Cryptography and Network Security

Prof. D. Mukhopadhyay

Department of Computer Science and Engineering

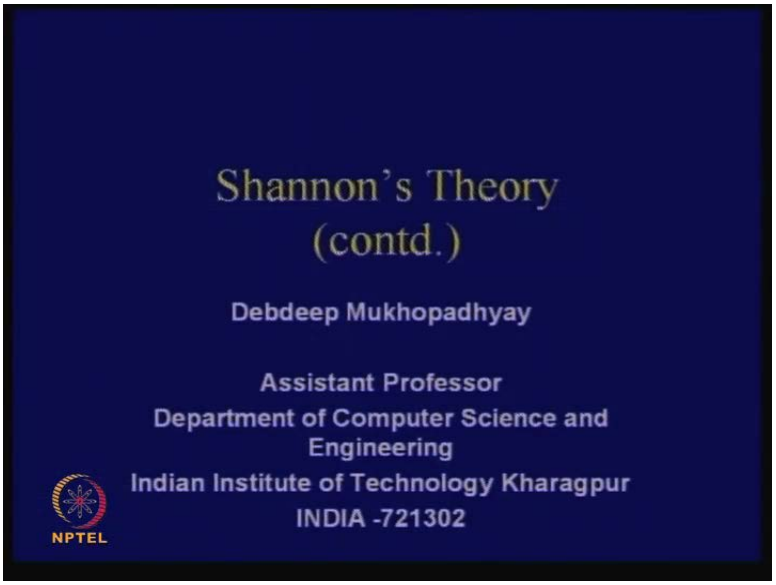
Indian Institute of Technology, Kharagpur

Module No. # 01

Lecture No. # 08

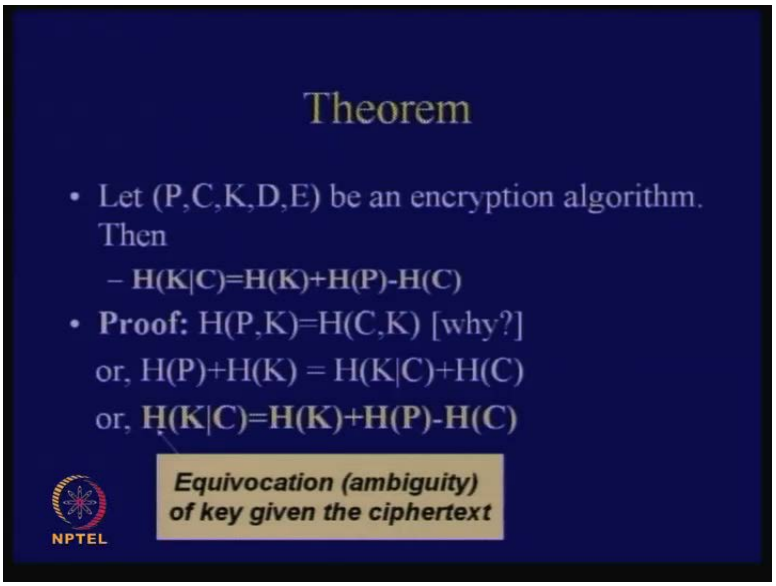
Shannon's Theory (Contd.)

(Refer Slide Time: 00:27)



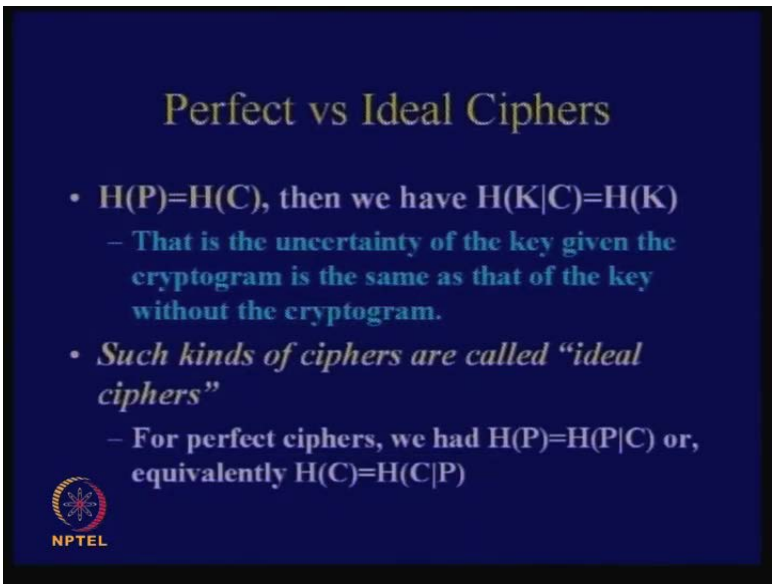
So, in today's talk, we will conclude the topic of Shannon's theory. So, in the previous class, if you remember, that we had concluded with the idea of equivocation of keys. So, we have defined briefly, that what is the idea behind spurious keys. So, today, we will try to understand, whether we can compute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

(Refer Slide Time: 00:47)



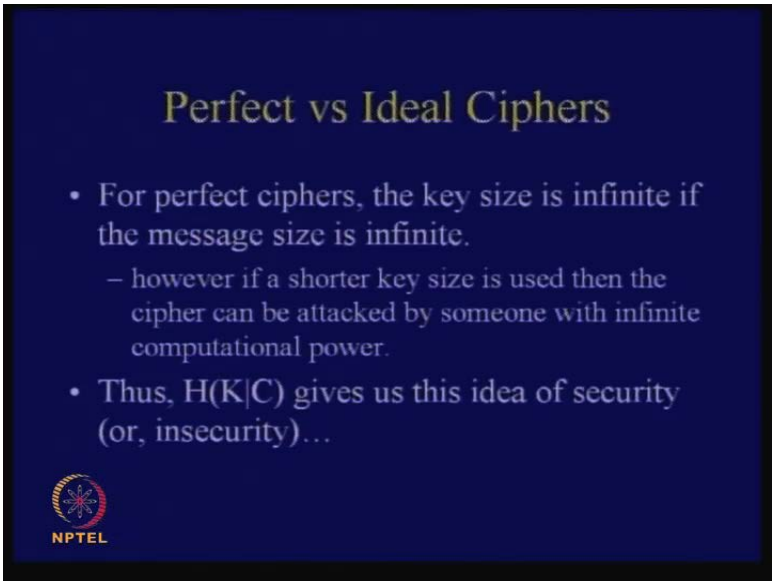
So, if I just would like to recap, that we essentially proved this particular formula yesterday, that is, HK given C is equal to HK plus HP minus HC. So, therefore, the ambiguity of a key, given the ciphertext is described as follows, that it is addition of the ambiguity of the key plus the ambiguity of the plaintext subtracted with the ambiguity of the cipher text.

(Refer Slide Time: 01:12)



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

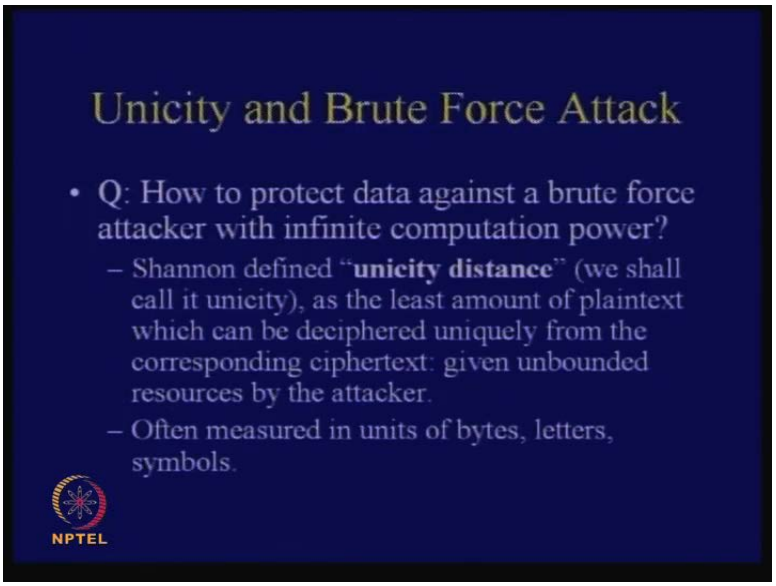
(Refer Slide Time: 01:32)



So, these are the things that we described yesterday and concluded with it, and so, therefore, HK given C gives us the idea of the security or insecurity. 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 were not practical. So, when we define another kind of ciphers, called the ideal ciphers, where HK given C is equal to HK and 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 an abounded 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, up to 2 to the power of 80 computations, then we are fairly happy and we say, that the cipher has achieved computational security.

(Refer Slide Time: 02:36)



So, the question is how to protect? So, therefore, but still 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? Therefore, that is, an attacker who has got infinite computational power and still, can I protect a cipher?

So, the idea is, that is, Shannon defined a certain parameter, which is called the unicity distance, and he said, that, that is the least amount of ciphertext, which I would like to make it available to this, to the adversary, who is an unbounded adversary, so that he does not find out a unique value of the key.

So, the strategy of the, of the adversary is as follows, what he does is, is that he takes the ciphertext, he assumes a key, decrypts it back and finds out the plaintext. If it, 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, any language for that matter, the adversary will not actually find out the main, I mean, key unique. So, what he will essentially have? He will have a set of keys. So, apart from the actual key, the other keys are called the spurious keys.

Sir, how will the attacker say, that doing the iterations and how will you actually decide, how will the computer decide that your text is meaningful? We cannot check each and every file.

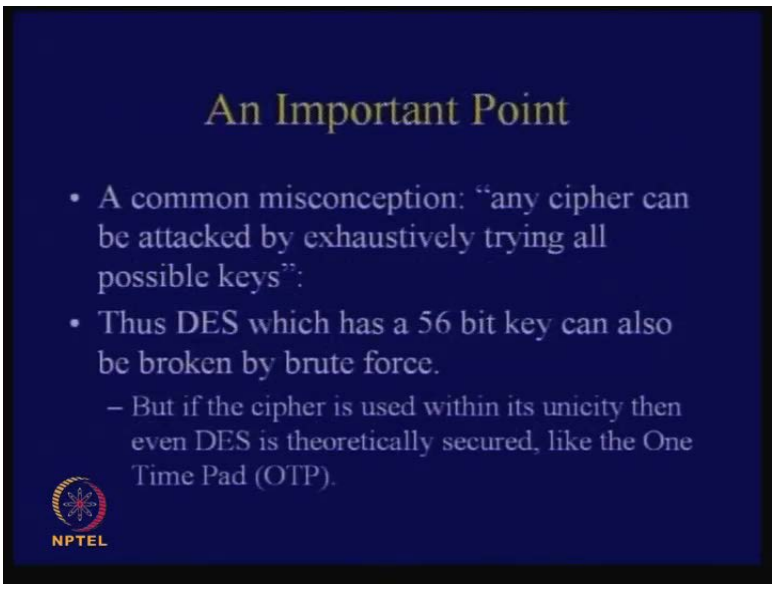
No, no. So, the idea is as follows, that is, you take the ciphertext, consider a shift cipher, so you have got a unique value of the key, so you find out the value of the key, you, so what you do is that you decrypt it back and then you check, that whether the plaintext makes sense or not? So, what, 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 he, 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.

