means, M and S were commutative and we have also proved that $M \times M$ is equal to M and $S \times S$ is equal to S . So, that is they were idempotent as well. So, if you compute their products, then essentially, you are left with the same thing.


So, therefore, computing their products and composing them does not help. So, therefore, you require some other additional quantity, which will help you and that is the idea of rounds.

(Refer Slide Time: 48:16)



Concept of Rounds

- Consider : $S=f(x)$ and $P=x+k$
- What is $S \times P$? $f(x)+k$
- What is $(S \times P) \times (S \times P)$? $f(f(x)+k)+k$
 - For this multiplication to increase the key length, thus $S \times P$ should not be idempotent.
 - that is $f(f(x)+k)+k \neq f^2(x)+k$
 - This happens if f is non-linear wrt. $+$
 - **Hence we compose linear and non-linear functions to increase the security of a cipher**

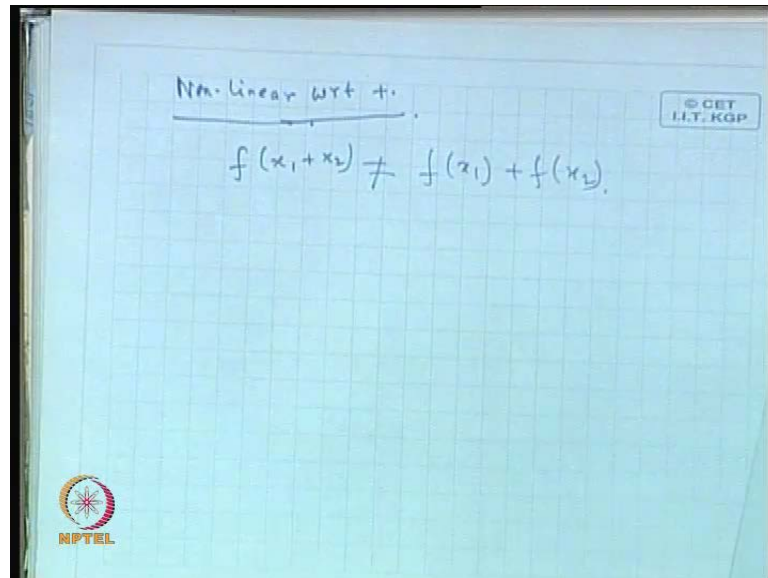
 NPTEL

So, therefore, the idea is that how can you work? What is that feature? So, till now, whatever we have seen, there is a feature missing, which we have not yet seen then. Therefore, the concept of round we are still not able to achieve.

So, what is that concept? That concept is of non-linearity, but I will define that non-linearity concept further in our subsequent classes, but this is a brief introduction to that.

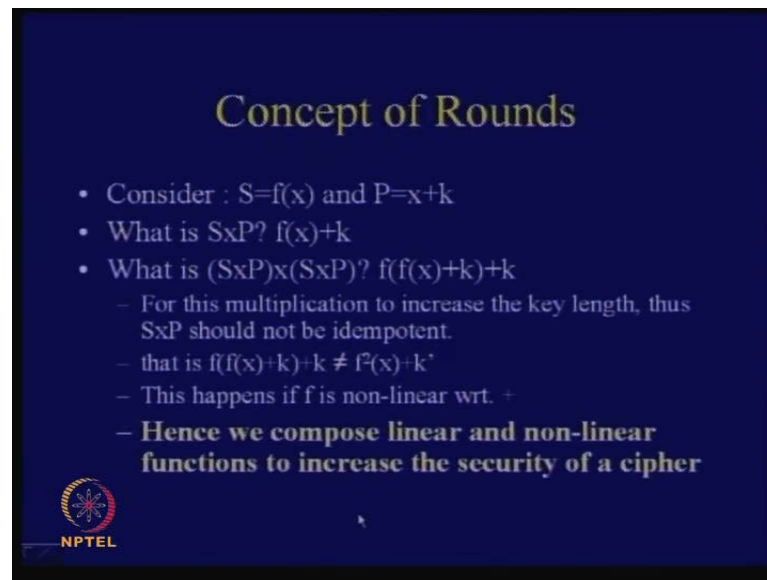
So, therefore, consider that instead of, I mean, let us consider these 2 functions, S and P, where P is actually equal to x plus K and S is the output of a function f x. Now, I claim, that this function is actually a non-linear function with respect to addition operation. So, what does it mean?

