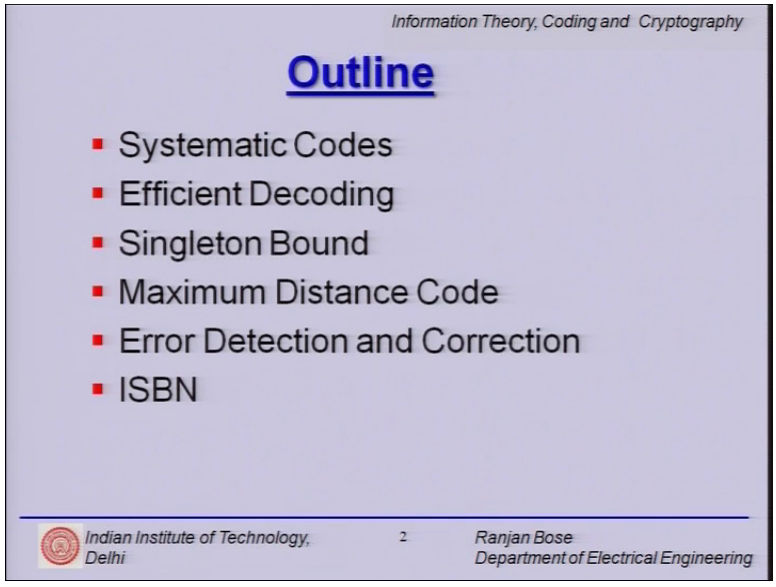
Information Theory, Coding and Cryptography
Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Module - 16
Linear Block Codes
Lecture – 16

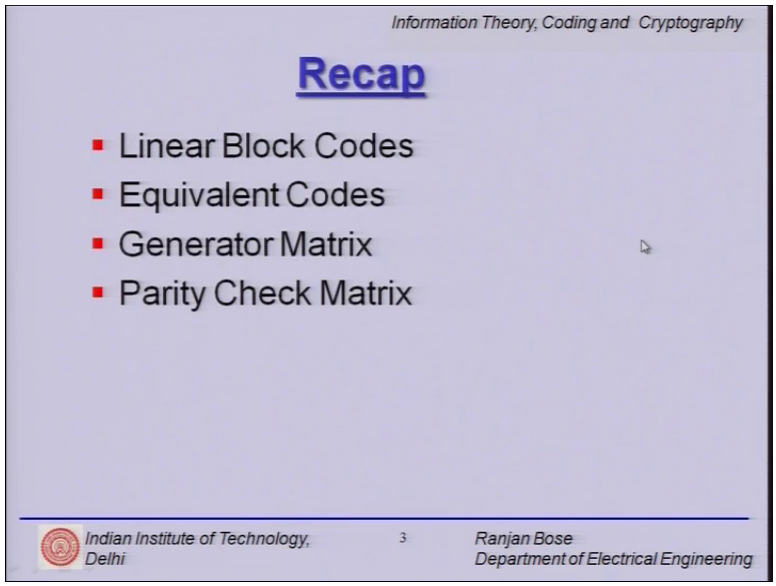
Hello and welcome to the next module on Linear Block Codes.

(Refer Slide Time: 00:31)



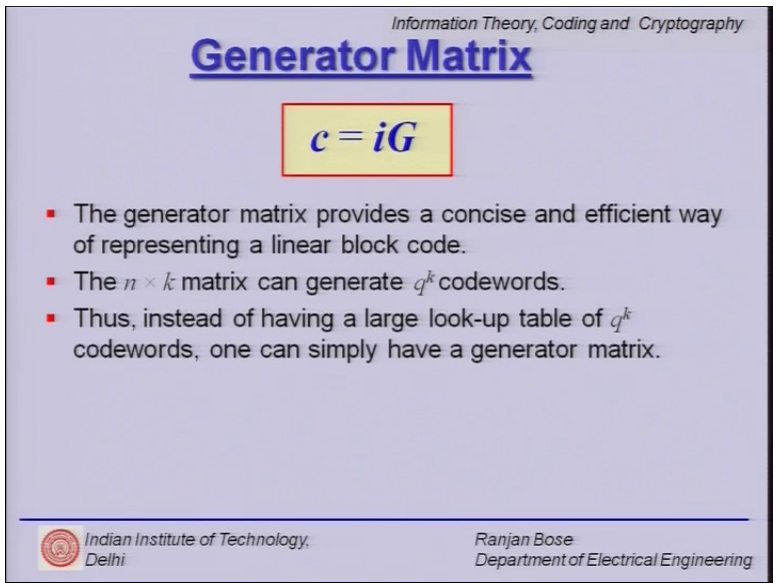
We will start with a brief outline for today's talk, we will revisit systematic codes and talk about efficient decoding, we will look at something called as the singleton bound, then we will move on to maximum distance codes, finally our main aim is to figure out how to perform error detection and correction we will spend some time on that and we will look at some examples; so this is the brief outline for today's talk.

(Refer Slide Time: 01:01)



Let us see what we have already covered so far; we have introduced the linear block codes and talked about the notion of equivalent code, we then looked at generator matrix and the parity check matrix we have already covered this, but we will quickly refresh our memories.

(Refer Slide Time: 01:22)

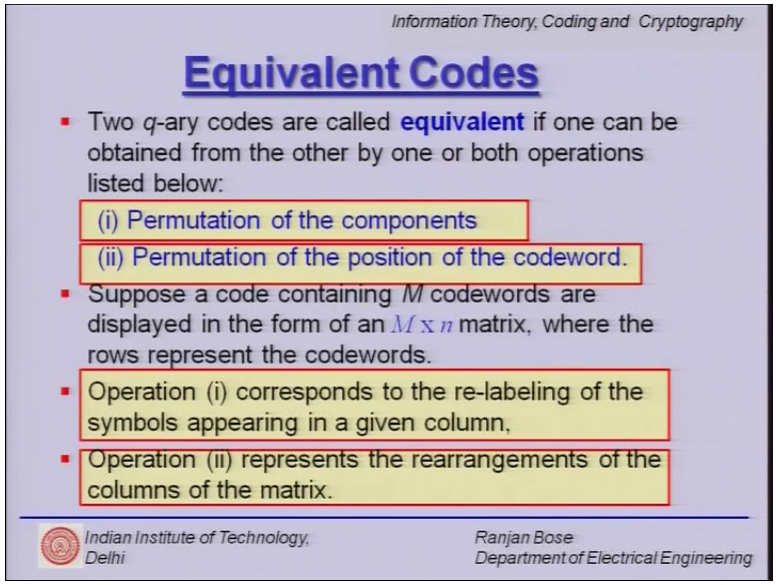


So, we realize that there is a very efficient way to represent a linear block code using the notion of a generator matrix G; what it does is it takes in a vector 1 cross k so k bit long

or k symbol long input vector and it is multiplied with G, which is k cross n and you get a 1 cross n so that is pretty efficient.

It is also a very concise way to represent your linear block code so G matrix is nothing, but an n cross k matrix and it can generate q raise power k codewords this will do away with the problem of putting it a very large lookup table.

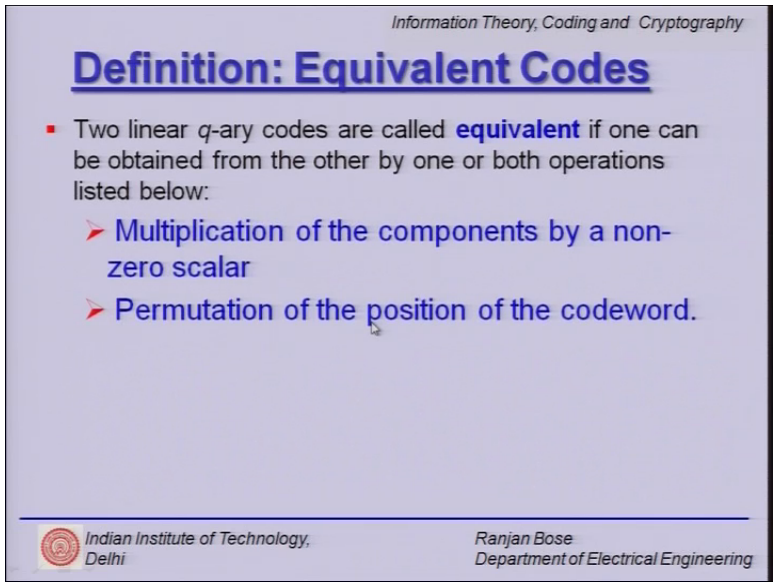
(Refer Slide Time: 02:06)



Then we looked at the notion of equivalent codes and we defined that two q-ary codes are called equivalent; if one can be obtained from other by permutations of the components and permutations of the position of the codewords.

