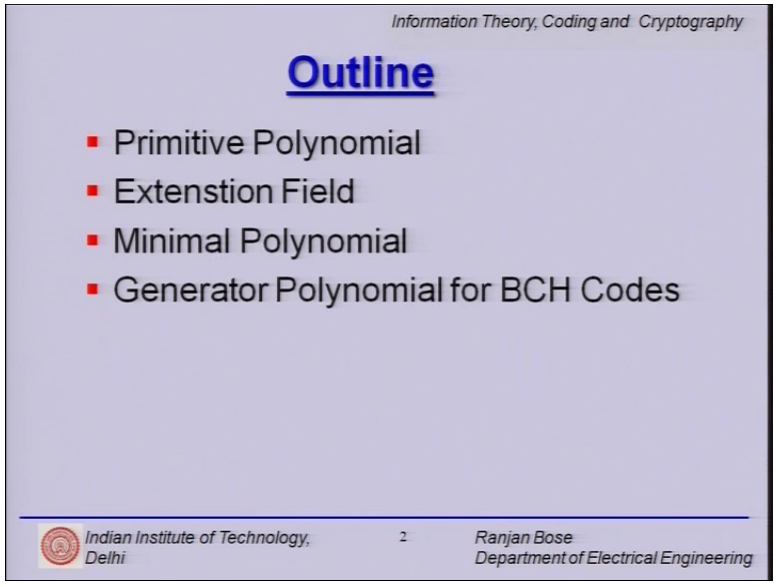
Information Theory, Coding and Cryptography
Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Module - 23
Bose-Chaudhuri Hocquenghem (BCH) Codes
Lecture - 23

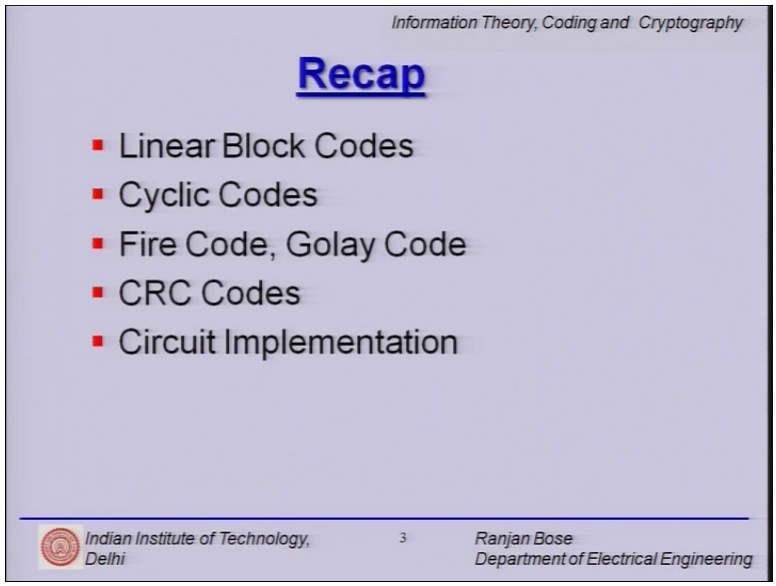
Hello and welcome to our next module on BCH codes; let us start with a quick outline.

(Refer Slide Time: 00:36)



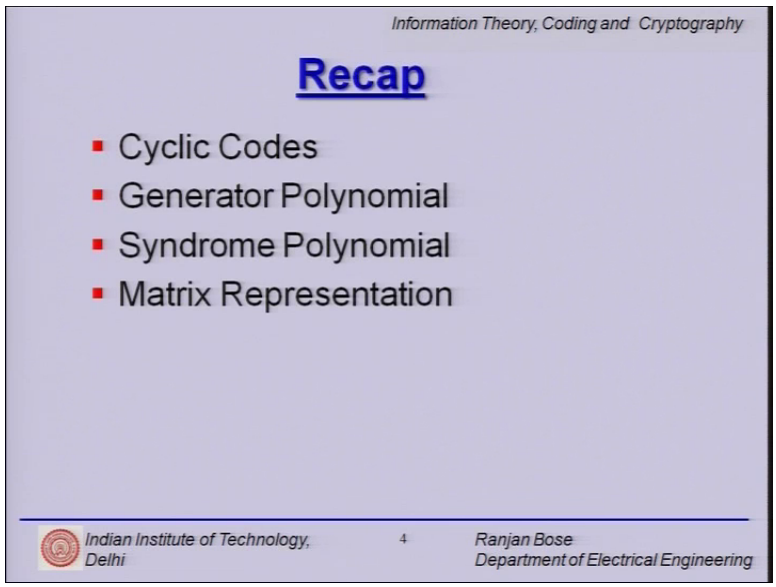
We will look at some mathematical preliminaries, will start with the primitive polynomial, we will talk about the construction of extension field, then we will talk about minimal polynomials. And finally, we will use all of this mathematical tools that we have used as discussed to form the generator polynomial for BCH codes.