Was that minimum number or maximum number?

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

(Refer Slide Time: 05:09)



So, therefore, a common misconception is that any cipher can be adapt by exhaustively trying all possible keys; this is a very common misconception. So, what you would 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 power of 56 computations, so you should be able to break it; so, the idea is that. But the, if the cipher is used within its unicity distance, then even, an all power adversary cannot break the cipher. It is because of the strategy, the strategy is therefore, what?

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, rest are spurious keys.

What is the significance of unicity distance sir?

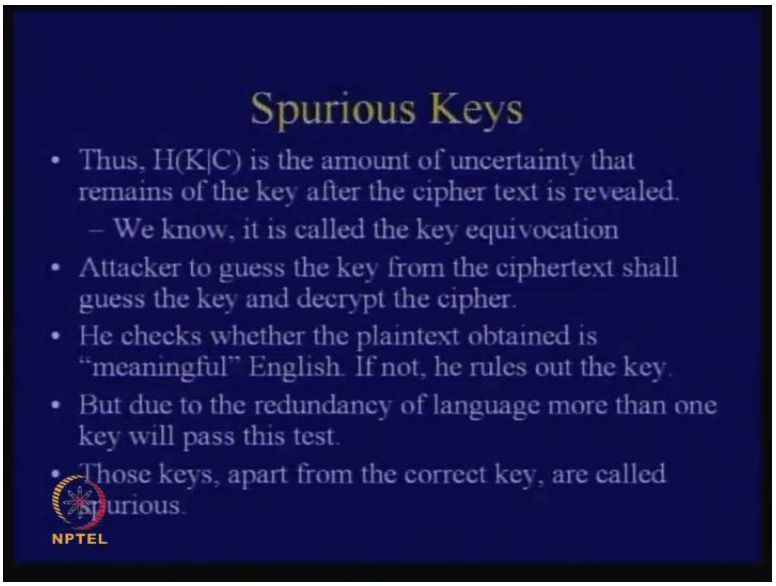
Yes.

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 adversary, then I would like to use the key only for set two times, after that I would like to change the key, if its unicity distance is true. So, therefore this is what I have just described.

(Refer Slide Time: 06:36)

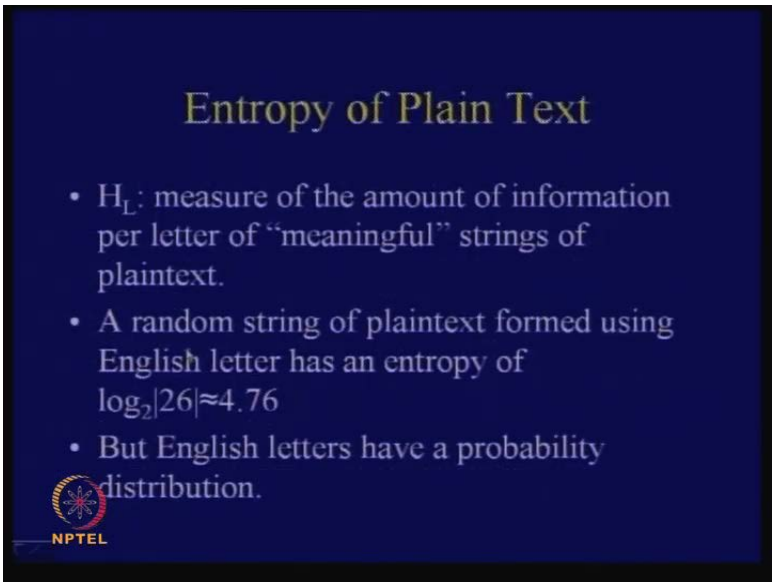


So, HK given C is the amount of uncertainty that remains of the key after the ciphertext is revealed. So, we know that, we know, it is called the key equivocation; we have already defined that

So, what the attacker does is that he guesses the key from the ciphertext and he shall guess the key and decrypt the cipher. So, what he does next is that 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)

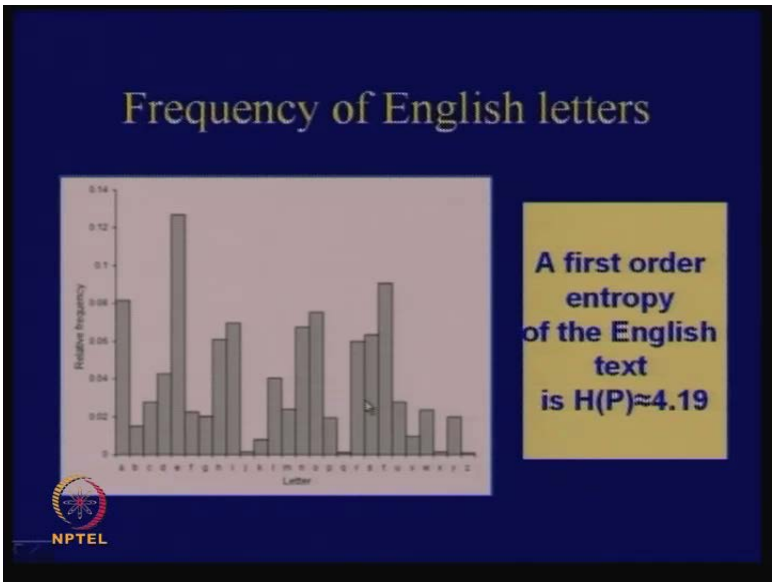


So, then we come to something which is called, we have already discussed about, what is entropy? We will be using this, rather, we will be using entropy to find out, or find out the minimum bound of the spurious keys. So, we will try to find out a lower bound of the number of spurious keys.

So, therefore, so, therefore, consider H L and 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 a random string of alphabets. So, there are 26 letters and if all of them are equally likely, then what is the entropy? It is equal to log 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.

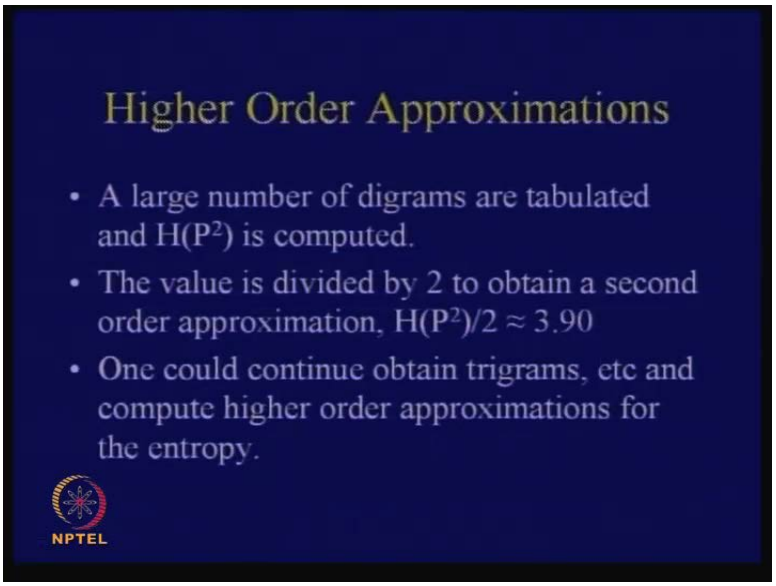
(Refer Slide Time: 08:17)



So, this was, this is a sort of a graph, which shows, how the English language characters, in general, vary. So, therefore, if you just do a first order entropy, that means, you just take one particular alphabet among and consider its probability and then you feed into the formula of pxi, log pxi negative and sigma, then your first 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, the first order entropy is not enough; so, what you do is that, I do a second order approximation.

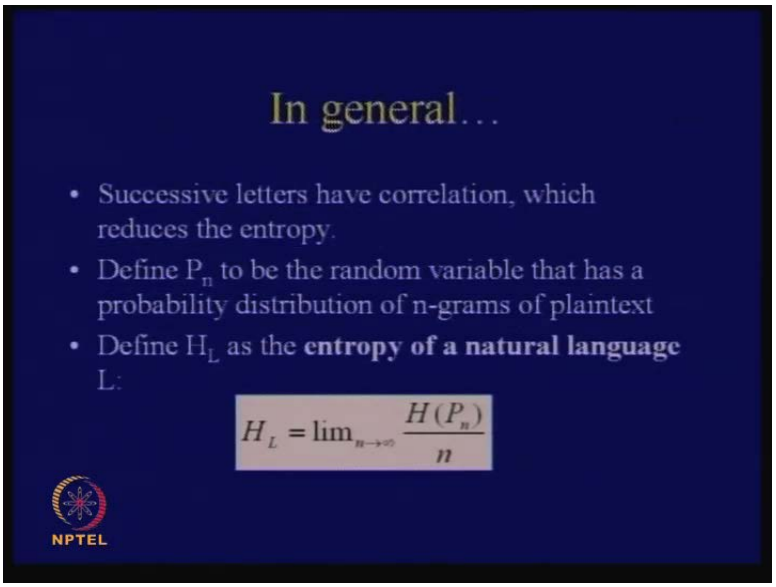
(Refer Slide Time: 09:09)



So, in second order, what I do is that I compute all possible diagrams, I find out their probabilities and subsequently, their entropy and then I divide that HP square 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 HP 3, and then again, divide that by 3 and similarly, you can do higher order approximations of the entropy.

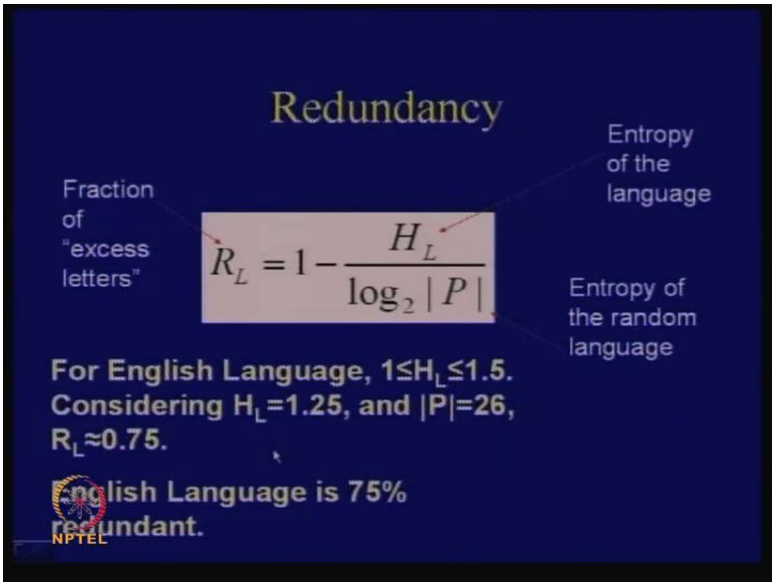
(Refer Slide Time: 09:35)



So, the idea is that in general, the successive letters have got correlation, which reduces the entropy. So, therefore, for example, define, that a 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 HP n divided by n. So, you find out the value of HP n, means, you find out all n-grams, their probability distribution. Fine.