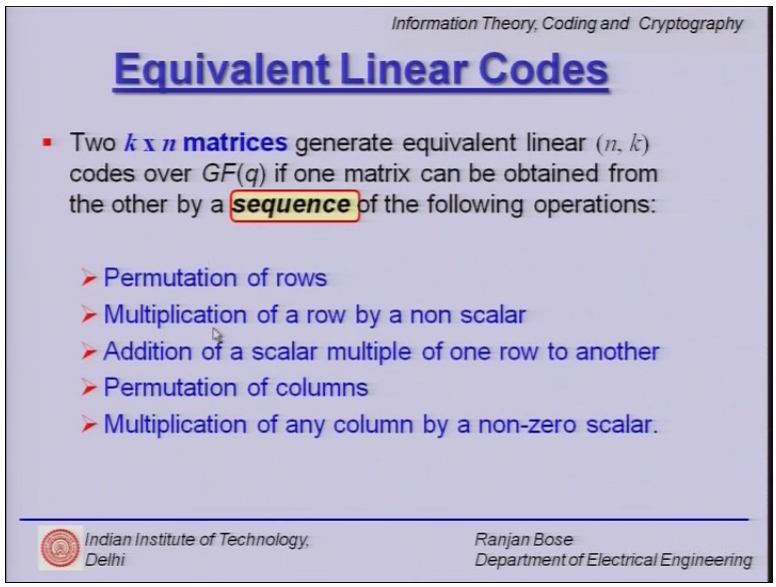
And we said that we can define m codewords in the form of a matrix which is m cross n and then we can do two kinds of operations operation; one which pertains to permutations of the components and operation, two which pertains to the permutation of the position of the codewords; any of them really does not change the distance properties of linear block code and hence the error correcting capability is unaltered hence these are the equivalent codes.

(Refer Slide Time: 02:55)



We looked at the formal definition of equivalent code where two linear q-ary codes are called equivalent codes.

(Refer Slide Time: 03:17)

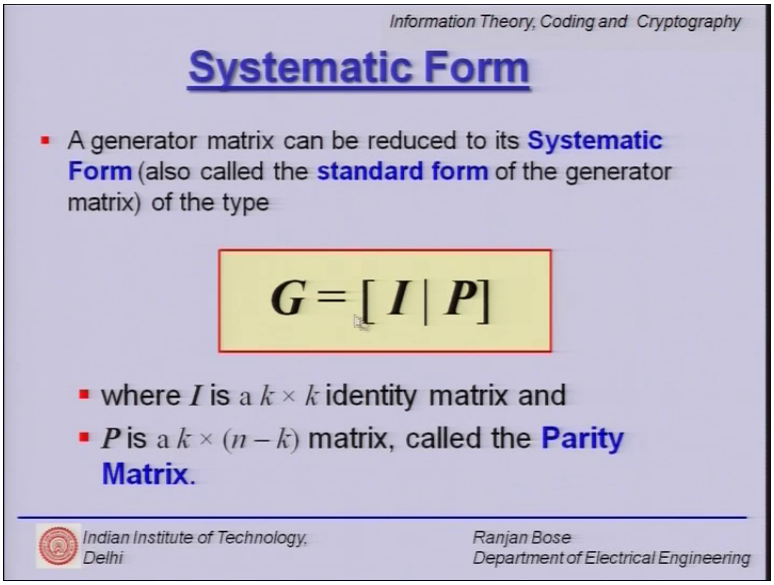


If one can be obtained from one to the other using the generator matrices, and how do we do that; multiplication of the components by a nonzero scalar permutations of the position of the codeword. So if two k cross n matrices generate equivalent linear codes of dimension n comma k over GF q then one of the matrices one of the generator matrices can be obtained from the other using a sequence of operations given as follows. We can

either permute the rows of the generator matrix, multiplication of a row by a non scalar or addition of a scalar multiplied rows with another, sum of two rows, sum of 3 rows replacing another row, permutation of columns, multiplication of a any column by a nonzero scalar a nonzero scalar all of these operations in any sequence if can lead from one generator matrix to the other then these two are called equivalent linear codes.

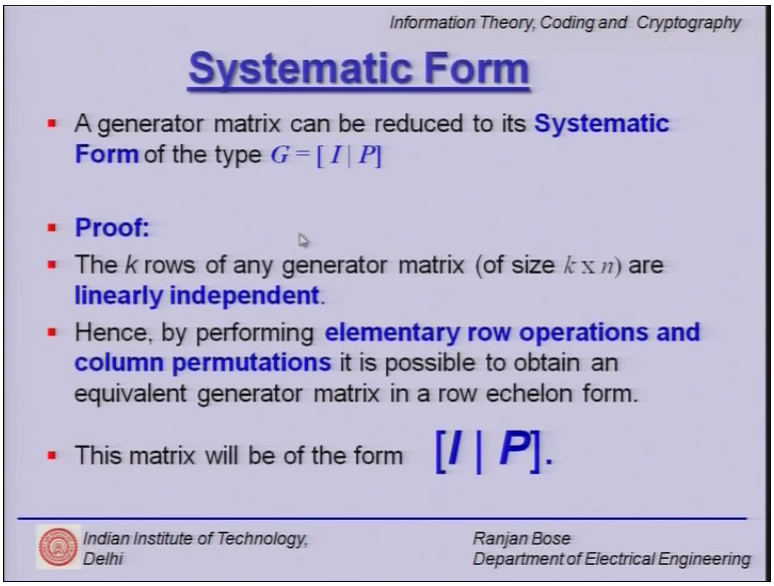
So, we have already studied these so far so what is important is you can have any sequence of these operations.

(Refer Slide Time: 04:15)



Then we got into the notion of systematic form a generator matrixes reduced to its systematic form which is also called a standard form if it is represented as follows; the generator matrix G is partitioned into an identity matrix, which is a k cross k matrix followed by a parity matrix p. So I here is a k cross k identity matrix and P is the parity matrix whose dimension is k cross n minus k thereby making this entire matrix k cross n.

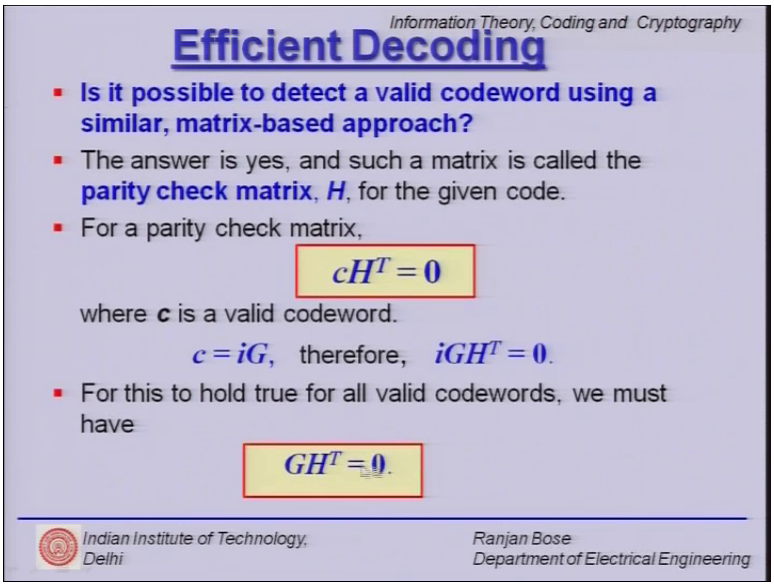
(Refer Slide Time: 04:54)



So, generator matrix can be reduced to a systematic form of this type and you can prove this that this is basically the k rows of any generator matrix are linearly independent we have established that, in fact the rows of the generator matrix forms, the basis vectors which generates the codewords. So hence performing an elementary row operations and column operations we can always find a k cross k identity matrix right here and rest whatever remains is the parity matrix.

So, clearly it is pretty easy to see that by simple row and column operations I can actually show that we have a rank k matrix and here we have an identity matrix of size k cross k. So this is a simple way to look at why each and every generator matrix can essentially be reduced to an equivalent systematic form; so we have used these tools to come up with a systematic form.

(Refer Slide Time: 05:59)

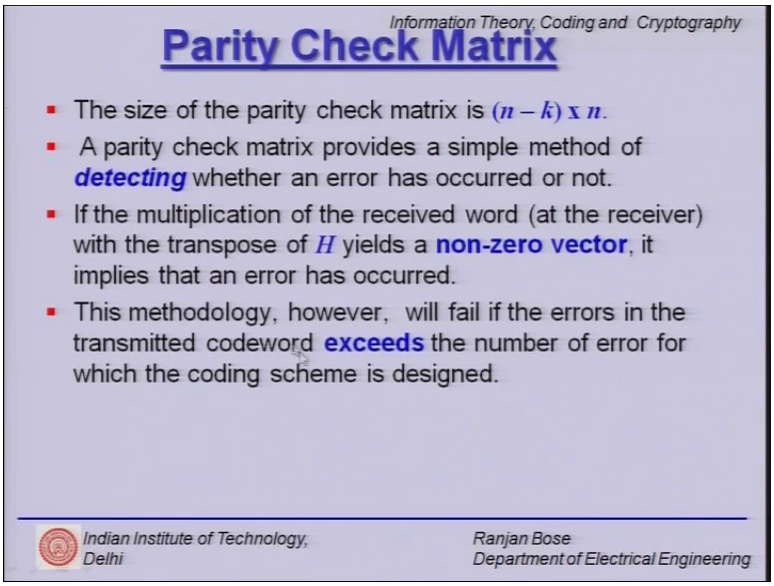


Now question we ask ourselves is generation is easy we have done away with a lookup table we can generate codewords on the fly, but is it also possible to decode efficiently; that is to say in order to check whether a received vector is a indeed a valid codeword that indeed it has not undergone any errors can we have are equally efficient methods. And that could also be matrix based, the answer is a vehement yes we have a matrix called a parity check matrix H for a given code. So what is this parity check matrix two well it does what it is name suggests given a vector c and if that vector c is a valid codeword then c H transpose should be a 0 vector for a valid codeword.

So, the parity check matrix can immediately tell you whether a codeword has a undergone error or not. So take a valid codeword multiply with H transpose and you should end up with a 0 ok. Now we know how a codeword is generated it is done so using a generator matrix c is equal to i information vector into G and therefore, in general if you replace the c by i into G you get i G H transpose equal to zero, but this must be true for all valid codewords all information word.