(Refer Slide Time: 01:01)

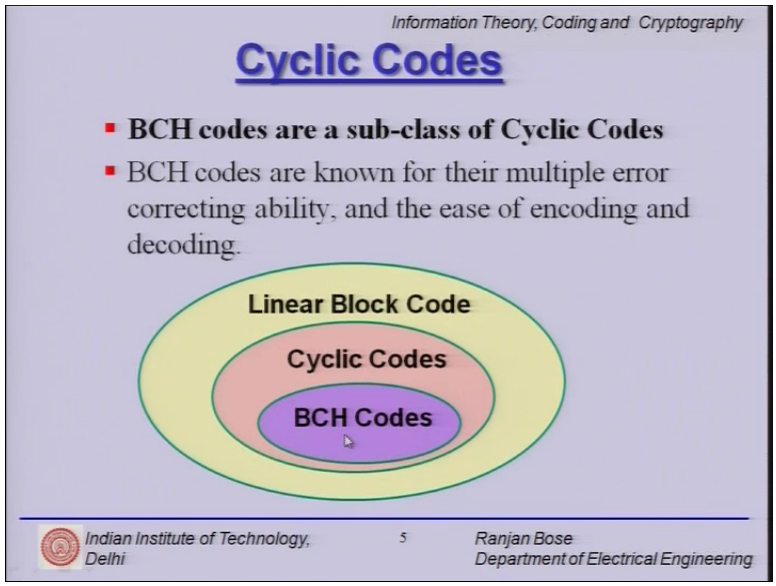


So, let us do a quick recap as to what we have done already we have looked at linear block codes which were put in certain algebraic constraints and enabled us to detect and then correct errors. A subclass of linear block codes, cyclic codes we had additional algebraic constraints which made them much stronger codes in terms of error detection and even correction. We looked at certain types of cyclic codes and we looked at the circuit implementation what we want to do today is to look at yet another subclass of cyclic codes called the BCH codes.

(Refer Slide Time: 01:43)



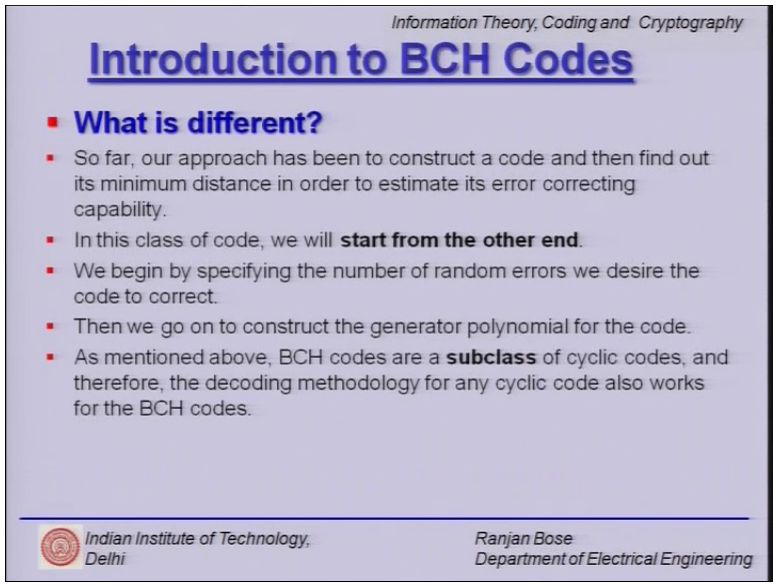
(Refer Slide Time: 01:47)



So, if you remember the cyclic codes right they formed a subclass of this linear block codes. Now we are moving within the domains of cyclic codes, so BCH codes are a subclass of cyclic codes and are known for the multiple error correcting ability. We will see that very good BCH codes which stands for Bose-Chaudhuri-Hocquenghem codes are over non binary fields. So we have GF q over which BCH codes are defined and they can correct several errors, burst errors and are pretty strong in the error correcting capability.

So if we go to a sub classification we find that BCH codes form a subclass of cyclic codes. So linear block codes we developed a rich theory, then we went to cyclic codes again more theory to use cyclic codes in terms of polynomial representation, and then we will have BCH codes. We can use the techniques already developed for LBC, cyclic codes all are applicable for BCH codes.

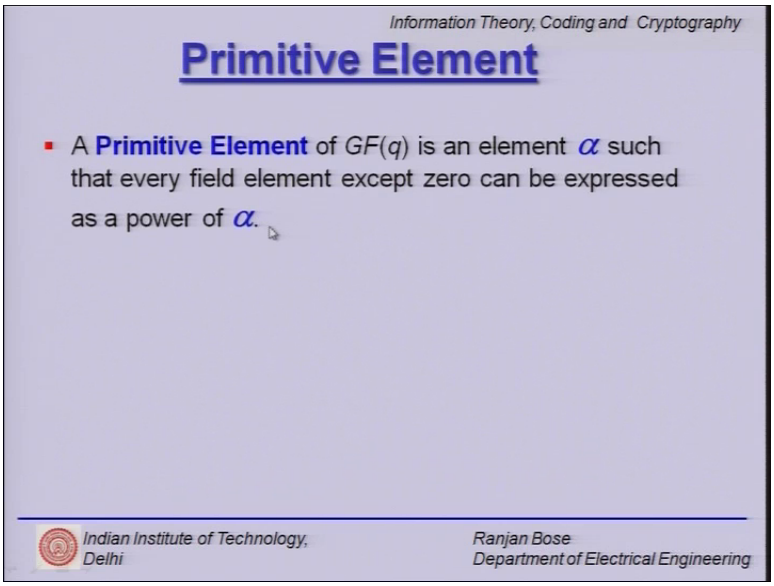
(Refer Slide Time: 02:59)



So, let us have a quick introduction, what is so different if there are subclass we need to ask ourselves what are we getting which is more than what we have studied so far so the approach is slightly different. So far our approach has been to first construct a code then find out the distance properties and based on that find out the error correcting capability that has been our sequence. In this class of codes we start from the other end, because at the end of the day we need to tell what is the error correcting capability in a earlier strategy error correcting capability came at the end, we will start from how many errors would we like to correct.