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 found experimentally, that the value of H L, if you consider very high values of n in general, ranges between 1 and 1.5. So, we will find that H L falls between within 1 and 1.5.

(Refer Slide Time: 10:27)

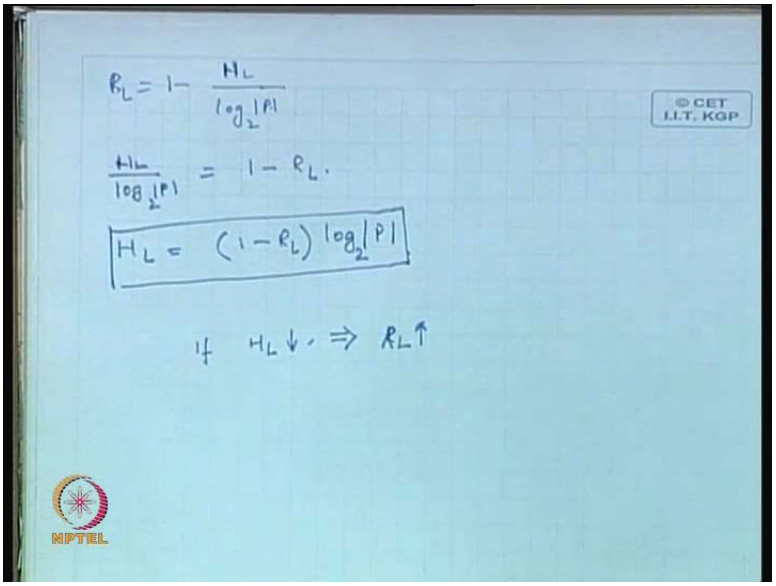


So, so, from there what do we understand? We understand that there is a, some amount of redundancy in the language.

So, in order to understand that better, let us quantize the redundancy by using this parameters, say R L. And 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 form this formula. For example, we will find that R L is equal to 1 minus H L. Therefore, if you just consider a random language, that is, we will just, if English language would have been random, then what would have been a entropy of H L? It would have been log P base 2.

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: 11:42)



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, if 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 increases.

So, therefore, if the entropy reduces of the corresponding language, that is, equivalent to saying that the redundancy of the language has increased. Therefore, this formula is able to capture this intuitive result.

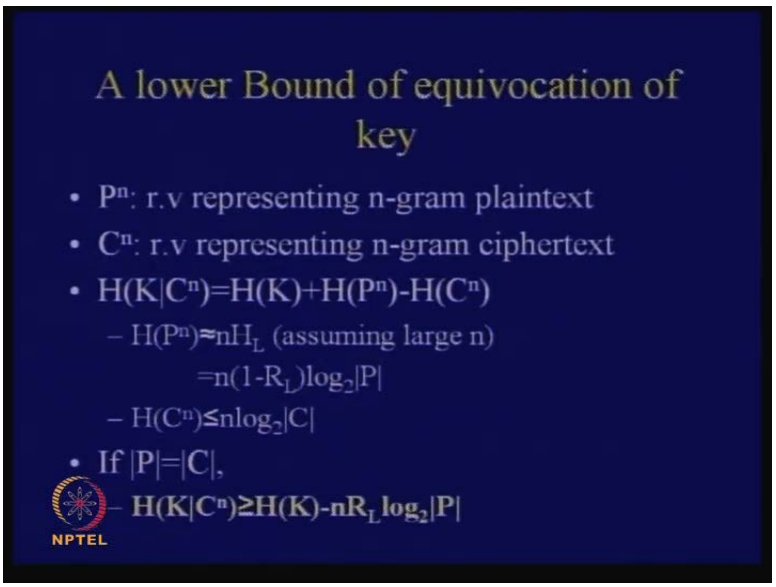
So, let us, therefore, the, let us consider what is the corresponding redundancy of an English language? So, we will have to quantize that. So, if you find out, we will find out, that H L lies between 1 and 1.5, so consider that H L is equal to 1.25 and you know that the number of plaintext characters is 26.

So, therefore, if you feed into this formula 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 only means is, that if you do, for example, a Huffman coding, then essentially, you are expected to get such a kind of compression.

So, therefore, that is the idea of redundancy of a language and that is precisely the reason why cryptanalysis is favored. If you had 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 serves as extra information to the attacker.

(Refer Slide Time: 14:07)



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, I have already

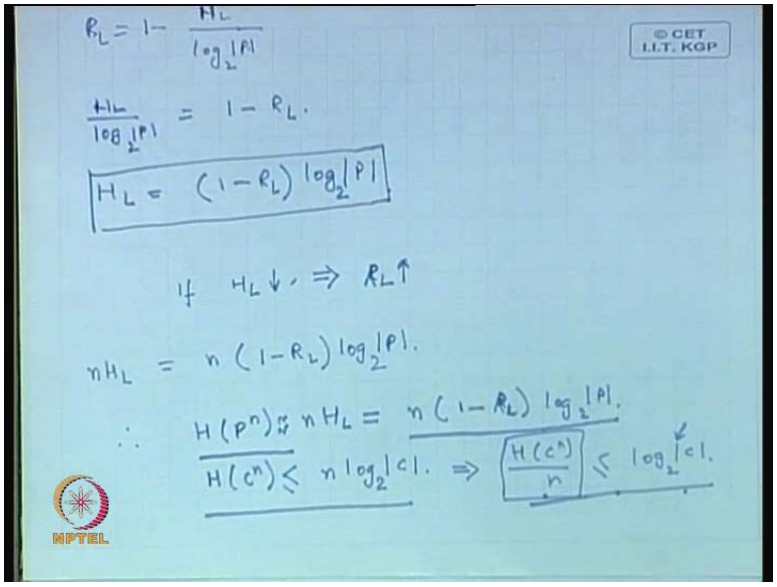
defined n-grams, so consider P n and R n and P n and C n to be two random variables, defined to represent n-grams of the plaintext and n-grams of the ciphertext.

So, all of us know already this formula, so it is, what besides HK given C n is equal to HK plus HP n minus HC n. So, this formula, we have already proved in the last day's class.

Yes.

So, what is HP n equal to? So, HP n, we have defined from the definition of HL. If the value of n is quite large, you can approximate that by nH L, and you know that H L by my previous result was equal to 1 minus R L log cardinality of P base 2.

(Refer Slide Time: 15:12)



So, therefore, if I would like to calculate the value of nH L, I would have just, I need to multiply this particular thing by n, so I get n into 1 minus R L into log of cardinality of P base 2; so, that is the value of nH L.

So, therefore, I can say from here, that HP n, which is approximately equal to nH L, I can write that as, equivalent, equal to n into 1 minus R L into log of cardinality of P base 2. And the next thing that I 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.

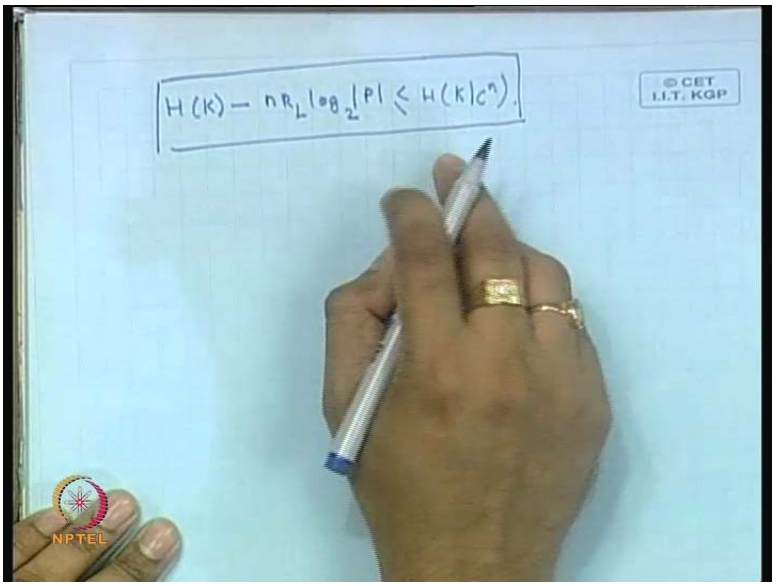
So, all of us know this fact because this essentially, captures the entropy of a random cipher text. So, whatever be it, the entropy that is divided by n, should obviously be less than a random thing. So, the x has got the, you know, the maximum entropy, most uncertainty.

So, therefore, this value is obviously lesser than this, so you will be using these two bounds in our calculation. So, one bound is given by H of C n is less than equal to n log cardinality of C base 2 and the other approximation is H of P n is approximately equal to n into 1 minus R L log cardinality of P base 2. So, we will plug these equations, these two things, into our, into our equivocation formula, that we had.

So, the formula that we had was, H of K given C n is equal to H of K plus H of P n minus H of C n. So, we have a fair amount of estimate of HP n and HC n, so if you plug that, we get this value, that is, H of K given C n is greater than equal to H of K minus n R L log cardinality of P base 2. So, do you see that? So, you see that if I subtract HP n minus HC 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 that both of them are English letters for example, then the cardinality of P and cardinality of C are the same things; so, there is a same value. Therefore, they cancel each other and we have got a lower bound of HK given C n, which says that HK given C n is definitely greater than HK minus nR L log of cardinality of P base 2.

(Refer Slide Time: 18:13)



So, therefore, let us remember this formula, so, because we will be needing this later on. It says that H of K, if I write it in the other way, minus n of R L log of cardinality of P base 2 is lesser than equal to H of K given C n.

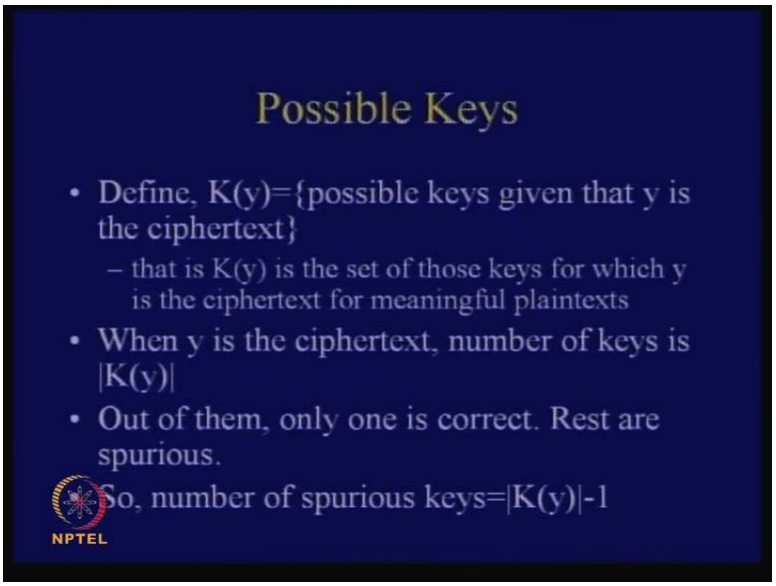
So, now, we will try to prove an upper bound of HK given C n, that is, HK given C n should be lesser than equal to some term. So, from there we will try to find out the quantized value of the spurious keys.