So, regardless of i this must hold true which forces us to put G H transpose equal to 0. So we have a simple mechanism to find out H given G or G given H; so if I ask you to represent a linear block code you can give me the generator matrix or equivalently you can give me the H matrix. We must observe that for a given G H need not be unique that is we can have several options of H which will satisfy G H transpose equal to 0.

(Refer Slide Time: 08:19)

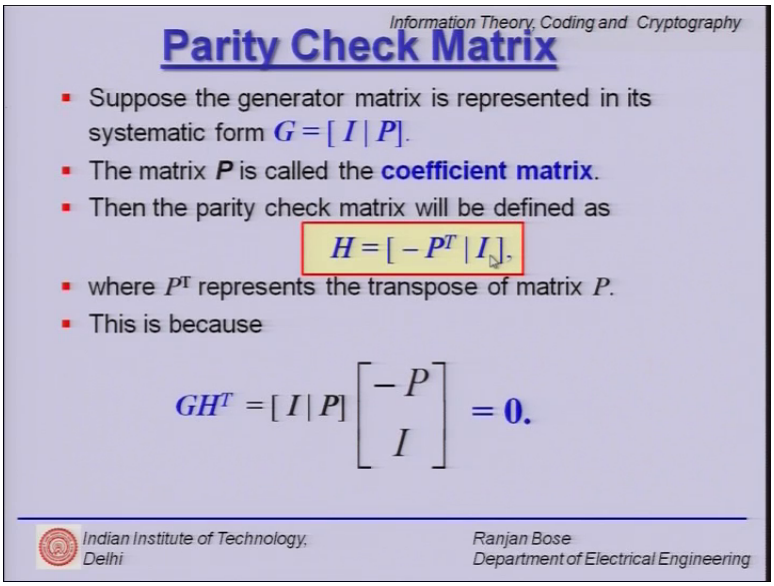


So, the size of this uniquely important parity check matrix is n cross k cross n and it is a simple method of detecting; please note it will only tell you whether an error has happened or not it is not doing any job of correction so far.

So, all you do is multiply the received vector with H transpose and wait for a nonzero vector if you get a nonzero vector you are happy because you can proclaim that no error has happened. On the other hand if it gives a nonzero vector; that means, something has gone wrong, but please note this methodology succeeds only if the number of errors for which you have designed the parity check matrix is satisfied. However, if the number of errors in the codeword exceeds the number of error for which the coding scheme is designed then this will fail.

So, you should be careful about this you can end up having a 0 vector by performing c H transpose and still have a lot of errors in the codeword.

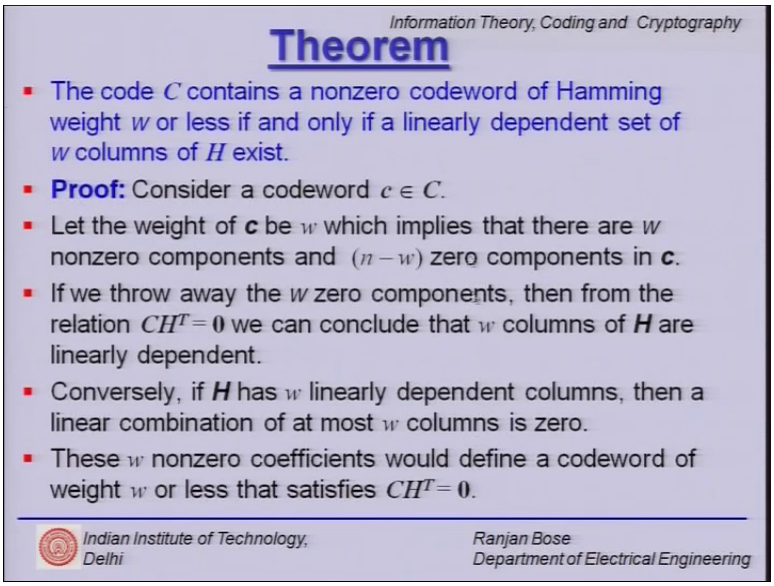
(Refer Slide Time: 09:40)



Now, how do we get this so we clearly know that any generator matrix G can be effectively represented as I and then partition and this parity matrix P; this P should not be confused with parity check matrix this P is the parity matrix. It is sometimes also called coefficient matrix to avoid the confusion.

So, the parity check matrix can now easily be defined as minus P transpose partitioned I and you can easily check that G H transpose if written in this form is necessarily equal to 0. So you can do a simple check, here there is a transpose so you have I P here minus P transpose I is equal to 0 and this is an identity, so you can easily first represent G in terms of it is systematic form and then immediately obtain H from there.

(Refer Slide Time: 10:49)



Now we move forward and look at some properties of this parity check matrix H; so the code C contains a nonzero codeword of a hamming weight w or less if and only if a linearly dependent set of W columns of H exist. So this is the statement and what we do is consider a any codeword c belonging to the set capital C and the let the weight of this codeword c be w which implies that there are w nonzero components and clearly the length of this vector is n, because it is a linear block code of block length n consequently n minus w 0 components would exist in c you partition into the 0 component and nonzero components.

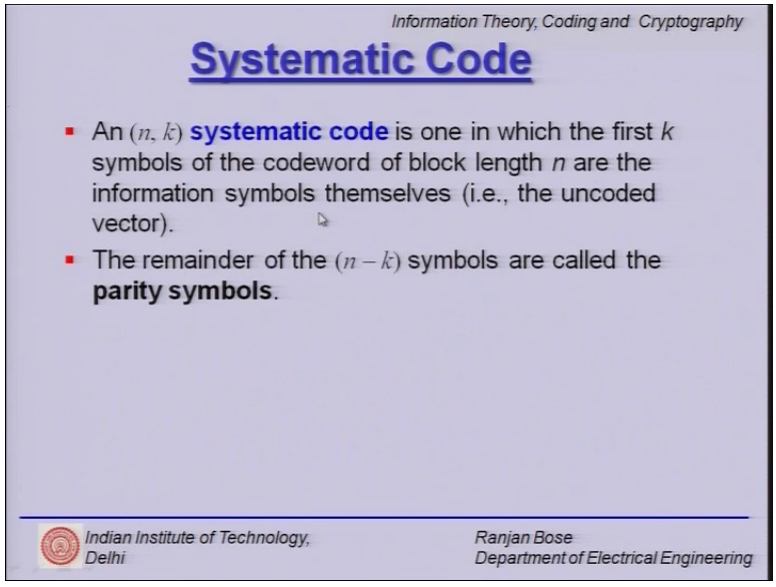
Now, we must have c H transpose equal to 0 ok, this capital C represents that all the vectors in this set when multiply with H transpose should give you 0 including all the valid c 1, c 2, c 3 all the valid codewords. Now we are considering any one particular codeword with a weight w that is w nonzero elements exist; so what does this do this product C H transpose 0 picks up the columns of H and the linear combination of this column is equal to 0 therefore, it; obviously, means that an a nonzero codeword of hamming weight w or less if and only if a linearly dependent set of w columns of H exist ok.

So, this process of multiplication is just picking out the w and showing them the linearly dependent w columns of H exist. Conversely if H has w linearly independent columns then a linear combination of at most w columns is 0. What is the practical use of this

observation? Well w has a very interesting connotation, the minimum w is called the minimum weight of the code; which is equal to the minimum distance of the code w star is equal to d star. Consequently by carefully observing the number of linearly dependent columns of H we can make a statement about the w star ok.

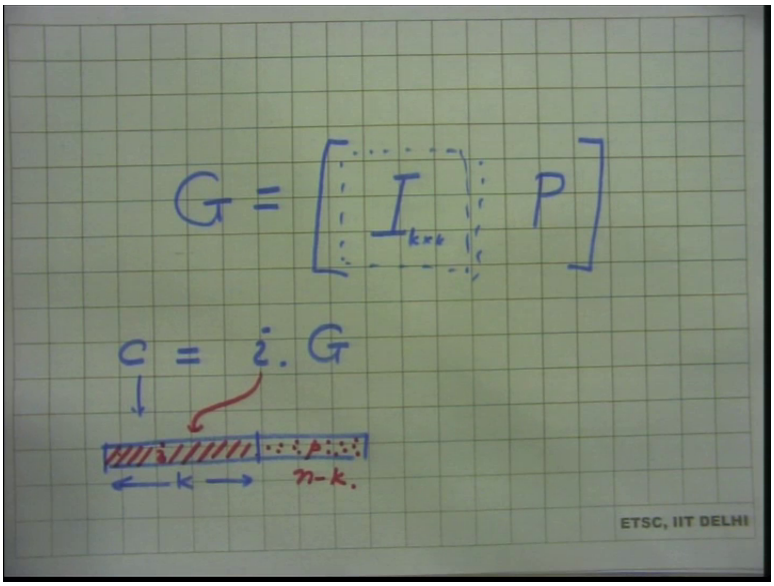
So, w nonzero coefficients would define a codeword of weight w or less that satisfies this condition, so we can use this technique to figure out what is the minimum weight of a code.