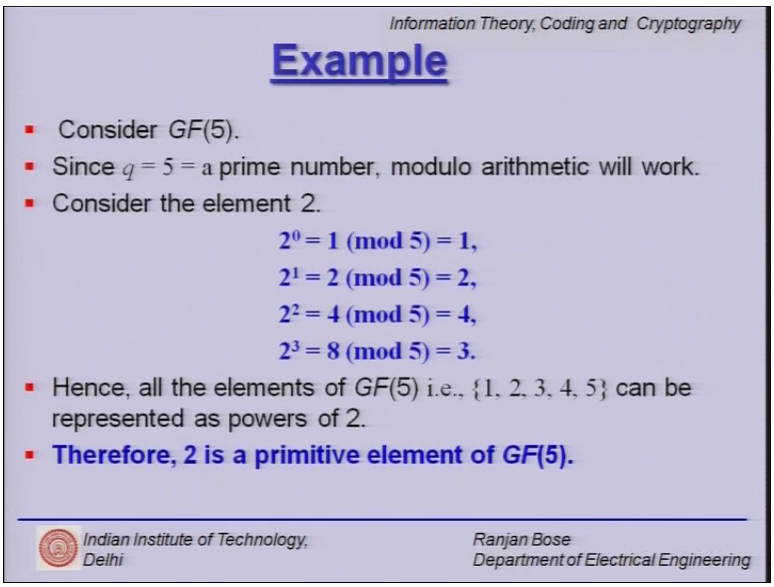
So we begin by specifying the number of random errors we desired the code to correct and then we go on to finding the generator polynomial for the cyclic code, as mentioned before BCH forms a subclass of cyclic code. So we will still have to factorise x raise power n minus 1, but what we will see is that for a certain special values of this n factorisation becomes much more easier.

(Refer Slide Time: 04:15)



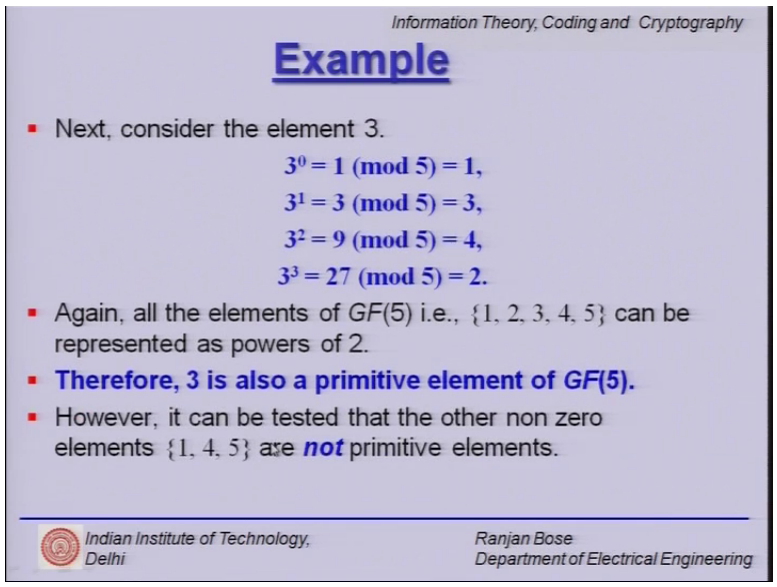
And for that we need to introduce this concept of primitive element; so what is a primitive element? A primitive element of GF q so if you remember Galois field with q elements may have at least one element alpha such that every field element except of course, the 0 element can be expressed as a power of alpha. If I can do that, then I can say that alpha is a primitive element. Please note: what is the power so a squared is a multiplied with itself of course, multiplication is a defined operation over fields so a squared is defined a cubed and so and so forth.

(Refer Slide Time: 05:04)