So, till this part is clear?

Sir in this independent of this identity

Yeah, we are not assuming anything of the key. So, for example, we have 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 is got independent of that information.

(Refer Slide Time: 19:05)



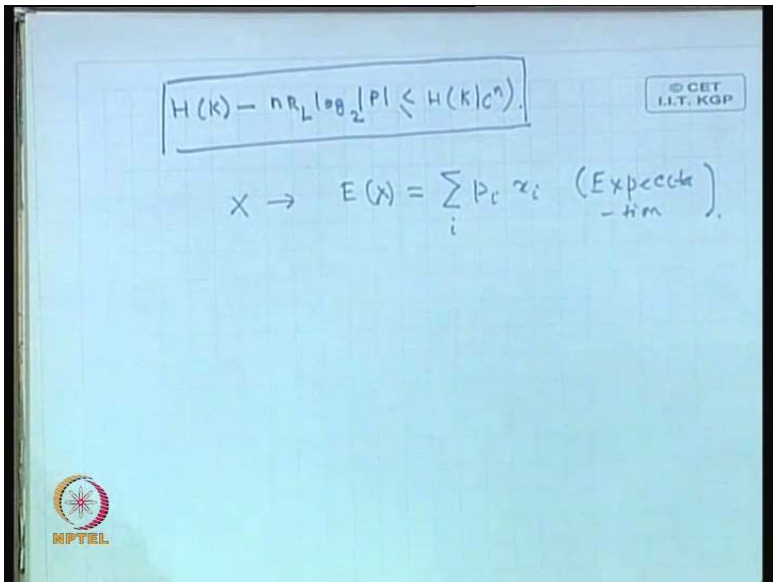
So, therefore, consider possible keys. So, therefore, Ky, it has defined Ky to be the possible keys given that y is the ciphertext; so, define this set. So, what does it mean? It is that possible keys, given that y is a cipher text. So, I have already defined that what it means, it means that Ky is a set of those keys for which y is the ciphertext for meaningful plaintexts.

So, therefore, as I told you, that a, that is, a cryptanalyst takes or an attacker takes the ciphertexts, assumes a value of key and finds out those keys are registers, those keys for which the plain text is meaningful, and that is denoted by the set Ky. Therefore, the Ky set holds those keys for which the corresponding plain text is meaningful.

So, out of these keys, how many is, how many are spurious keys? Cardinality of Ky minus 1, because only 1 key is the actual key, rest are spurious. So, therefore, we know, that when y is the ciphertext, number of keys is modular of Ky; so, out of them only 1 is correct, so rest of them are spurious.

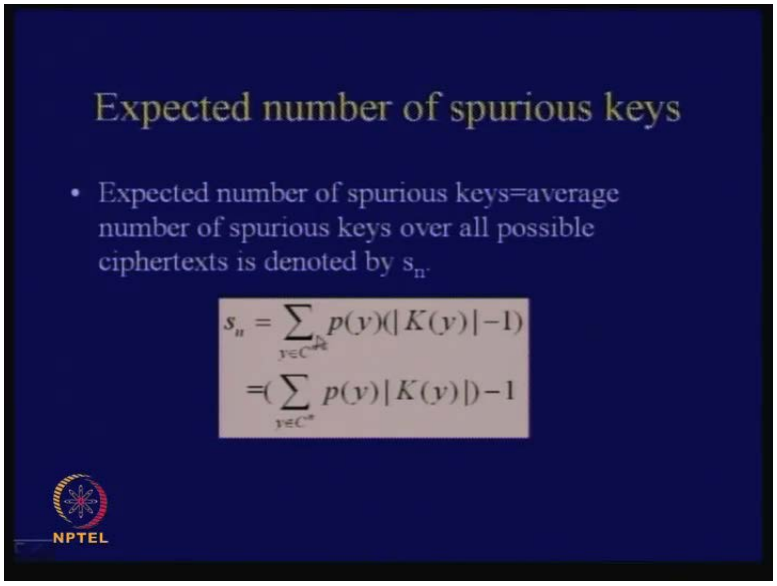
So, the number of spurious keys can be found out by cardinality of Ky minus 1. So, what is the expected size of cardinality? So, you know, that this is actually a distribution, so therefore, in order to calculate the expected value of a random variable, what do we do? We multiply the corresponding value with its probability and dual sigma.

(Refer Slide Time: 20:35)



So, I guess, we know this result, that if there is a random variable x, then its expected value Ex is computed by sigma of its corresponding probability into 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)



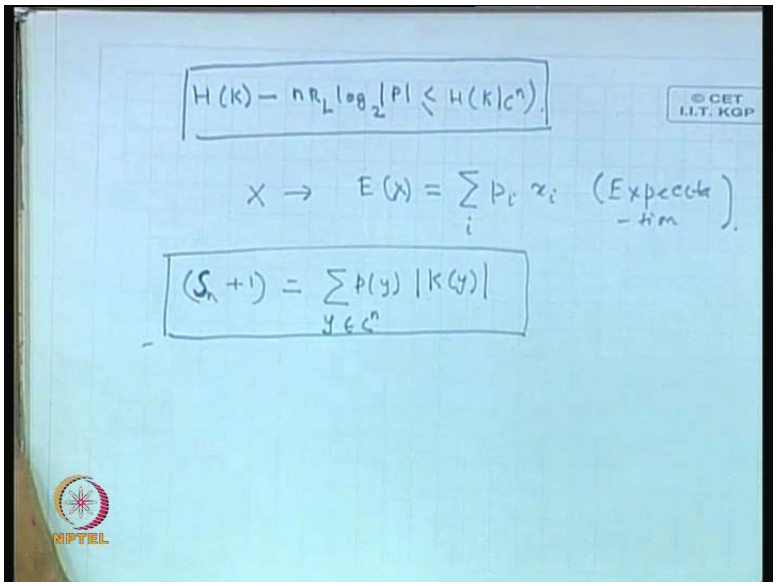
So, we apply this and we find out the expected number of spurious keys. So, what is the expected number of spurious keys here? It is the average number of spurious keys over all possible ciphertext and this is denoted by the variable S n.

So, S n is nothing but sigma of cardinality of Ky, I mean, we have just multiplying K cardinality of Ky 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? That is py, and that is varied for a done, that is, calculation is done for all possible ciphertexts.

So, in a, if I just simplify this formula, then I can actually multiply this sigma, this py with Ky and I obtain that, this if I distribute this py over 1, then I obtain sigma of py. So, what is sigma of py over all possible ciphertext? It is unity, 1; so, I get sigma of py multiplied with the cardinality of Ky and that from there, I subtract the value of 1. So, this gives me what? This gives me the expected number of spurious keys.

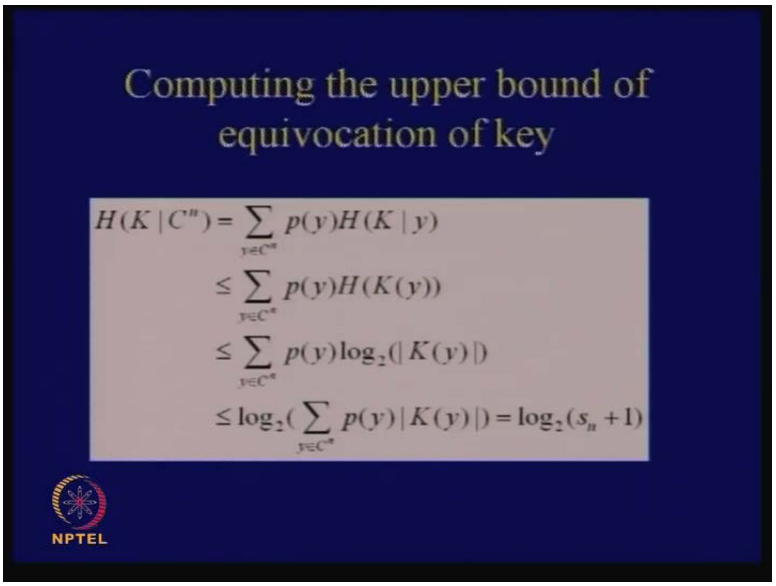
So, therefore, from here I can write, that S n plus 1, I can just reorganize this equation, I can write S n plus 1 is equal to this particular thing.

(Refer Slide Time: 22:22)



So, therefore I can write like, S n plus 1 is equal to sigma of py cardinality of Ky, where i varies over all possible ciphertext. So, this is actually also, which have this, a, this also we can put down from the definition of S n.

(Refer Slide Time: 22:47)



So, therefore, the, if you need to calculate the upper bound of the equivocation of key, we do further calculation from the definition of HK given C n.

So, what is the HK given C n? So, whereas I told you that there are two 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.

Therefore, from our definition of conditional entropy, we can write that is equal to sigma of py multiplied by H of K given y, so this we have actually written in last day's class, you can follow that.

So, therefore, this is equal to, actually this is equal to, should be actual an equal to, so this is equal to sigma of py, I just write this as Ky. Therefore, what does it mean? It means, K given y.

So, that is exactly the definition of H; that is exactly the definition of K y. So, I obtain the sigma, I obtain the, I multiply py with H K y and I take the corresponding sigma. Now, this is less than equal to py multiply with logarithm of cardinality of Ky base 2 and taken a sigma.

When this follows from what I already told you, that if this Ky would have been a, so I mean, if for example this Ky would have been a random distribution, in that case, this

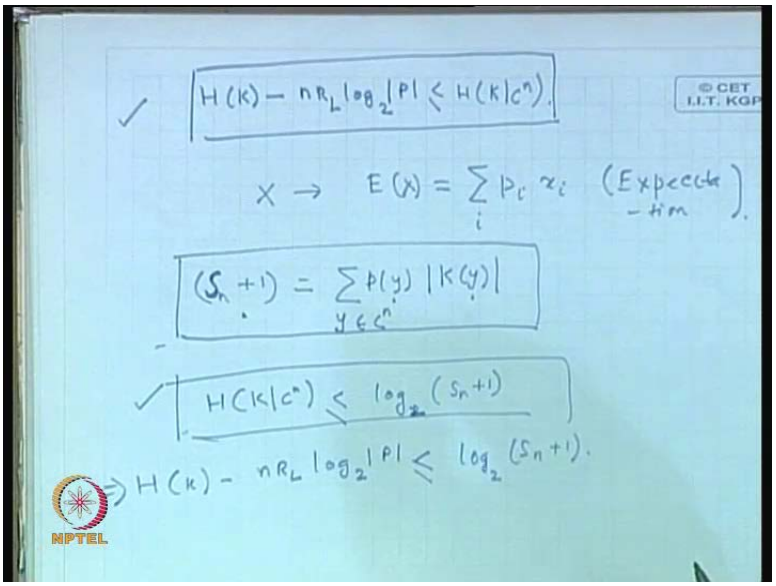
would have been upper bounded, this would have been equal to logarithm of cardinality of Ky base 2.