(Refer Slide Time: 13:59)



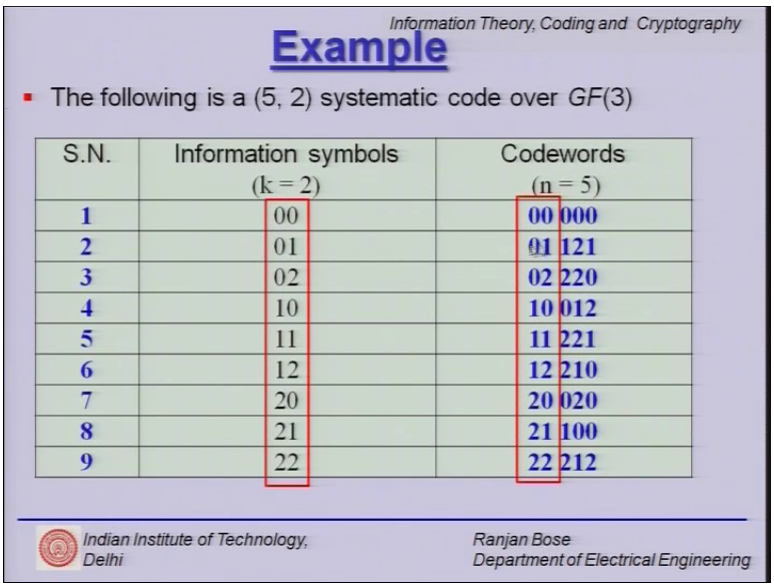
Now we go back to our systematic code and we have already seen that an n comma k systematic code is one in which the first k symbols of the codeword of block length n other information symbols themselves. Please remember if we go back to the matrix representation we have said that the generator matrix for a systematic code can be represented as an I which is the k cross k and a parity matrix P.

(Refer Slide Time: 14:28)



So, whenever we are in the process of obtaining a codeword c as a product of i into G, thanks to this identity matrix the first k the first k elements of this codeword will definitely be i and this n minus k will be the parity P; this is simply because of the construction of this G matrix in the systematic form. So we come back to our slides and we say that in a systematic code the first k symbols of the codeword of block length n are the information symbols themselves that is the uncoded vector and the remainder n minus k are the parity symbols.

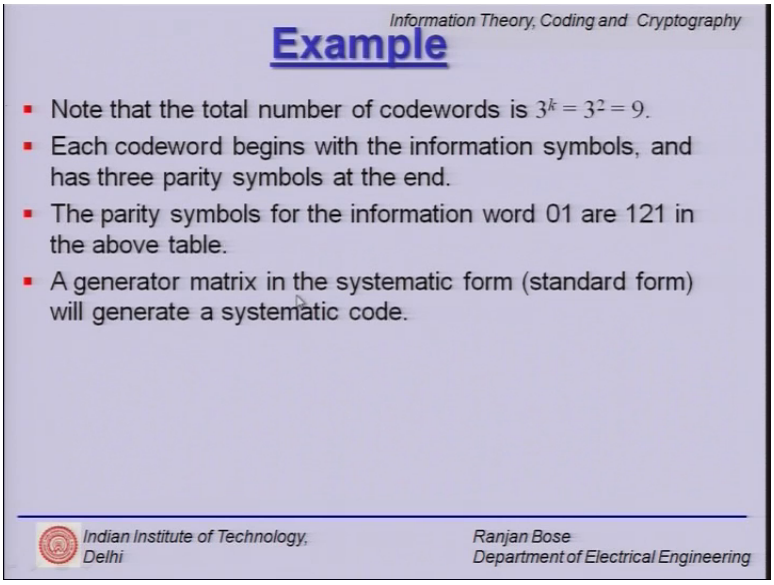
(Refer Slide Time: 15:48)



Lets quickly look at an example; so here is a code over GF 3 it is an n comma k where n is 5 and k is equal to 2 so it takes 2 symbols at a time it is no longer a binary code the elements can be 0 1 and 2, so it takes 2 symbols at a time and gives out a 5 symbol long codeword. Here if you see there are possible 9 codewords in this table and 0 0 has 0 0 0 0 0, 0 1 has 0 1 1 2 1 and so and so forth so this is a 5 comma 2 systematic code over GF 3.

If you do a careful observation you will notice that the first 2 why 2 k is equal to 2. So the first k elements of the codeword are the same as the uncoded symbols and remainder are the parity n minus k where n minus k 5 minus 2 is 3 in this case.

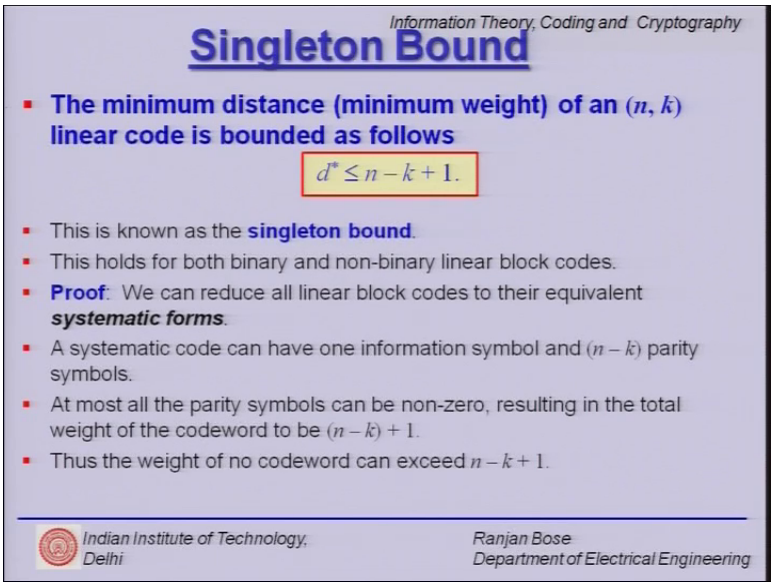
(Refer Slide Time: 17:04)



So why did we have the total number of codewords as 9 well you have q s bar k q is equal to 3 because the GF q is GF 3 here and k is 2, so 3 squared is 9 and each codeword as we observed begins with the information symbol and has 3 parity symbols at the end a simple example was the information word for 0 1 has parity 1 2 1 making it the codeword 0 1 1 2 1.

And a clearly as generator matrix in the systematic form which is also called the standard form will generate a systematic code.

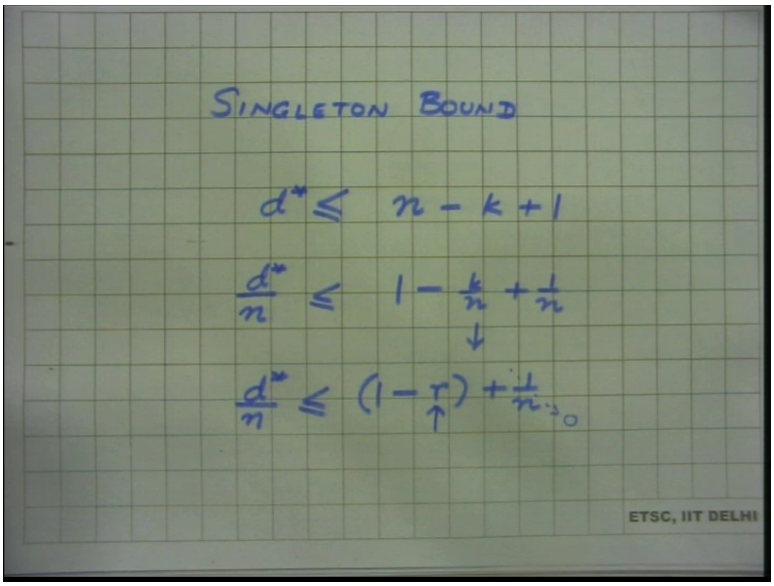
(Refer Slide Time: 17:46)



Now let us look at a very interesting and useful bound called the singleton bound. What does it say? The minimum distance and for us we always read the minimum distance as the minimum weight because we are in the domain of the linear block code so the minimum distance of the minimum weight of an n comma k linear code is bounded as d star equal to w star upper bounded by so less than or equal to n minus k plus 1.

This is fairly interesting because so far n and k were supposed to be independent I can have any k, I can have any n only the rate meant k over n is a rate of the code and you could always have any arbitrary choice of k and n provided n was greater than k, but please note for linear block codes we have suddenly a constraint on the minimum distance; so the maximum possible d star can never exceed n minus k plus 1 this is called a Singleton Bound.

(Refer Slide Time: 19:18)



So, if you just look at it we have here the singleton bound and we have said d star is less than or equal to n minus k plus 1. So please note if you divide both sides by n you have this normalized distance 1 minus k over n plus 1 over m, but if you see this is nothing but the code rate of the code. So you have the normalized minimum distance 1 minus r plus 1 over n, so as you let n tend to a large number that is you are looking at a larger and larger a block length this quantity kind of tends to 0 and you have your upper bounded by 1 minus the code rate r.

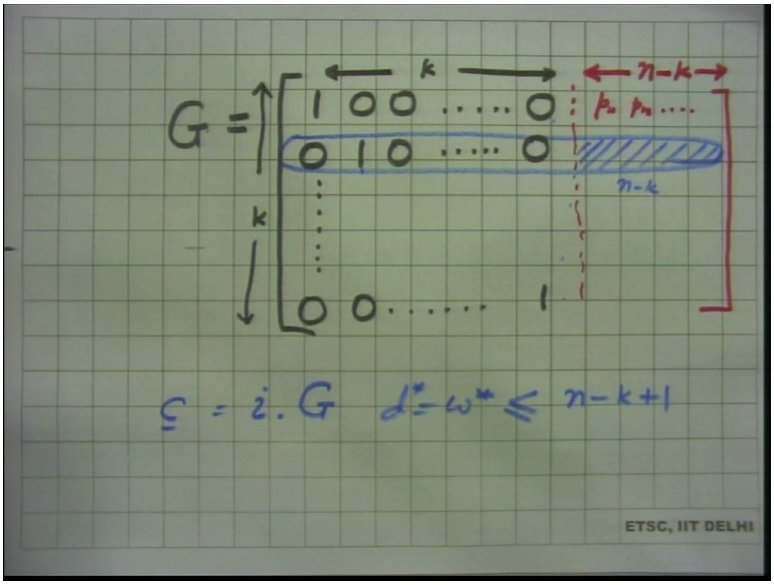
But please note r has a strong physical significance it is actually the efficiency of the code, how much overhead bits or symbols are you sending and r necessarily has to be as close as 1 as possible for practical codes. So this gives puts in perspective that how a normalized d star is related to the code rate which is the efficiency of the code and it also tells us how they both of them are contradictory requirements on one side I need r to tend to 1 because I do not want to put too many overhead symbols in my codeword.