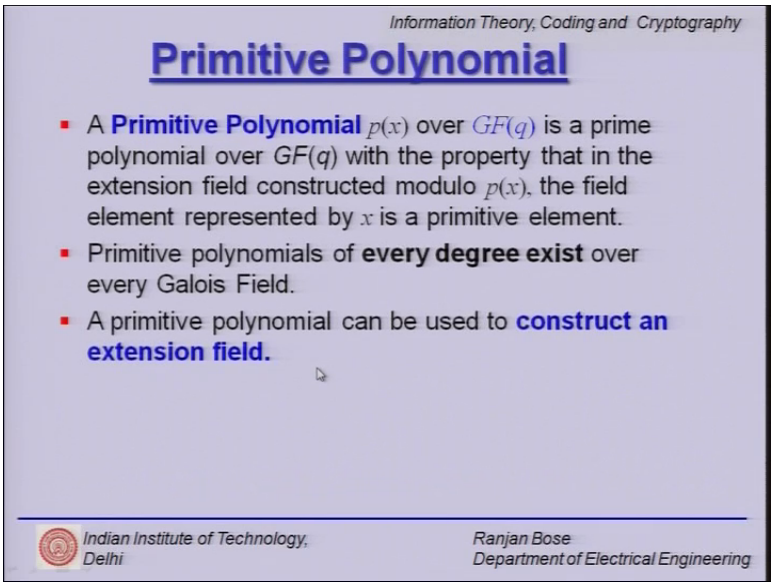
Let us do a very quick example, let us talk about GF 5 now 5 is a prime number so Galois field exist and modulo arithmetic will also work. Now let us say the elements of this field are 0 1 2 3 4 so consider the element 2 ok. Now let us look at all the powers of this element 2 the 2 raise power 0 is of course, 1 2 raise power 1 is 2 and so and so forth I get the powers as another field element. And once I raise it I can see that all the 4 elements other than the 0 element can be represented as some power of 2. Therefore, by definition this number 2 which element 2 is a primitive element so 2 is a primitive element of the feed GF 5.

(Refer Slide Time: 06:13)



Let us look at yet another example over GF 5 let us take the element 3. Now 3 raise power 0 is 1, 3 raise power 1 is 3 and so and so forth and once again we are surprised pleasantly surprised to see that all the 4 element other than 0 are represented as some power of 3; consequently 3 is also designated as 1 of the primitive elements of GF 5, but are we lucky, do we have more? Well, we can try out that all the other elements 1 4 and 0 are not really the primitive elements.

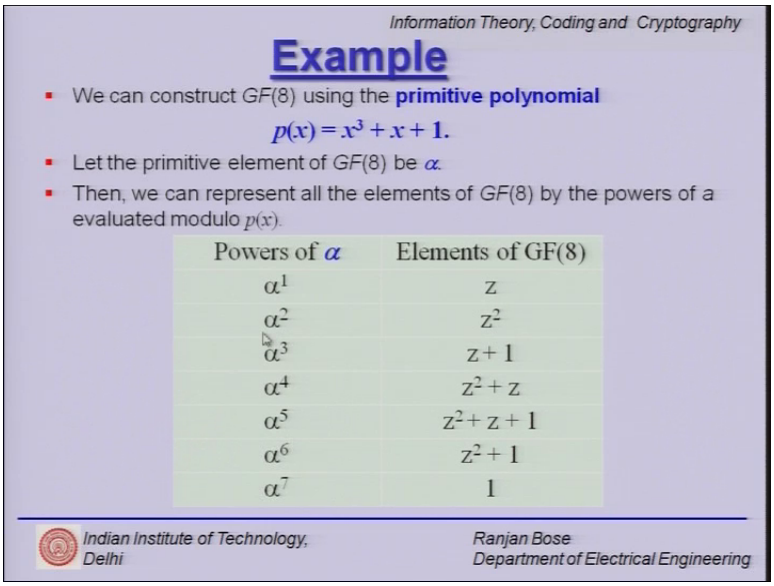
(Refer Slide Time: 06:50)



Now we go to another definition of primitive polynomial.

A primitive polynomial p of x over GF q is a prime polynomial so we already know; what is a prime polynomial you cannot have factors and it is monic over GF q. With the property that in the extension field constructed modulo p of x we have studied that you can construct an extension field using a prime polynomial, but the property here is that the extension field constructed modulo p of x the field element represented by x is a primitive element. So, under this condition you have the polynomial p of x is a primitive polynomial. Please note: primitive polynomials of every degree exist over every Galois field and if since this is the definition a primitive can easily be used to construct an extension field.

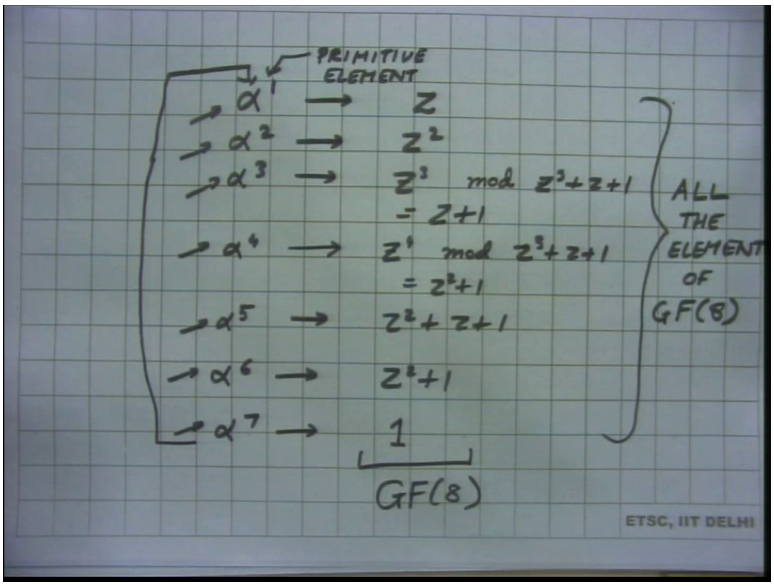
(Refer Slide Time: 07:59)



Let us look at an example because the definition was slightly convoluted. So, let us construct GF 8 using the primitive polynomial p x is equal to x cubed plus x plus 1 you can check that it is a prime polynomial over GF 2 and we wish to construct the extension field GF 8. So let us say alpha is my primitive element we know that every Galois field is at least 1 so let alpha be that 1 designated primitive element.