For any other distribution, this uncertainty is lesser than a random distribution. Therefore, I can actually find out an upper bound and this is the upper bound, then I apply a, a certain result from mathematics is called geneses 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, write this as follows.

So, what is this particular sigma equal to? This sigma is equal to S n plus 1. From this result, you see this, that is, sigma of py cardinality of Ky was equal to S n plus 1. So, we use this result and plug in to that equation, we obtain this. So, you see, that this is equal to sigma of py cardinality of Ky and instead of this I can write, S n plus 1.

(Refer Slide Time: 25:29)

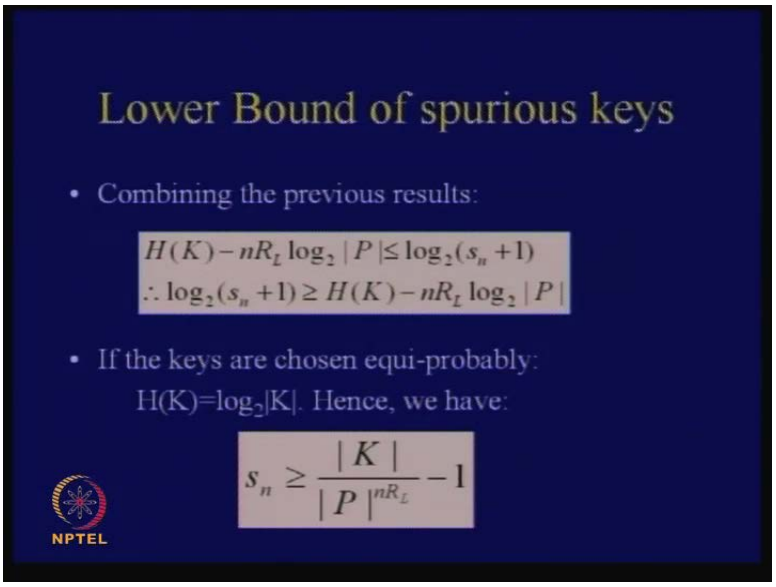


So, what we have proved is that H K given C n is less than equal to logarithm of S n plus 1 base 2.

So, what we have proved is HK given C n is less than equal to logarithm of S n plus 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 HK minus n of R L logarithm of P, logarithm of cardinality of P base 2 is less than equal to logarithm of S n plus 1 base 2.

(Refer Slide Time: 26:19)



So, that is precisely written, what, here. So, why I write that HK minus nR L log cardinality of P base 2 is less than equal to logarithm of S n plus 1 base 2? So, therefore, I can actually reorganize this, and write as, rearrange this and write as logarithm of S n plus 1 base 2 is upper bounded by H K minus nR 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 these are an equal; I can write that HK 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 is equality.

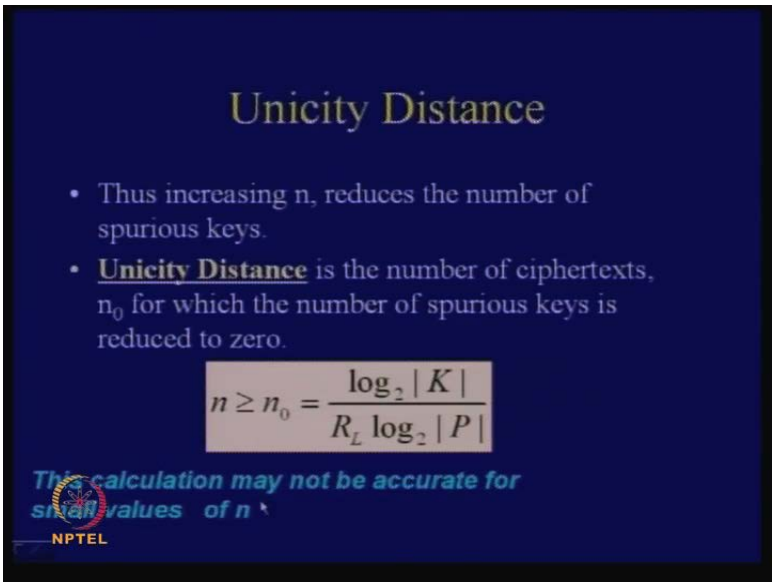
So, I can obtain S n plus 1, rather, S n plus 1 is greater than equal to cardinality of K divided by cardinality of P to the power of nR L; so, this follows from this equation. If you plug in the value of HK and make it equal to logarithm of cardinality of K base 2, so you just see, that if I take this and if I plug here, then I, and I can actually write this as logarithm of cardinality of P base 2 to the power of nR 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. Therefore, I can write that as logarithm of cardinality of K divided by cardinality of P to the power of nR L base 2 and the left hand side, I have got S n plus 1.

Therefore, since I have got two logs on both sides and we have got an increasing function, I can actually write, that S n plus 1 is greater than equal to cardinality of P, cardinality of K divided by the cardinality of P to the power of nR L.

So, what do you obtain from here? What is the objective? The objective is, what is, he note, what is the value of n? What was n? n was the number of ciphertext that I have provided. So, therefore, now I would have, like to make the number of spurious keys equal to 0; that was the objective, finally.

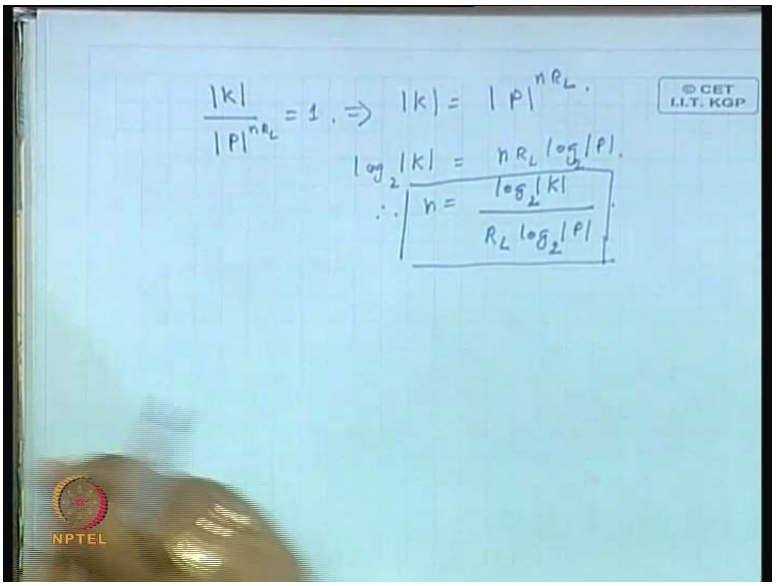
(Refer Slide Time: 28:25)



So, unicity distance says that the, thus the increasing n, so we 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, in unicity distance, is that particular number of ciphertext, so I call it n 0 for which the number of ciphertext or rather, number of spurious keys is actually reduced to 0. So, 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: 29:13)



You see this, that is, if this was equal to 0, then I would have obtained, we would have obtained what? We would have obtained that cardinality of K divided by cardinality of P to the power of nR L is equal to 1. So, that means that cardinality of K is equal to cardinality of P to the power of nR 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 nR 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, for unicity distance would be greater than that. Therefore, if I would provide you more and more cipher text, then actually your number of keys would have been 0.

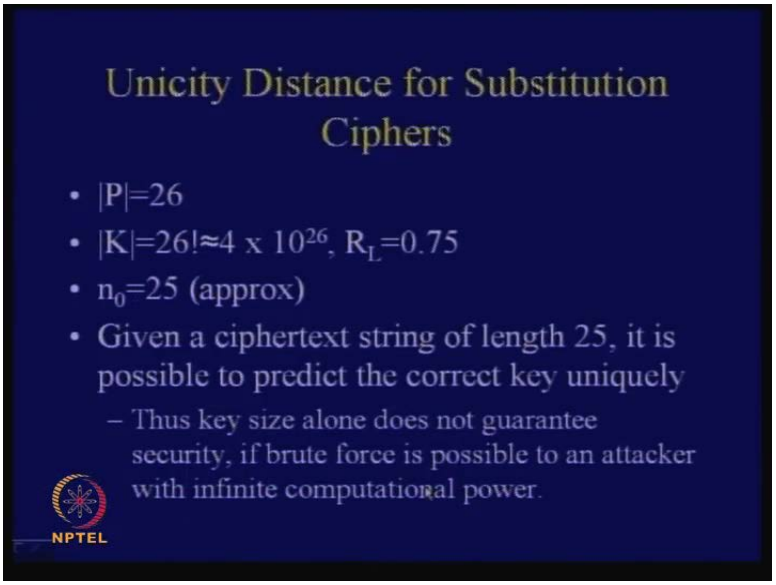
Therefore, if I use a cipher within this unicity distance, then the number of spurious keys will not be 0 that means, the attacker is not able to exactly find out the unique value of the key and we have, we have, so throughout our calculation, we actually consider an unbounded adversary.

So, even an unbounded adversary would not actually find out the actual value of the key, not the unique value of the key, but he will have a set of possible keys. And unicity distance is actually, that number of ciphertext for which this number of spurious keys

just becomes equal to 0. Therefore, we obtain this particular lower bound; therefore, this is a lower bound of the number of spurious keys, number of, for the, for the lower bound for the unicity distance.

So, beyond that, the attacker is actually able to find out the unique value of the key. So, note that this calculation may not accurate for small values of n, why? And because my original H L definition relate upon the fact that limit n tends to infinity, that were the assumption that I made, therefore, this may not be very much true for n equal to 1, 2 or so on. It should be fairly o.k. for large values of n.