So, r should touch 1 if I want to push that then this quantity becomes closer and closer to 0. At the same time I want this normalized d star not going to 0; so on one hand I want to push my r, on other hand I would like to increase my d star because d star has a direct implication through the error detection and error correction capability as we will shortly see. So we go back to our slide on singleton bound and we make another observation that this holds for both binary and non binary linear block codes.

In fact, to prove this you only need the condition of linearity; how do we go about proving it well we know that we can reduce all linear block codes to the equivalent systematic form what does that mean; that G itself will be I partition P. Now any systematic code will have one information symbol and n minus k parity symbols; please recall that each of the rows of the G matrix is a valid codeword and it is forms the basis.

But suddenly how does my valid codeword look? Well if you go back and observe your systematic form you will notice G represented as follows and then you have your matrix.

(Refer Slide Time: 23:08)



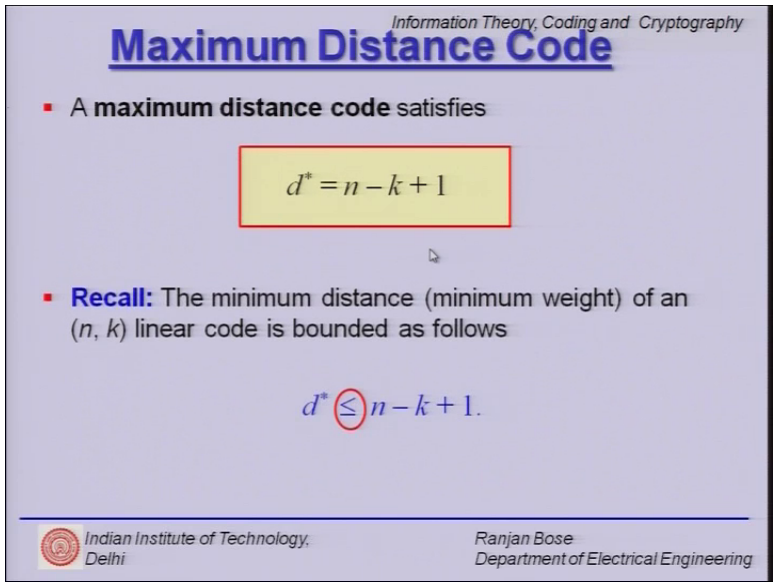
So P 1 1, P 1 2 and so and so forth, but please note that this is n minus k and this is k this is also k. Now we have established that each and every row is a valid codeword and since it is a linear block code sum of any two codewords is a valid codewords.

In fact, the moment we multiply i with G i tells us which of those codewords to pick and add them up fine. Now if you just pick any one of the valid codewords you will have here at most n minus k, so you can have this as all of them nonzero in this case the weight of this vector is n minus k plus 1, because in this row all of them are 0's except any one. This is true for all of these rows and the linear combination you can always try that.

So, no matter what you do the minimum weight w star will always be less than or equal to n minus k provided all of them are nonzero plus 1. So n minus k plus 1 is the upper

bound and which is the singleton bound. So we go back to our slides and we observe that at most all the parity symbols can be nonzero and therefore, the weight of any codeword can be n minus k plus 1 thus no codeword can exceed n minus k plus 1.

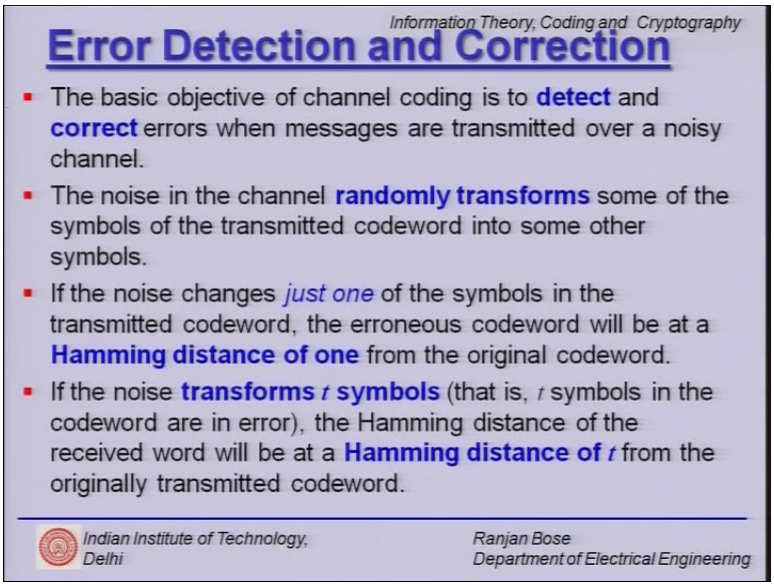
(Refer Slide Time: 25:54)



So, now this is important because consequently we will have to relate the d star to some error correcting capability, so let us talk about a maximum distance code in terms of the minimum distance; because minimum distance is a good property to have we would like the codewords to have as large minimum distances as possible. So we look at the condition where d star is actually equal to n minus k plus 1 and all those codes that satisfy this condition they are called maximum distance codes right.

So, this is coming directly from this inequality and the moment we can reach the equality part of it the code becomes a maximum distance code.

(Refer Slide Time: 26:53)

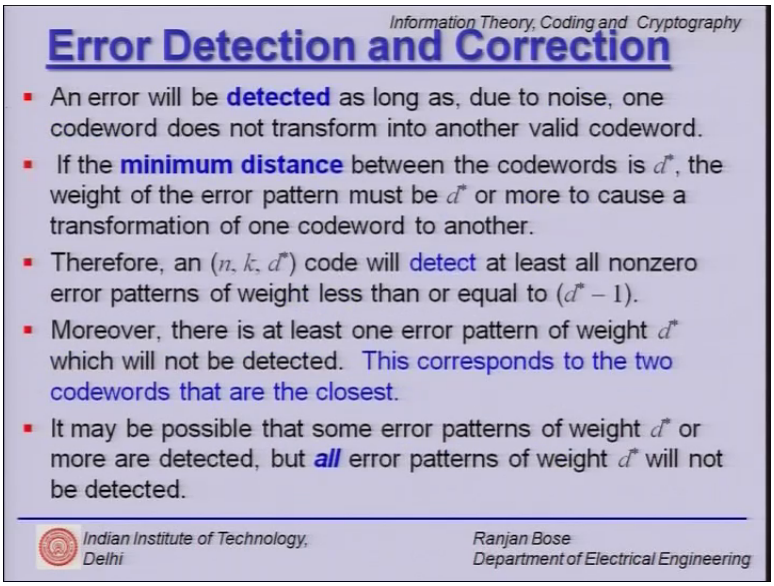


Now, let us take one step further and talk about error detection and error correction because that is what we set out to do. The basic objectives of the channel coding is to detect and correct errors when messages are transmitted over a noisy unreliable channel, so what does the noise do? Well it does it is job it transforms some of the symbols or bits into something else.

So, for binary it can flip the bits, for non binary it can transform any of the symbols to be interpreted as any other symbol at the decoder. Now just observe that if only one of the symbols in the transmitted codeword is erroneous then the received vector will be at a distance one from the original codeword. What distance are we talking about? The hamming distance; if there are t symbols which have been changed altered flipped then the hamming distance of the received vector is t.

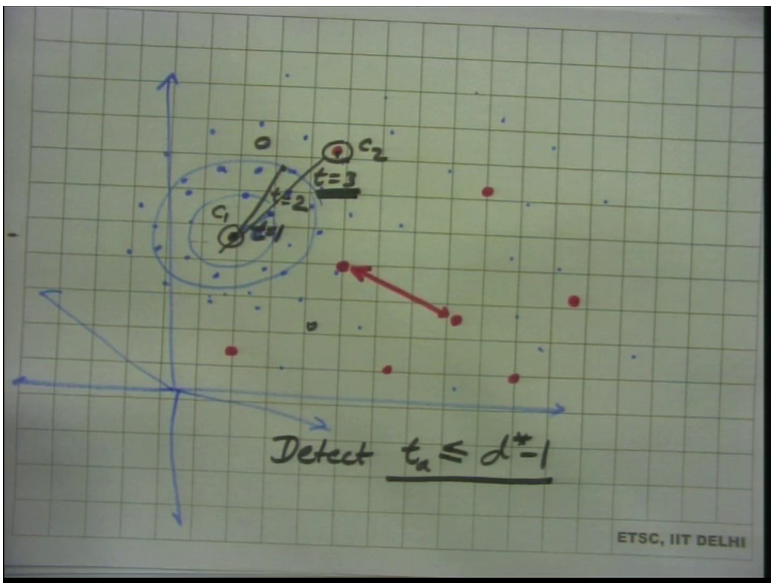
So, from now onwards we will start talking about a t error correcting codes, that is we wish that r code of block length n can correct t symbols in error you will put that constraint and try to figure out what are the conditions to do so.