Then we can represent all the elements of GF 8 by the powers of alpha taken modulo p of x so let us try that say alpha raise power 1 is designated using this indeterminate z alpha squared is z square, alpha cube now so if you have to take alpha cube then we have to take modulo x cubed plus x plus 1. So if you just look at the working you have alpha raise power 1 is z, alpha squared goes as z squared, alpha cube goes as z cube, but everything is modulo your p x.

(Refer Slide Time: 09:11)

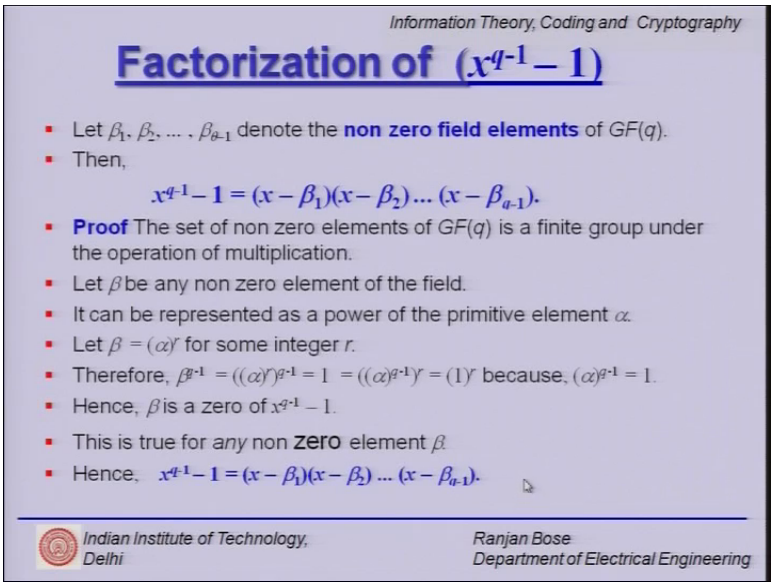


Now p x happens to be z cube plus z plus 1. So if you take z cube model z cube plus z plus 1 if you divide it you get equal to z plus 1.

Similarly, alpha raise power 4 because alpha is a primitive element so I hope it will generate each and every element of the extension field. So alpha 4 is represented as z power 4 and if you again take it as modulo z cubed plus z plus 1 then you can get nothing, but z squared plus 1 and I can do this mechanically alpha raise power 5 and I will get z squared plus z plus 1, I can do alpha 6 I will get z squared plus 1 and if I do alpha 7. Well, you will magically get it as 1 and I can keep raising it to the power and again I go back here.

So, other than 0 I have been able to generate all the elements so these are all the elements of GF 8 not only are the elements available to us, we in the process I have also constructed the addition multiplication tip and alpha is my primitive element why; because right in front of eyes we have generated using all the powers of alpha all the elements of GF 8. So we have actually constructed GF 8, after all what is the field; field is a collection of elements with the associated addition table and multiplication table.

(Refer Slide Time: 11:43)



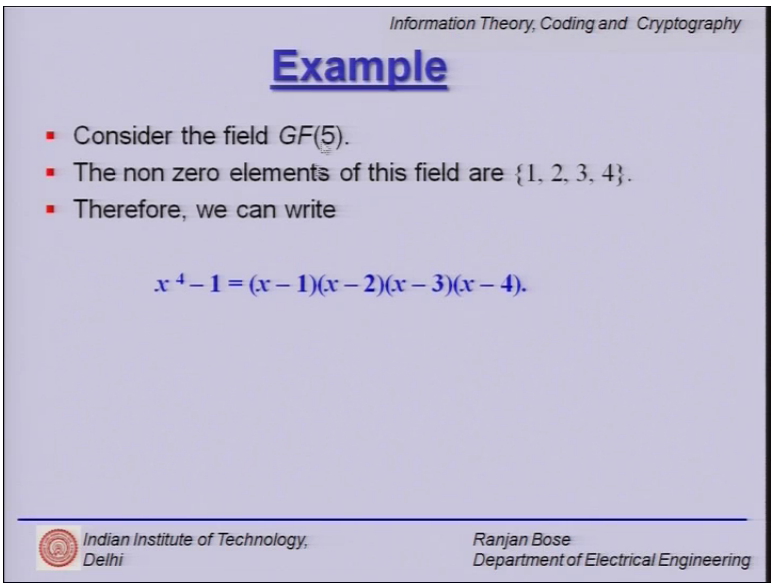
So, we come back to our slide and we pose to ourselves the fundamental question of factorising x raise power n minus 1, but before we do that what we want to do is would like to say that my n is represented as q minus 1 right now ok. So let b 1 beta 1, beta 2 and so and so forth be the nonzero field elements of GF q, then what we would like to do is we want to express x raise power q minus 1 minus 1.

So please note: this is your n because at the bottom of the calculations we wish to factorise x raise power n minus 1 that is a primary motivation; so all this match is being done to have a very efficient quick way to factorise x raise power n minus 1. So we can write x raise power q minus 1 minus 1 as x minus beta 1 x minus beta 2 so and so forth up to x minus beta q minus 1. So there is a simple proof that we can write it like this.