(Refer Slide Time: 31:23)

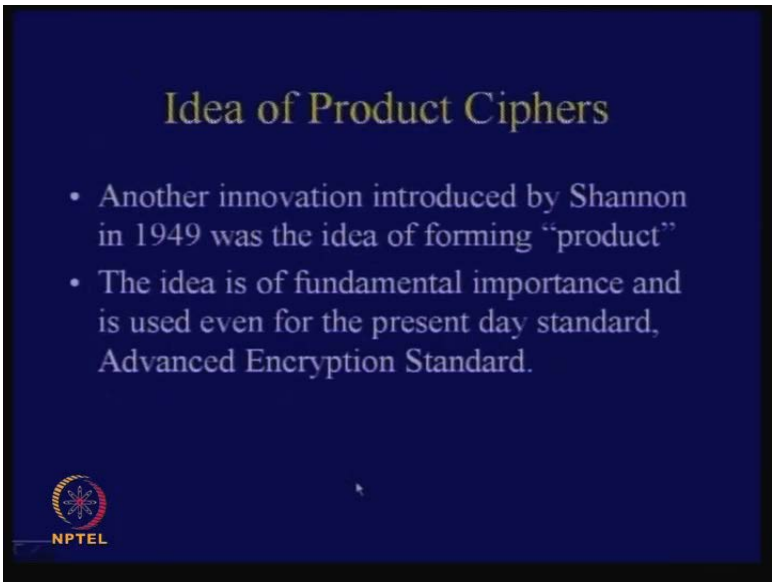


So, let us do an example calculation of, with the substitution cipher. So, we had number of plaintext characters equal to 26, so the cardinality of P was 26, cardinality of K was 26 factorial, so that was around 4 into 10 to the power of 26, fairly large value of the key. So, here R L, if you assume is equal to 0.75 of English language, then if you plug in, will find that n 0 is approximately equal to 25. So, it 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 guaranty security, if brute force is possible to an attacker with infinite computational power. So, an attacker who has got an unbound, I mean, infinite computational power, for him such a big size of key, he requires just 25 ciphertext actually, find out the value of the key.

All these mirrors are, of course probabilistic, but it will match with the actual result quite closely.

(Refer Slide Time: 32:35)

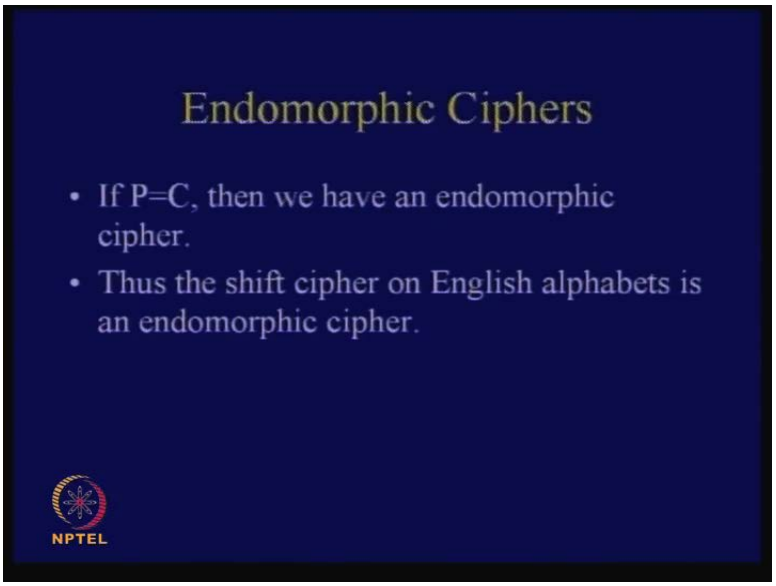


So, this, with this we essentially conclude this part of unicity distance, but we will conclude the remaining part with an idea of product ciphers.

So, actually, if, after this we will actually, go start talking about real ciphers like block ciphers and stream ciphers, but before this I would like to mention about the idea of product ciphers. This was actually, also mentioned in Shannon's paper and that is why, it is called a seminar paper. It was mentioned, as old in 1949 and he get the idea of forming products. So, the idea is still fundamental because even 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, that, through that lot of things becomes meaningful; lot of things, which we will see in our future ciphers, will actually become meaningful. So, let us try.

(Refer Slide Time: 33:28)

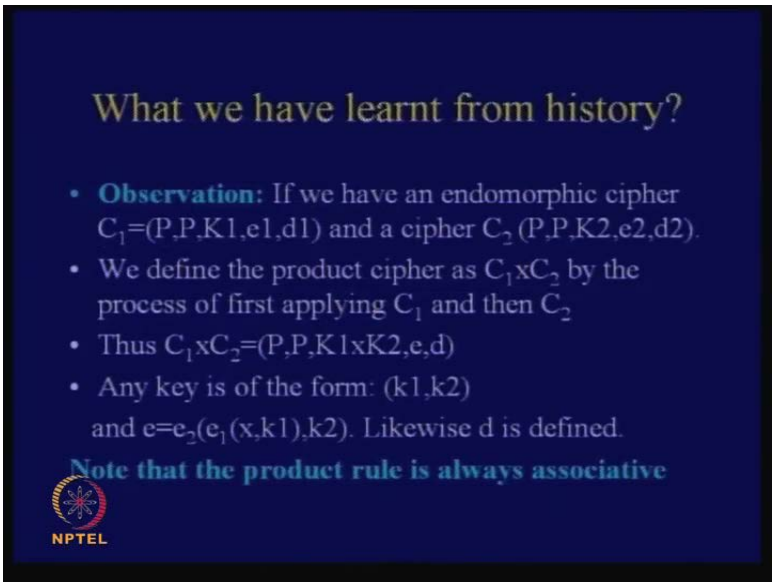


So, see, so where, before that I would just like to, in order to simplify our life, let us, I will just coin a term, endomorphic ciphers. So, what is an endomorphic cipher? Endomorphic ciphers are those ciphers, for the plaintext and ciphertext are the same sets.

Say, for us, normal substitution cipher where plain text and cipher text, where just English language, that means English characters, so if P and C are the same, then we have, what is called an endomorphic cipher.

So, therefore, the shift cipher of an, on English language, on English alphabets was an example of an endomorphic cipher.

(Refer Slide Time: 33:58)



So, consider an endomorphic cipher and let us try to understand certain things from history. 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. So, the plaintext set and the ciphertext sets are the same thing, so I write (P,P) and then followed, follow that with K1 because K1 denotes the key set of the encryption function C 1.

So, if the cipher C 1 and we have got an encryption function e1 and corresponding decryption function d1; we also have a cipher, which is called C 2 and I denote that with (P,P,K2,e2,d2); so, that means what? The K2 is the set key of the keys for the second cipher, e2 is the corresponding encryption function and d2 is the corresponding decryption function.

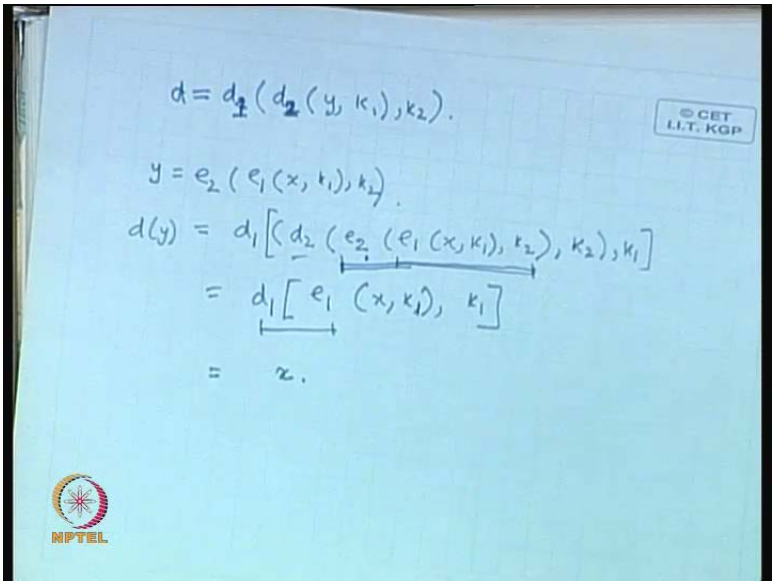
So, let us try to understand or define, what is meant by the product cipher C 1 cross C 2? So, what does C 1 cross C 2 means? It just means, like as you know, that we apply two functions, after, one after the other; therefore, if I say, C 1 cross C 2, it means, that first I will apply C 1 and follow that with the application of C 2.

So, I think, we have seen this in the case of composition of functions in discrete structures class. Therefore, they exactly, precisely, in similar kind of thing, so you see, C 1 cross C 2, I would define as P cross P because it is still endomorphic. You see, why it is endomorphic?

Yeah, because I mean, that repeated application also keep, still keeps its endomorphic property, so it is still endomorphic, but here, key set is the Cartesian product K1 and K2, that is, K 1 cross K 2, and we have got the corresponding encryption function e and decryption function d. So, any key I can write in the form of an ordered pair K1 cross K, (k1,k2), I can form ordered sets like that.

So, the encryption function is defined as e equal to e2 and, but initially you take the plaintext x, you take the key K1 and you apply the function e1. Subsequently, you choose a key K2 and you apply the encryption function e2.

(Refer Slide Time: 36:11)



So, what will the corresponding decryption function look like? The corresponding decryption function will look like this, it will look like d equal to d 2 (d 1(y, K 1)) follow 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. 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 have obtained that where I started with. So, you can see this because of this.

Yeah, it will be first d 1 and followed that with d 2, yeah, so that you can obtain from the decryption function because what you do is, that you take the corresponding, so you

write this e 2(e 1(x, K 1), K 2)), so that is my y. So, I take this as y and I apply my d over that.

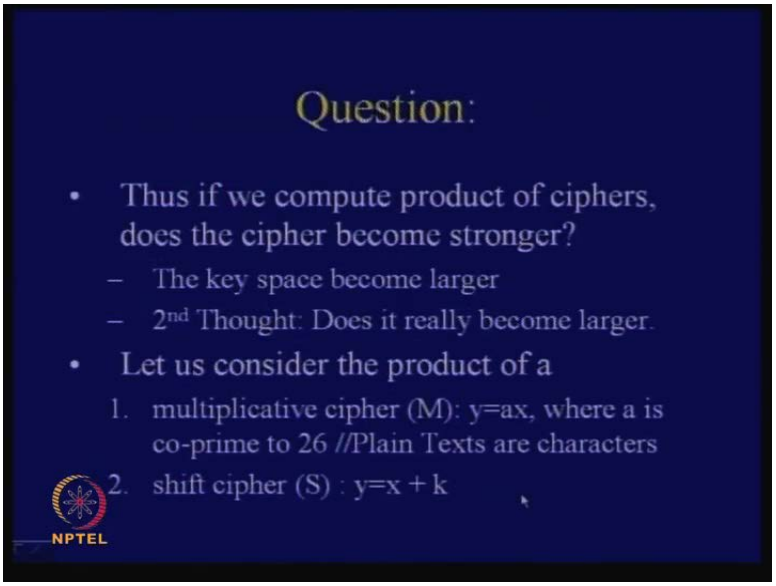
Therefore, what I do is that I want to compute this, I want to compute dy, so for that I apply d 1 d 2 over e 2(e 1(x, K 1), K 2)). And then, so this is my, therefore, this is my e 1, so therefore, what I do is that I have actually encrypted, so this is my scope of the function e 2, then I apply d 2; so, in order to, encrypt, 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, K 1),K 1).