(Refer Slide Time: 28:22)



So, first we look at the problem of error detection, because if we can detect then only we can think about correcting it. So an error will be detected as long as one of the codeword does not completely transform into any other valid codeword, please note we are in a space where n tuple represents a point in the space.

(Refer Slide Time: 29:02)



So, let us look at another diagram here so we have this nice n dimensional space, where any point in that space is a vector of length n and needless to say you can have this n dimensional space represented. Now you can have another point, which is another n

tuple and the hamming distance between them is the number of places one of the codewords differs from the other codeword.

So, a code is a collection of codewords and suppose in my space even though there are many points only some of the points have been picked to be valid codewords. Now what happens when you transmit any one codeword say c 1 and it gets one of the symbols in error. Well the received vector is at a hamming distance 1 from this codeword so there are some possible vectors which are at a distance d equal to 1.

But since we are saying that we would like to represent an error by t we can say that t equal to 1 in this case. Suppose 2 bits or symbols get flipped then there are another set of codewords which is at now at a distance t is equal to 2 all of these codewords are at t equal to 2. These codewords are just not valid codewords their vectors which are two places different with respect to original c 1 and suppose my next tier of codewords distance t is equal to 3 actually involves c 2.

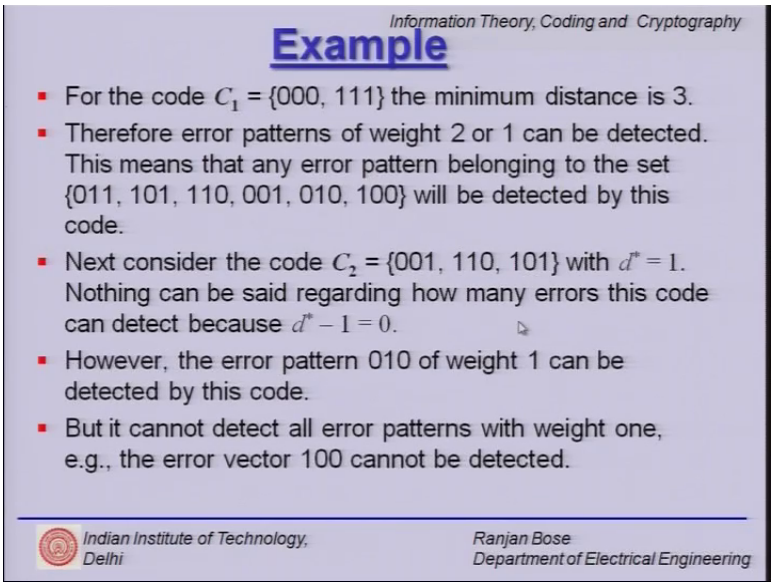
Suppose c 2 is at a distance t is equal to 3 away then suppose c 1 is in error such that 3 of the elements are interchanged and it resembles now c 2 what I will received is a vector c 2 which is a valid codeword. At the decoding side I will have no way to figuring out whether c 1 was sent and it had 3 errors or c 2 was sent and it had no errors.

So, the detection problem can only be solved if the number of errors are fewer than the minimum distance of the code; what is the minimum distance? Distance between the smallest distance between any two codewords. Now please note that a different combination of 3 errors could have happened and what I received is this vector or this vector these are not valid codewords, so still I can detect them to be an error.

But if my luck is bad and my c 1 gets transformed to c 2 with 3 then there is no way I can detect. So the worst case I would like to say that the number of errors to detect, so I can detect maximum d star minus 1 I cannot detect d star errors because one codeword will become another codeword and there will be no way to finding out. So we go back to our slides and we look at that there is at least 1 error pattern of weight d star, which will not be detected, which is the 1 this corresponds to the two codewords that are the closest and 1 will get transformed to the other codeword.

It may be; however, possible that some error patterns of weight d star or more are indeed detected, because you do not transform one valid codeword to another valid codeword; however, the ambiguity remains, but all error patterns of weight d star will not be detected some will get transformed into another one. So let us look at a quick example.

(Refer Slide Time: 34:02)



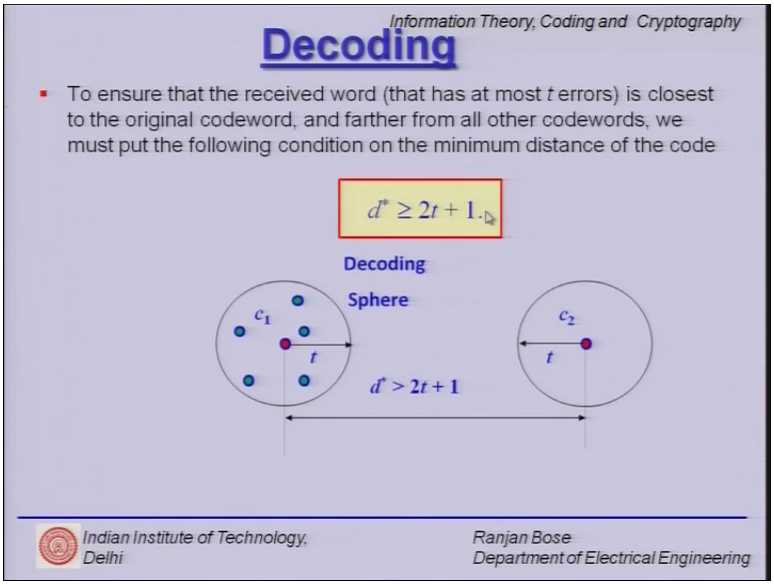
Let us look at this simple toy example of a repetition code n is equal to 3 k is equal to 1 and the minimum distance is 3 there are only two codewords 0 0 0 and 1 1 1 minimum distance is 3.

So, what does it mean any error pattern of weight 2 or 1 can be detected, so how many are there well 0 1 1, 1 0 1, 1 1 0 so and so forth patterns of weight 2 patterns of weight 1 all of these can be detected. That is to say if I transmit 0 0 0 and either 1 or 2 errors happen I can always say look I did not receive 0 0 0 or 1 1 1, so my flag goes up and some error has happened.

But if 3 errors occur d star is equal to 3 then 0 0 0 becomes 1 1 1 I really do not know whether to be happy or sad because 1 1 1 is a valid codeword. So if you have another example where c 2 is these 3 code with d star is equal to 1 well if d star is equal to 1 we do not know whether it can detect errors or not the error pattern of 0 1 0 of weight 1 can be detected by this code because anyone when transformed will be detected because it is not one of the valid codewords.

But suppose 1 0 0 happens so 0 0 1 will become 1 0 1 and there is no way i can detect what was sent.

(Refer Slide Time: 35:54)



So, now we would like to look at the correction part, the decoding part; so you would like to ensure that the received word which has t errors or at most t errors it is closest to the original word codeword and at the same time farther from any other codewords ok. So we want to have a unambiguous decoding so what I want to do is the minimum distance, which is the relationship between the 2 codewords which are closest to each other that separation should be greater than 2 t plus 1 well it is easy to visualize.

Let us have our coordinate system in place and I have just shown 3, but it was an n dimensional space and I can always put a point in the space which represents codeword c 1 and there will be a decoding sphere around it which is of the size radius t. It means within this the circle there are many points which are at a hamming distance t or less so just remove that axis and just focus on c 1 and c 2 all right.

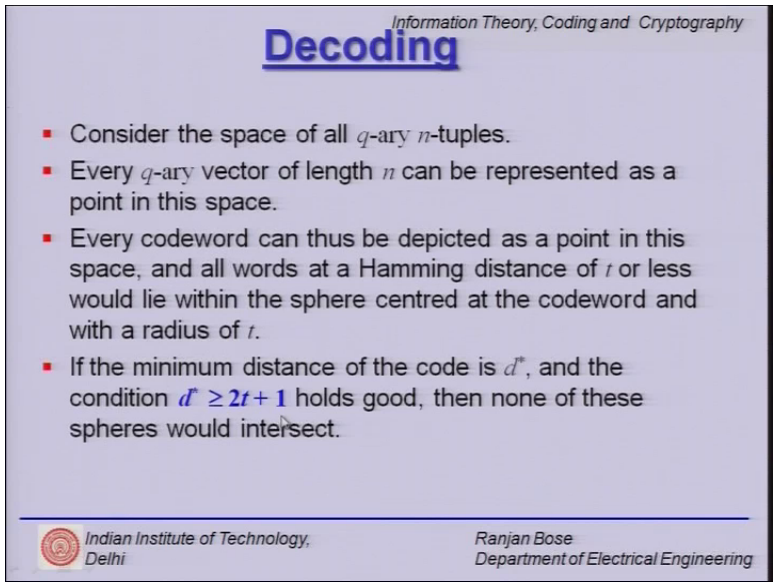
So, we are trying to figure out what is the best, what is the constraint we can put to ensure that I can correct t errors. So we draw a circle around c 1 and there are many many points inside this circle, which are at a hamming distance t or less whoever is sitting on the circles circumference is exactly at a hamming distance t just r closer to or farther away, but definitely less than t, but the worst case is that this guy is sitting on t.

Now, we can have non overlapping spheres such that c 1 and c 2 do not overlap in terms of their decoding spheres, but if t touches the circumference of these two circles touch each other and a codeword lies on this then we do not know because it is equidistant from c 1 or c 2 and we cannot do the nearest neighbor decoding, otherwise if I have any valid codeword sitting inside this sphere then if this vector is received most likely c 1 was transmitted not c 2.