(Refer Slide Time: 49:06)




So what does it mean? So it means that if, so therefore, what does a non-linear function mean? It means that if you consider f of x 1 plus x 2, so non-linear with respect to plus non-linearity is always with respect to an operation. So, f of x 1 plus x 2 is not equal to f of x 1 plus f of x 2 and equality would have meant linearity.

(Refer Slide Time: 48:16)



Concept of Rounds

- Consider : $S=f(x)$ and $P=x+k$
- What is $S \times P$? $f(x)+k$
- What is $(S \times P) \times (S \times P)$? $f(f(x)+k)+k$
 - For this multiplication to increase the key length, thus $S \times P$ should not be idempotent.
 - that is $f(f(x)+k)+k \neq f^2(x)+k$
 - This happens if f is non-linear wrt. $+$
 - **Hence we compose linear and non-linear functions to increase the security of a cipher**

 NPTEL

So, therefore, now consider these function S is equal to $f(x)$ and P equal to $x + K$ and consider $S \times P$. So, what is $S \times P$ equal to? It is equal to $f(x) + K$ and what is $S \times P \times S \times P$? So, that, that means, that you take $f(x) + K$ and then you do a further application of f and add that with k . So, for this multiplication to increase, to increase the value of the length of the key, so thus what, what is needed? Therefore, it is needed, that $S \times P$ should not be idempotent.

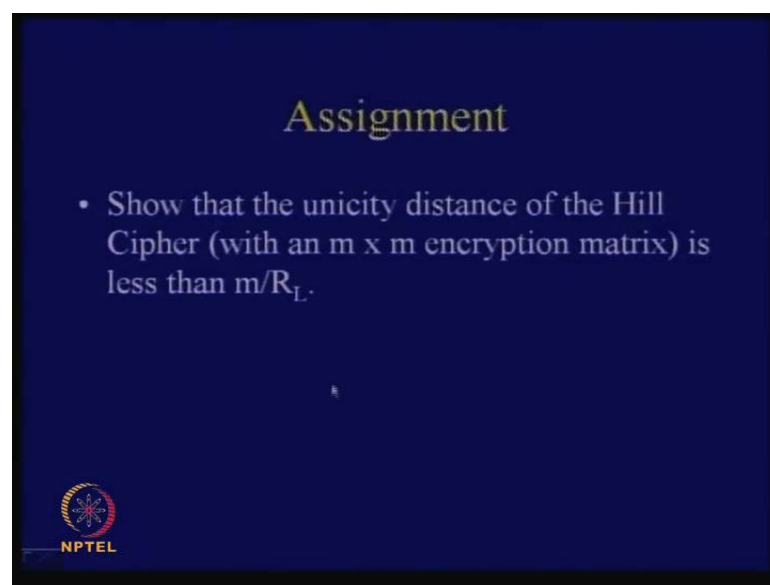
So, if that, so what we require is, that $f(f(x) + K)$, which is equal to this particular thing. When it is added with K should not be equal to $f^2(x) + K$ because if this, if you had a linear function f , then you can actually, if this, while a linear function you could have actually distributed this and this would have computed to some $x^2 + x$ plus some value of K dash.

So, that you see is exactly similar with your $S \times P$ function, it, some other of an application of f and it is added with the key K . So, therefore, what it, so therefore this happens only if f is non-linear with respect to class. So, if this was the linear function f , then actually it would have distributed and that result, that we would have obtained, would have obtained, would have been similar to that of $S \times P$.

So, the size of this function, so the size of the key of this function and the size of this composed function would have been the same. So, therefore, what we need is something a deviation from this fact. So, we need not linearity, but we need non-linearity. So,

therefore, hence we have to compose linear and non-linear functions to increase the security of a cipher. So, in order to increase the security of a cipher, but we have seen till now are only linear components, linear transformations. We essentially, found out multiplication with the matrix, which is the linear operator addition with the key, that is, also, that is also, that is also a linear function. So, therefore, all these are linear transformations. Therefore, we need, therefore you see that nicely from Shannon's theory, we can actually arrive at the fact that we require at composition of linear functions and as well as non-linear functions.