So, again this d 1 and e 1 cancel each other and I finally obtain that the value of x. Therefore, you are 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)



So, the question is that, if you can compute product of ciphers, does the cipher become stronger? That is what is most important.

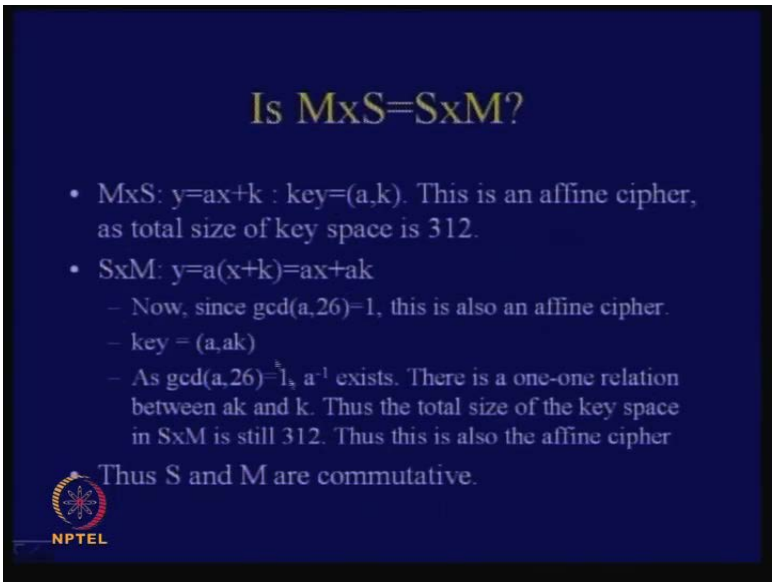
So, I take two small ciphers and I compose them, I compute the product. Thus, the key, thus is, thus, the cipher become stronger, that means, thus the key space becomes really larger. So, in the initial, on the surface, we have actually (K 1,K 2).

So, the product size should increase, but in a second thought, does it really become larger? So, in order to understand that, what is your opinion on that, will it really becomes larger? Will the key size become larger? So, actually, not always.

Let us try to consider one example, I guess it be very 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. 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: 40:05)



So, now you consider, that for example, that you have got M and you have got an S and what we do is that we 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 as, 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 the double (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.

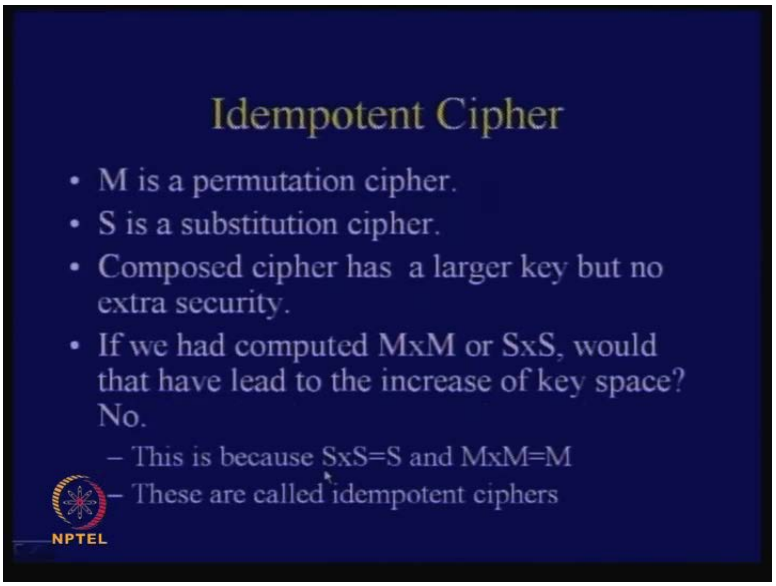
So, now you consider S cross M; S cross M is equal y is equal to a(x plus K); so, that is, you can also write that ax plus ak. Now, you know, that gcd of (a,26) is 1, therefore, this is also an affine cipher, and the key would be (a,ak), but since the gcd of a and 26 is 1, so an inverse exists. This we discussed and actually, there is a one-to-one relation between ak and k, therefore, the total size of the key space in S cross M is still, 312.

So, you see, that here we have got the key ak 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 ak, since you are doing a modulo 26, also have 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 you call commutative. So, we have got a commutative cipher.

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. 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)

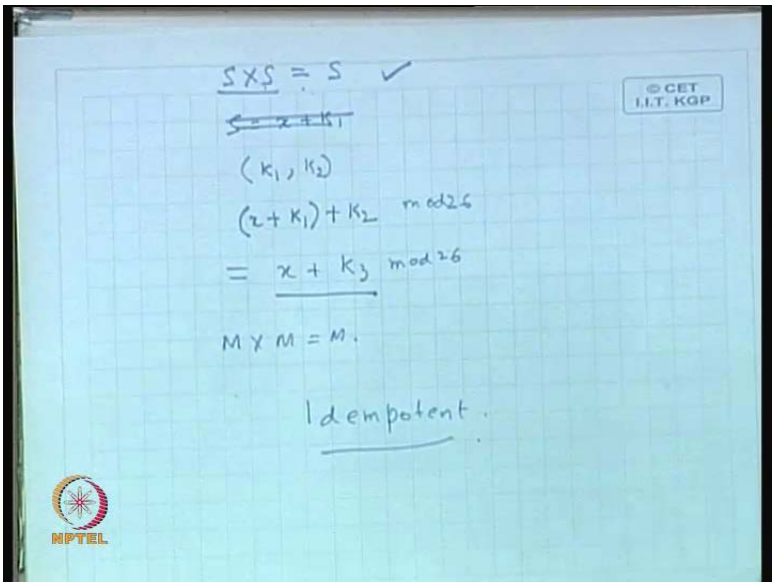


So, then let us see, what is an idempotent cipher? So, therefore, what is an idempotent function? It means, if we apply the same function twice, you obtain that, the same function. Therefore, M is a permutation cipher, S was a case of a substitution cipher and both of them where actually idempotent ciphers. So, a composed cipher has a larger key, but no extra security because M cross M, if it is equal to M, then even composing Ms for, I mean, more than the, more than 1's essentially, leaves you with the same kind of transformation. So, essentially, it does not add to a security.

Therefore, for example, if you have computed M cross M or S cross S, that would not have led to the increase of the key space. So, this is because S cross S and is equal to S and M cross M is also equal to M, and this class of ciphers are called idempotent ciphers.

So, you could easily observe from this fact, that is, if I had done M cross M, what would have be the, would have, what would have that meant? I mean, I would have done ax and then a ax, but my key size would not have still increased because doing S square essentially, does not increase the key space. Similarly, you consider, for shift cipher you do one shift and you do another shift; so, in both the cases, you can represent that, I mean, you represent that that by a third shift. So, do you understand what I am saying?

(Refer Slide Time: 43:33)



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?

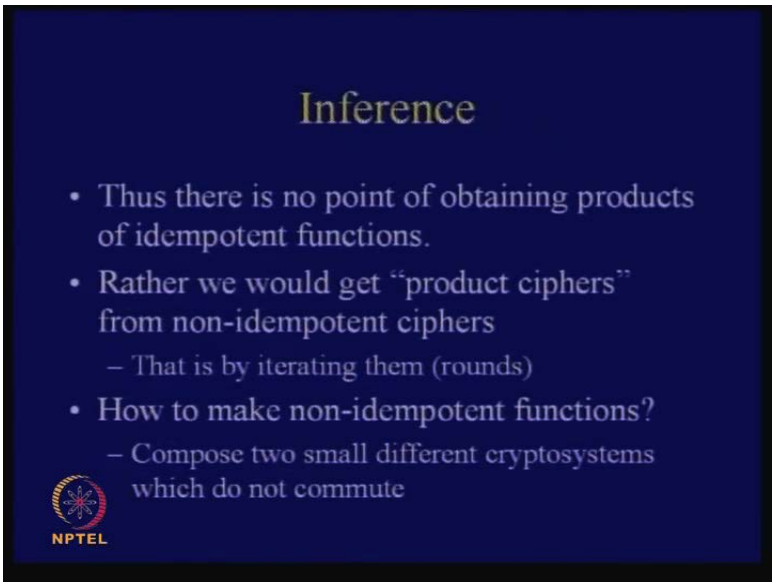
So, therefore, imagine that in the first phase you have got, you have got the function S, I mean, you choose the key as K 1 and in the second case, you choose the key as K 2. So, in the first application of S, I would have computed x plus K 1 and in the second application of S, I would have computed as K 2.

So, 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 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, we can actually show that M cross M is also equal to M. So, both this class of ciphers are something, which we called as idempotent ciphers.

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

(Refer Slide Time: 44:58)

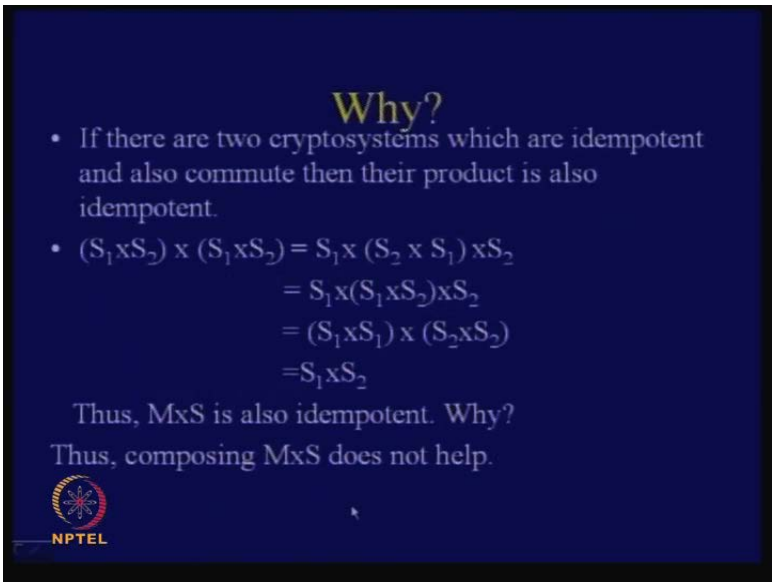


So, actually that, you can observe from this fact. So, therefore, what we are trying to observe is that there is no point of obtaining products of idempotent ciphers; so, if you take M cross M, is the same thing as M. So, that is no point of doing such products.

So, rather, we would get product ciphers from non-idempotent ciphers, that is, by iterating them. So, if we have some non-idempotent ciphers, I would have liked to iterate them and 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, if the idea would be, that compose two small different cryptosystems, which do not commute.