So, please note that beta is beta is an element of the extension field, so since beta is an element of the extension field it can be represented as some primitive element alpha raise power r for some integer r because every nonzero element in the extension field can be represented as the power of the primitive element. So beta raised to power q minus 1 you can write as alpha raise power r into q minus 1 and the you interchange these exponents and then alpha raise power q minus 1. If you remember we will go back to 1 unity we saw that case repeat and then 1 raise power r is 1 clearly therefore, this is a factor, because here I am talking about x raise power q minus 1 minus 1.

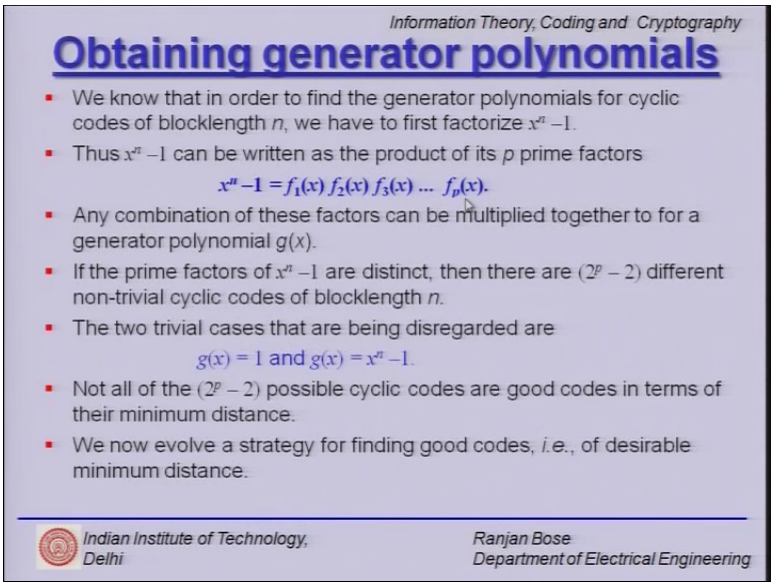
So, beta is a 0 of x raise power q minus 1 minus 1 and this is true for any nonzero element beta because, if shown for general so for beta 1, beta 2, beta 3 this is true. So I put together all these elements nonzero elements multiply them out and we have a factor. Now this immediately tells us that I can factorise a polynomial x raised to power q minus 1 minus 1 provided this q is such that the Galois field exist. So we have an immediate factorization available so far we was struggling, but here we have right in front of us this linear factors.

(Refer Slide Time: 14:53)



Let us take a simple example, let us take GF 5 which is or a Galois field with 5 element so we look at the nonzero elements 1 2 3 4 because we are leaving out 0 here therefore, without making any effort we can immediately write the factors of x raise power 4 minus 1 please note q minus 1; what is q q is 5 GF q GF 5 so we have x raise power q minus 1 show x raise power 4 minus 1. So if I have to factorise x raise power 4 minus 1 without doing any effort I can write it as x minus 1, x minus 2, x minus 3, x minus 4 the product of this. Why because I have covered all the nonzero elements of the extension this Galois field.

(Refer Slide Time: 15:40)

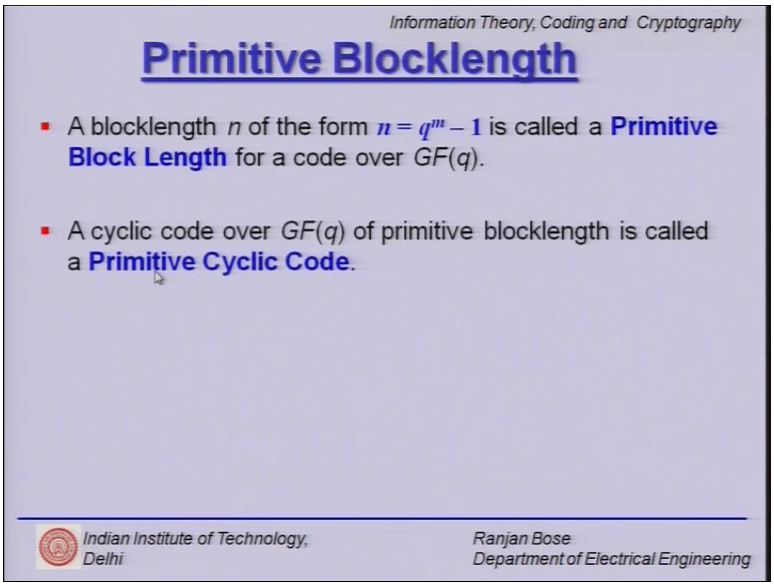


So, we now try to connect this to our factorization problem because, please note we are still in the domain of cyclic codes of block length n and r whole and sole purpose is to factorise x raise power n minus 1. So we can always right x raise power n minus 1 as some product of its prime p prime factors ok; I really do not know how many p's will be there, but there will be this prime factors.

And we remember that any factor of x raise power n minus 1 is a potential generator polynomial g x of a cyclic code; so if you can see that for this p factors there is 2 raise power p minus 2 different nontrivial codes because for example, 1 f 1 x f 2 x f 3 x so and so forth are individual, then f 1 x f 2 x f 2 x into f 3 x they are also factors, then f 1 x into f 2 x into f 3 x f 2 x into f 3 x into f 4 x. So it takes 3 at a time, then 4 at a time and 5 at a time till all of them at a time. So if you add them up you can get 2 a so p minus 2 because 1 is a factor and x raise power n minus 1 is also a factor.