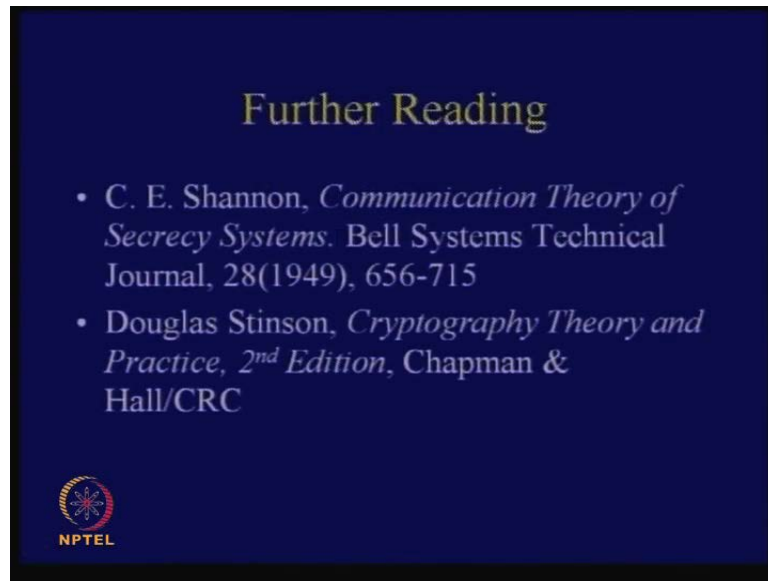
(Refer Slide Time: 51:54)



So, with this I conclude my talk, but I would like to give an assignment, which you are supposed to again do it and submit it on 20th, before 20th, both, I have given 1 assignment already. The other assignment is that show the unicity distance of the Hill Cipher with an m cross m encryption function is actually less than m divided by $R L$, where $R L$ is the redundancy, as defined in the class.

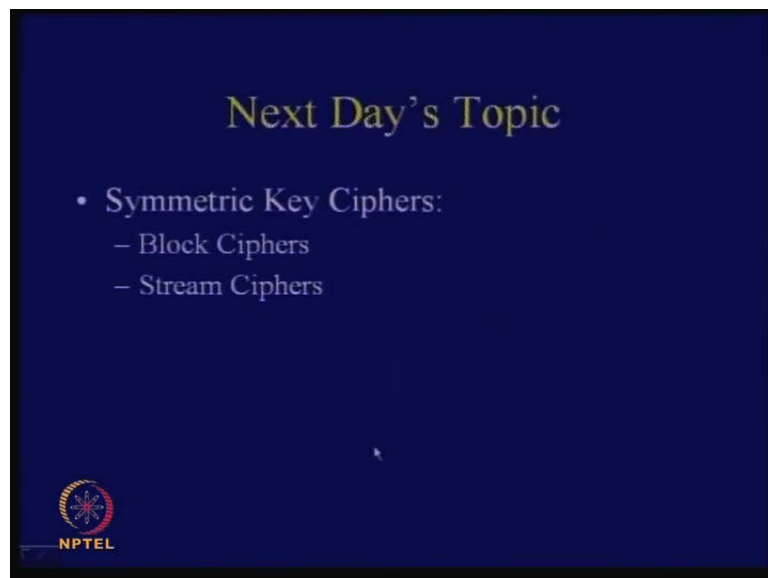
So, you can show, that the unicity distance of the Hill cipher with a m , m cross m encryption function is actually less than m divided by $R L$.

(Refer Slide Time: 52:31)



So, that is an assignment, which is given to you and you can read further things from Shannon's books which is Communication Theory of Secrecy Systems. It is actually a paper, it is a classical paper, so it Bell Systems, it appeared in Bell Systems Technical Journal, but I am sure, that you will get online. And the other text book that I have followed is from Douglas Stinson Cryptography Theory and Practice, a 2nd edition book you can follow. So, I have followed that book, although the 3rd edition exists.

(Refer Slide Time: 52:53)



And next day's topic would be symmetric key cipher, so we will use these concepts to go and build ciphers now.

So, therefore, a symmetric cipher is our next day's topic and we will start with block ciphers and follow that with stream ciphers.