So, I can any codeword any vector sitting inside the sphere will be mapped back to c 1 and therefore, this is the decoding sphere for c 1; similarly this is the decoding sphere of c 2. Any received vector inside c 2 will be mapped back to c 2, but in to ensure that in the pathetic case when this circumference is touch then any codeword sitting on or any vector sitting on the circumference will be it will be ambiguous whether to put on to c 1 or c 2 we need the separation of one it can only be an integer so it is 2 t plus 1.

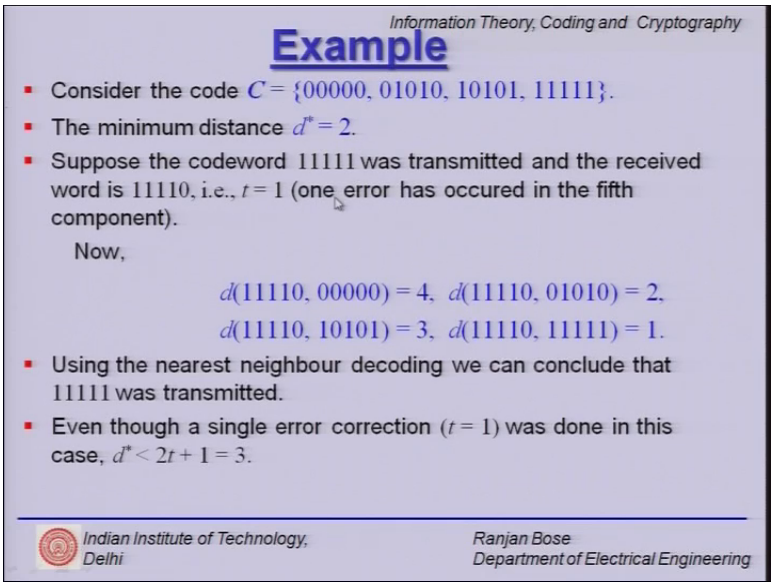
So we have to make sure that the codewords are separated to greater than 2 t plus 1 or 2 t greater than or equal to 2 t plus 1, if I want to ensure that t errors are indeed corrected.

(Refer Slide Time: 39:51)



So, what does it mean well every q-ary vector of length n can be represented as a point in this space and every codeword is depicted as a point as we saw last time and the sphere contains all the valid vectors within it of the distance t or less and we establish the d star should necessarily be greater than or equal to 2 t plus 1, then the spheres will never intersect and nearest neighbor decoding can indeed happen.

(Refer Slide Time: 40:23)

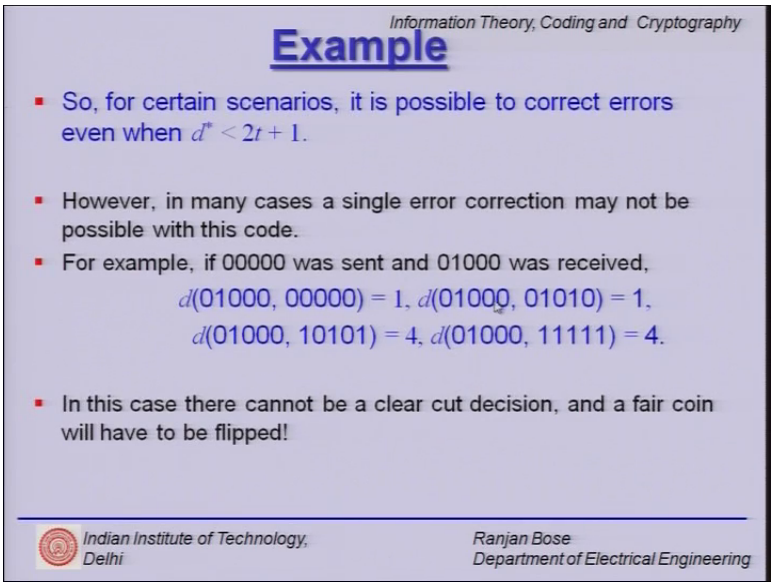


So, let us look at another simple example well k is equal to 2, n is equal to 5 and we can find out very quickly that the minimum weight and minimum distance of this code is indeed 2. Suppose this 1 1 1 1 1 vector was transmitted, but 1 bit got flipped so what we received was 1 1 1 1 0 so the fifth component is an error. So you can find out the distance, the hamming distance of this received vector with each one of the 4 codewords and I find that the distance is least with respect to 1 1 1 1.

So, in the code space the received vector is closest to 1 1 1 1 1 than any other codeword and hence the nearest neighbor decoding tells me that not only did you make an error you have first chance of correcting it and the real codeword that was transmitted was 1 1 1 1 1 and you have been able to recover from the error; so we applied this nearest neighbor decoding. Now please note that d star is 2, so in principle we should not be we should not be able to correct a single error because d star should be greater than or equal to 2 t plus 1 here t is 1.

So, d star should be at least 3 to correct one error, but we have been able to correct one error, but this is not true in general so some special cases you can correct the point of this example is to show that d star greater than 2 t plus 1 ensures all sad pathetic cases are also covered, you will never make a mistake you will always be able to detect and correct t errors.

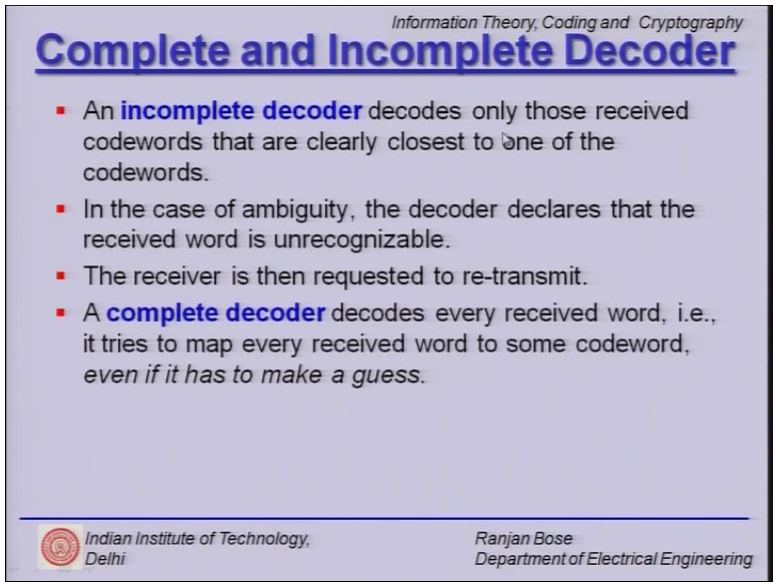
(Refer Slide Time: 42:30)



So for certain scenarios it is possible to correct errors even if d star is less than 2 t plus 1.

But in many cases a single error correction may not be possible for example, if 0 0 0 0 0 was sent and we make 1 error t is equal to 1 again second component is an error, I now again find the distance with all of them and suddenly I am in a dilemma because two of the valid codewords are equidistant from my received vector; what do I do? I need to guess and I can always toss a fair coin because anyone could be valid and I have no way of being sure which one to suggest.

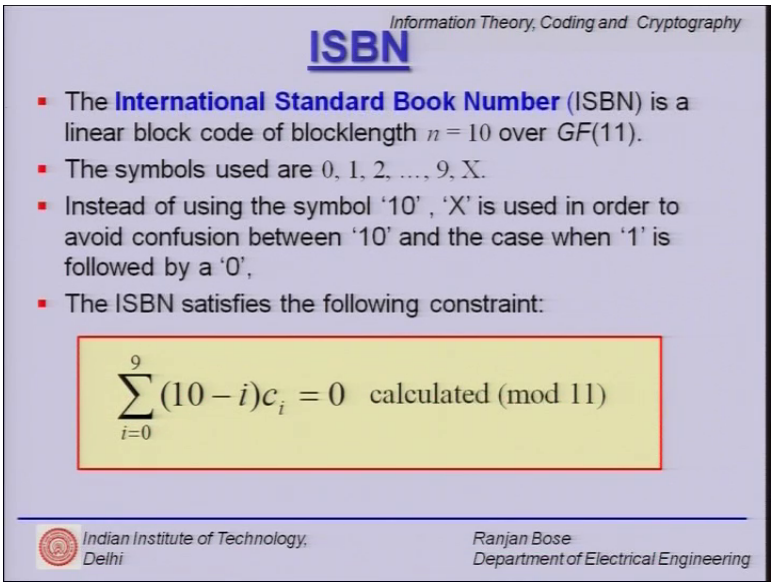
(Refer Slide Time: 42:31)



This brings us to this definition of a complete and incomplete decoder that example motivates this definition and incomplete decoder decodes only those received codewords that are clearly closes to 1 of the codewords ok.

In case of ambiguity just like we saw in the previous example, the decoder declares that the received word is unrecognizable and it raises its hands. Then with the receiver then it says look error is detected, cannot correct please re transmit ok. It can also do this automatic repeat request the ARQ. On the other hand we can be adamant we can say a complete decoder is one which definitely decodes every received word whether there is an ambiguity or not, even if it has to make a guess it will do so, but it will give you an answer it will never ask for a retransmission.