So if you remove these 2 trivial cases you are left with 2 raise power p minus 2 nontrivial cyclic codes and what we will try to do is for special values of n we will be able to write this factors out completely and that is the aim of today's lecture.

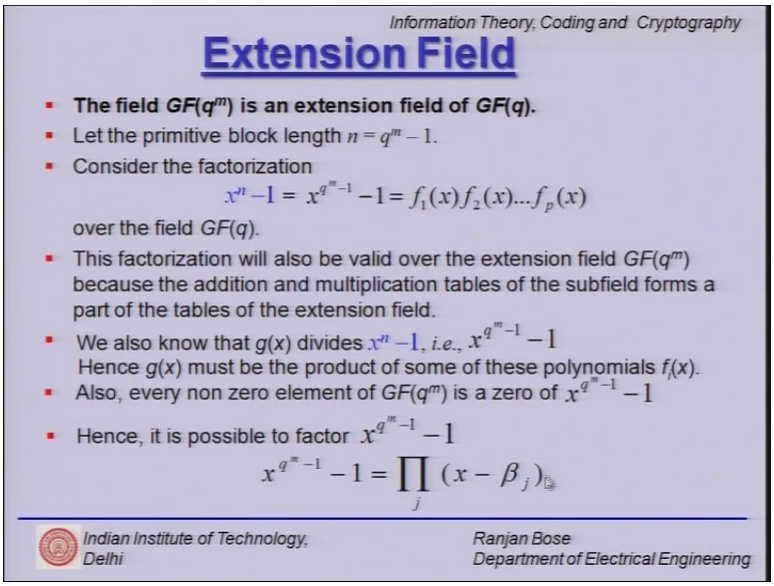
(Refer Slide Time: 17:28)



So, this n is critical and we define this special n for which factorization will be extremely easy as the primitive block length equal to q raise power m minus 1 if I am walking over GF q. So please note right in the beginning we are saying that we are willing to work with non binary codes, so it is not necessarily GF 2 yes GF 2 will be also acceptable, but in general for GF q let us talk about a primitive block length n equal to q raise power m minus 1.

So please note: for all BCH codes the starting point is a block length which probably has to be a primitive block length, so a cyclic code over GF q with primitive block length is called a primitive cyclic code.

(Refer Slide Time: 18:25)

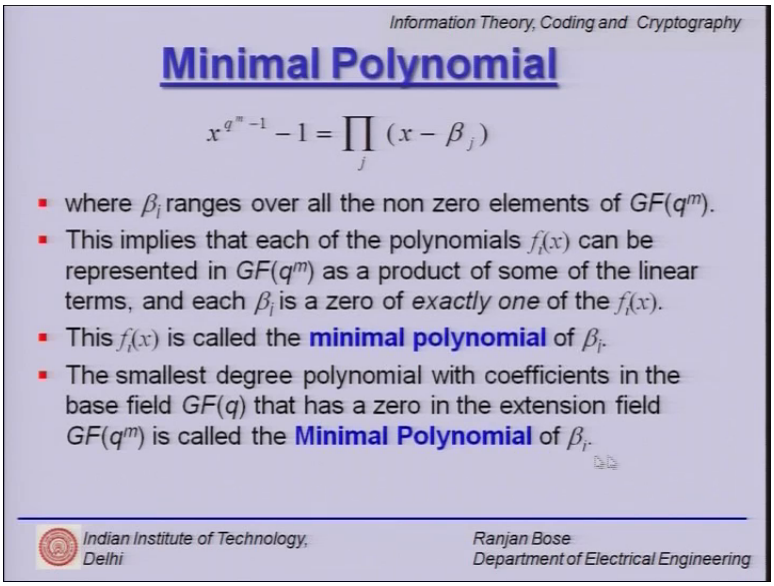


Now, we would like to work a little bit more on the extension of the factorization part and so let us say that GF raise power m is an extension field of GF q. So q is a prime number q raise power m prime power is also an extension field it has to be a Galois field by definition; so GF q raise power m is an extension field of GF q. So GF 4 is an extension field of GF 2 GF 16 is an extension field of GF 2 GF 9 is an extension field of GF 3 and so and so forth.

Now, let us start with the primitive block length n given by q raise power m minus 1 where m is some integer. So we have now our basic aim to factorise x raise power n minus 1, but my n happens to be of a primitive block length q raise power m minus 1; so I substitute the n here. So I am now in the business of factorising x raise power q raise power m minus 1 minus 1 and let us say we have this p factors ok. So what we would like to do is we have to choose a finite number of these factors in order to satisfy the distance criteria we need to work with.

So, what we do is we observe that every nonzero element of this extension field is a 0 of x raise power q m minus 1 minus 1 we just now saw that right. So we can just write x raise power q m minus 1 minus 1 as the product the linear factors x minus b j j being all the nonzero elements of the extension field we have just seen that.