(Refer Slide Time: 45:40)



So, do you follow this? If you do not follow, then this will become clear because of this calculation. So, what I have 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 their product is also idempotent. So what does it mean? It means that if you obtain a function of this class, then if you take, and if you take them and if you still product, rather I mean 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 required to, I mean, required to compute the products of ciphers, which even if they are idempotent, they do not commute. So, that explains this point, that is, compose two small different cryptosystems, which do not, do not commute; so, those kind of ciphers, if you iterate them, will actually make sense. 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.

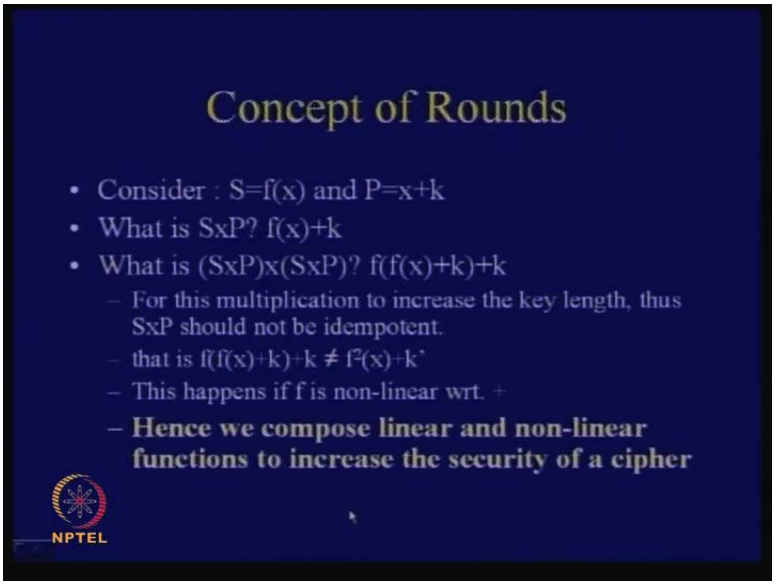
So, S 1 cross S 2 cross S 1 cross S 2. So, I am considering the product of these things. Therefore, so, if I observe that from the associativity, I can write like, S 1 cross S 2 cross S 1 cross S 2 and since this commutes S 2 cross S 1 becomes equal to S 1 cross S 2.

So, now you know, that S 1 cross S 1 is equal to S 1 and S 2 cross S 2 is also equal to S 2. So, what we have obtained is that S 1 cross S 2 and product and multiplying that with S 1 cross S 2, essentially gives you with S 1 cross S 2. So, what does it mean? It means this is an idempotent function.

So, in your previous case, we are proved that M cross S, M and S were essentially, both of them were idempotent. Therefore, can you show, can you understand, why M cross S is also idempotent? Why?

And because we are proved, that M cross M was equal to S cross M, so that means, M and S were commutative and we are also proved that M cross M is equal to M, and S cross S is equal to S, so that is, they were idempotent as well. So, if you commute their products, then essentially, you are left with the same thing. Therefore, computing their products or composing them does not help. So, therefore, you require some other additional quantity, which will help you and that is the idea of rounds.

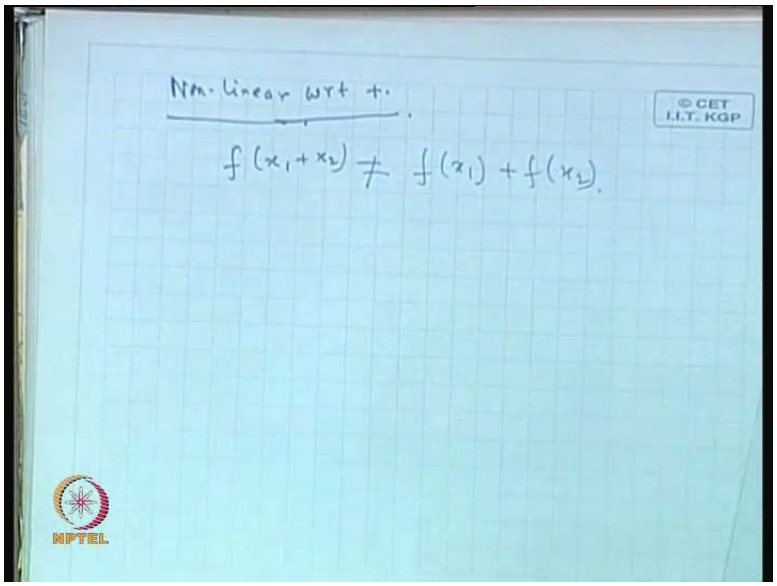
(Refer Slide Time: 48:15)



So, therefore, the idea is that, how can you, I mean, what is that future? So, till now, whatever we have seen, that is a future 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 the brief introduction to that.

So, therefore, consider that instead of, I mean, let us consider these two functions S and P, where P is actually equal to x plus k and s is the output of a function fx. 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: 49:03)



So, it means, that, 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 mean, meant linearity. So, therefore, now consider this function S equal to fC and P equal to x plus k and consider S cross P. So, what is S cross P equal to? It is equal to fx plus k and what is S cross P cross S cross P? So, that means, that if you take fx plus k and then you do a further application of f, and add that with k, so for this multiplication to increase the value of the length of those key, key, so, thus, what is needed?

So, therefore, it is needed that S cross P should not be idempotent. So, if that so, what we require is that f of fx plus k, which is equal to this particular thing or when it added with k should not be equal to f square x plus k dash because if this, I mean, if you had a linear function f, then you can actually have, if this was a linear function, you would have actually distributed this and this would have computed to some f square x plus some value of k dash.

So, that you see is exactly similar with your S cross P function, with some other application of f and it is added with a key K, with a key K. Therefore, what did, therefore this happens only if f is non-linear with respect to plus.

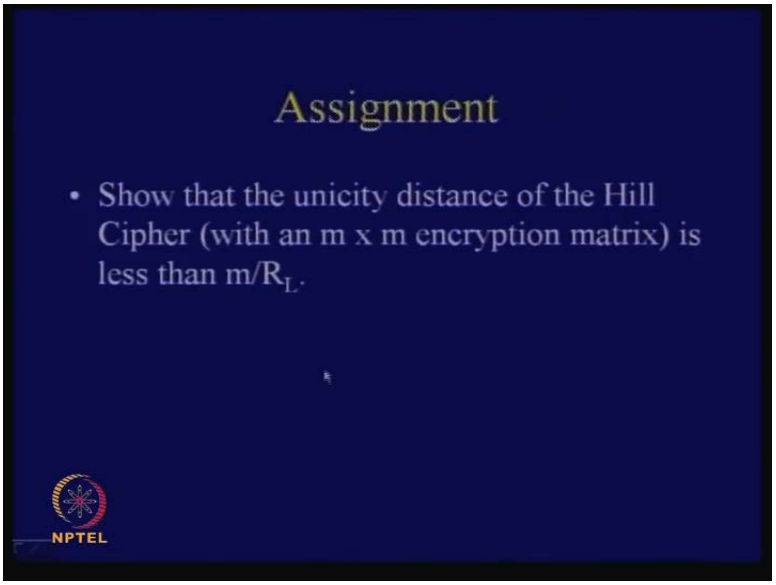
So, if this was a linear function f, then actually this would have distributed and that result that we would have obtained, would have been similar to that of S cross P.

So, the size of this function, whereas 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, what we have seen till now are only linear compounds, linear transformations. We essentially, found out multiplication with a matrix with a linear operator addition with the key, 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: 51:55)

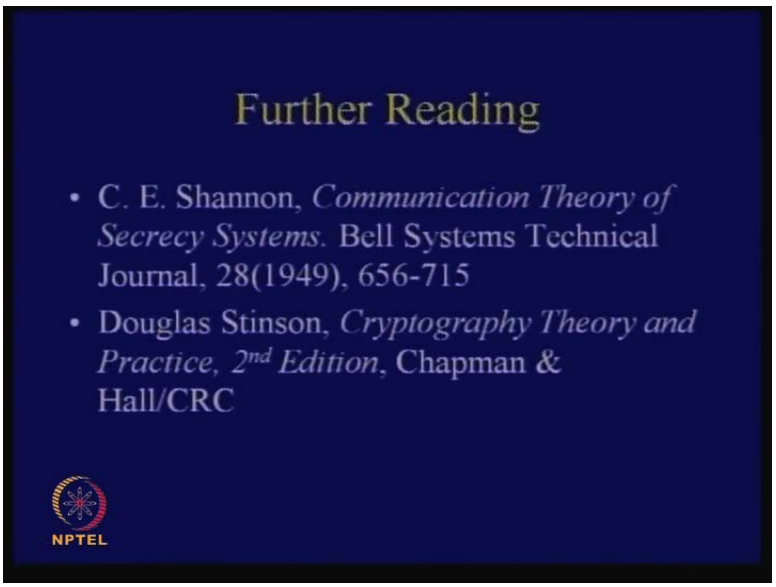


So, so, with this I conclude my talk, but I would like to give you an assignment, which you are supposed to again, do it and submit it on twenty eight, before twenty eight.

I gave you one assignment already, the other assignment is that, show that 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 an m cross m encryption function is actually less than m divided by R L.

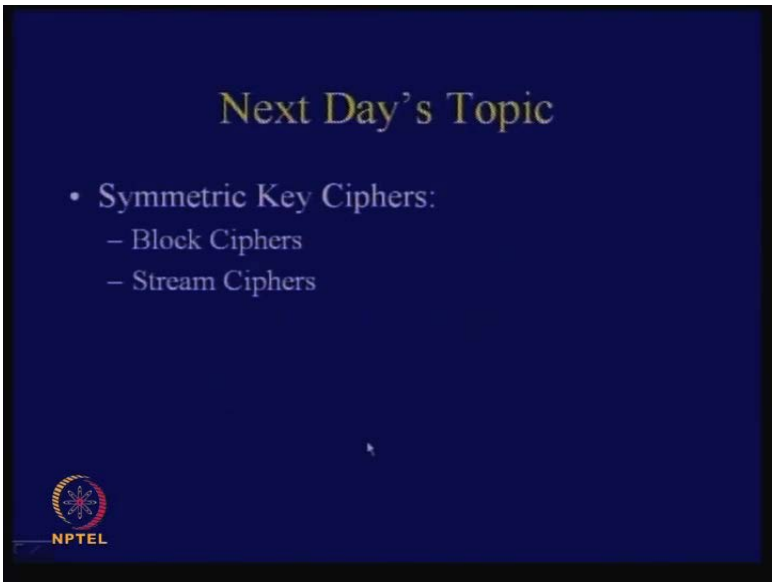
(Refer Slide Time: 52:31)



So, that is an assignment, which is given to you and you can read for the things from Shannon's book, which is, Communication Theory of Secrecy Systems, which is actually a paper; it is a classical paper, so it is 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 second edition book, it is you can follow. So, I have followed that book although third edition exists.

(Refer Slide Time: 52:54)



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

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