(Refer Slide Time: 44:24)

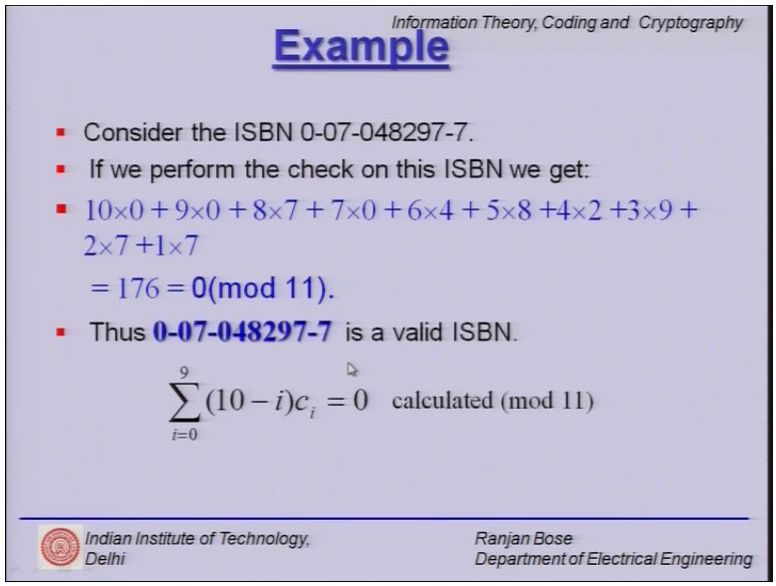


So, let us look at one final example a practical example it is this is the ISBN or the international standard book number which is present behind every book that you see it is a linear block code over GF 11, please note this is a prime number so GF 11 exists and the block length is 10, now ISBN 13 is also available, but this example is for ISBN 10 where n is the block length. How do we represent this GF 11? Because GF 11 must have eleven symbols including 0 and 1.

So, we have 0 1 2 3 4 9 and I should have the 10, but tenth can be confused as 1 followed by a 0 and 1 and 0's are already valid so we put in X ok. So X represents 10 so your codeword whatever is written behind every book will have either 10 or 13 symbol long codewords and the ISBN satisfy the following condition; 10 minus i into C i where i is the component i-th component i is equal to 0 to 9 because their total 10 elements in the codeword should be 0 provided it is called calculated mod 11 of course, the arithmetic has to be mod 11 because it is a code over GF 11.

So, if you have this condition satisfied then you can declare whether an ISBN is a valid ISBN or not ok.

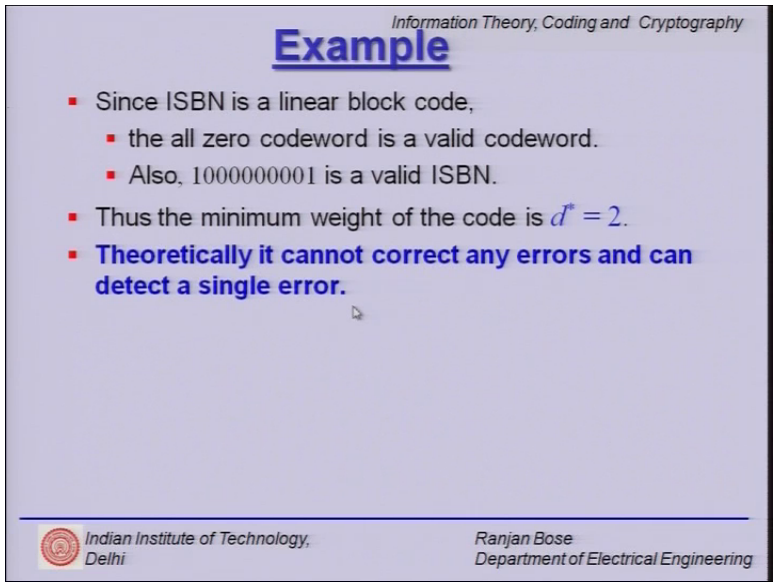
(Refer Slide Time: 46:19)



So, let us look at a quick example, now let us consider this 1 2 3 4 5 6 7 8 9 10, 10 symbol long ISBN ok. We can do a quick check on this ISBN and we can apply the formula that we saw in the previous slide the summation and you multiply the first one with so the first number second number with 10 minus i, so 10 into the first number 0, 9 into 0, 8 into 7 and so and so forth then go right up to the last one 1 into 7 and you take modulo 11 and you end up having a 0.

So, we can carefully observe this and say that yes indeed it is a valid ISBN ok. You can do this check so what is the use of this, I mean where are we doing error correction here this is supposed to be error detecting error correcting code so let us look at what we can actually get out of it.

(Refer Slide Time: 47:33)

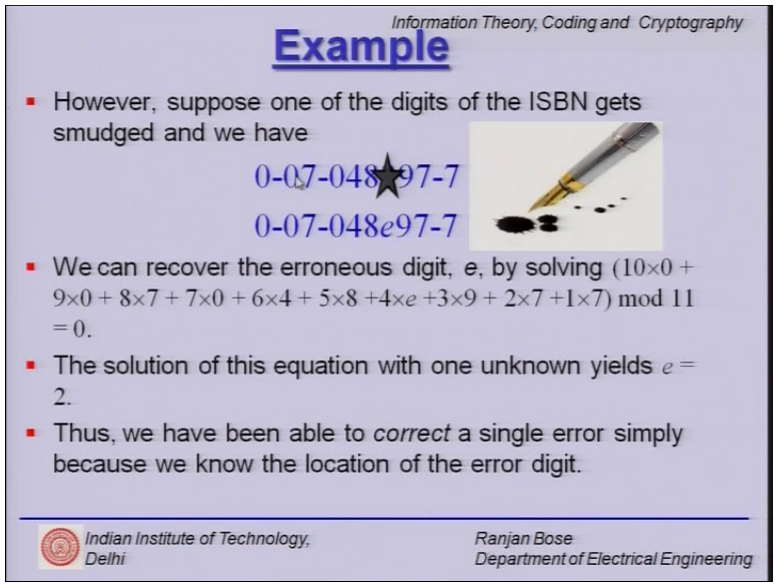


But before that we must look at the property of an ISBN, so an ISBN is a linear block code which means that the all 0 codeword is a valid code codeword and a sum of any two codewords is also a valid codeword.

So, the moment you throw a linear block code at me I first find the d star, I have to find out the minimum distance because then I will know the error detecting and error correcting capability. So one of the ways is either to list out all of the codewords well you will need a lot of good luck and compute power to do so or we can plays smart trick we focus hard enough and come up with another seemingly valid ISBN.

So, you can check that this number also satisfies the constraint for an ISBN code, so it is a valid ISBN code. So I have identified the all 0 and this one and of course, the minimum distance will be 2. So the ISBN code has d star is equal to 2 so it can detect, but in principle it should not be able to correct a single error because error correction requires d star to be 3 for a single error correction ok. So are we looking at a useless code? Theoretically it cannot correct any errors and can only detect a single error.

(Refer Slide Time: 49:12)

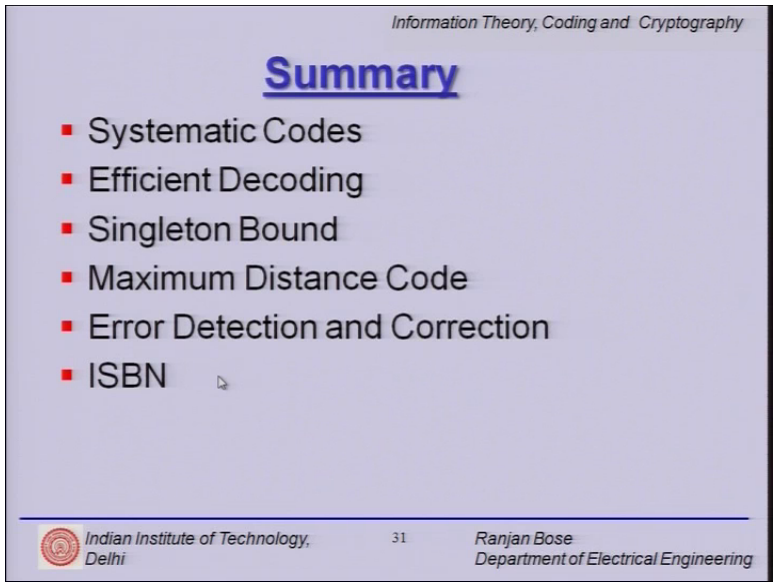


So, so far so good, but can we extract more out of it? Let us look at an example so we put another valid ISBN you can satisfy yourself that this is a valid ISBN, but you know books we work with pens there are moths around and you can have an ink pen which is leaky and you splash some ink, so one of the digits is blotted or smudged so we have no clue what it was the moth could have eaten it or the ink has dropped on it and I need to know whether the this is a valid ISBN or not.

So, this is a problem not of error detection clearly the error is there I can see that error, but can I correct, can I recover from, can I guess what is this, so I modeled this ink blot as my error e ok. So what we do is we again apply that condition and that was an equation one equation, one unknown is easy to solve and little bit of math and it tells me that e is 2. So I have been able to give you the value of e not only did I detect the error, I corrected the error; what is this? This is an error and I have corrected it e is 2 and how did I do it because theoretically I should not be able to correct an error well the answer is we have been able to correct a single error simply because we know the location of the error.

In linear block code you do not even know the location so you can do much better if you can know the location of the error that is an additional information which can help you recover from errors.

(Refer Slide Time: 51:11)



So, with that we come to the end of this lecture; let us summarize what we have looked at in this past 1 hour we have looked at systematic codes, we have understood how they are written, what is the purpose, what is the advantage of putting codes in a systematic form, we looked at efficient decoding, then we observed what is a singleton bound and how it can lead to the definition of a maximum distance code, we then looked at the notion of error detection and correction and finally, we looked at a very practical example the ISBN; with that we come to the end of this lecture.