(Refer Slide Time: 20:33)

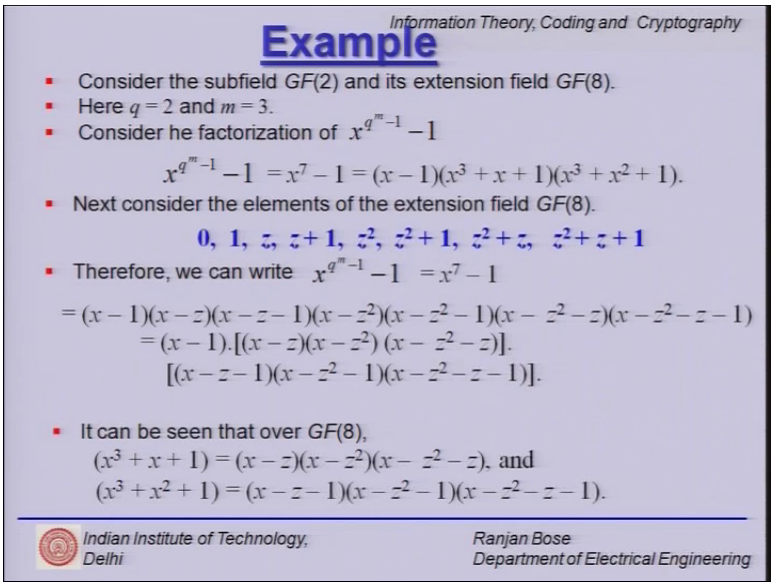


So, this can be written and b j ranges over all the nonzero elements of this extension field we just done this example.

Now, this implies that each of the polynomials f i x can be represented in GF q m and as a product of some of the linear terms; please go back and see where we are going. We are trying to factorise x raise power q m minus 1 minus 1; so this is easy factorization available here in terms of the product of the linear factors x raise to x minus b j's b j being the nonzero elements right.

Now, what we would like to do is we would like to have the polynomials f i which were the factors of my x raise power q m minus 1 minus 1 are some combination of this linear factors ok. And b j is a 0 of exactly 1 of the f i x; now this f i x is called the minimal polynomial of b j alright. So it is very clear from this so combination of that will be the minimal polynomial the smallest degree polynomial with coefficients in the base field, but has a 0 in the extension field why because b j's are the elements of the extension field is called the minimal polynomial b j.

(Refer Slide Time: 22:15)

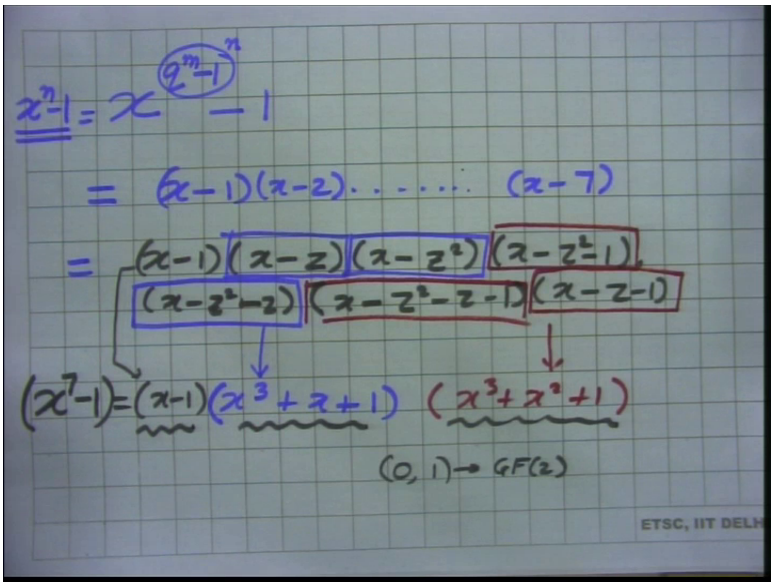


Let us look at a quick example suppose the sub field is GF 2 and the extension field is 2 raise power 3 GF 8; so consequently my q is 2 and m the integer m is 3. Now I want to factorise x raise power q m minus 1 minus 1; please note this is my primitive block length n equal to q raise power m minus 1, but q happens to be 2 m happens to be 3 so q raise power m minus 1 is 7. So I am trying to factorise x raise power 7 minus 1 over this field GF 8. Now without worrying too much I can always right if 1 2 3 4 5 6 7 are the non 0 elements of GF 8 and always right x raise 7 minus 1 as x minus 1 into x minus 2 into x minus 3 and so and so forth right, but we have a parallel factorization available.

So, please note that if you look at the construction of GF 8 using the primitive elements you have the elements of GF 8 written as the 8 these are the 8 polynomials which are the elements of GF 8. So, whenever we have these are nonzero elements we can always write the factorization of x raise power 7 minus 1 as x minus beta 1 times x minus beta 2 times x minus beta 3 and so and so forth up to x minus beta 7.

So this is exactly what it is written if you see there are 1 2 3 4 5 6 7 nonzero elements, right. And then you can expand it out and you can see that clearly x cubed plus x plus 1 is sum of these elements and x cube plus x square plus 1 is other 3 of this element, right. So, let us analyse it a little further so what are we trying to do.

(Refer Slide Time: 24:33)



We have this very clear x raise power q m minus 1 this is my m minus 1. So we are actually trying to work with cyclic codes where are starting point is always to factorise this.

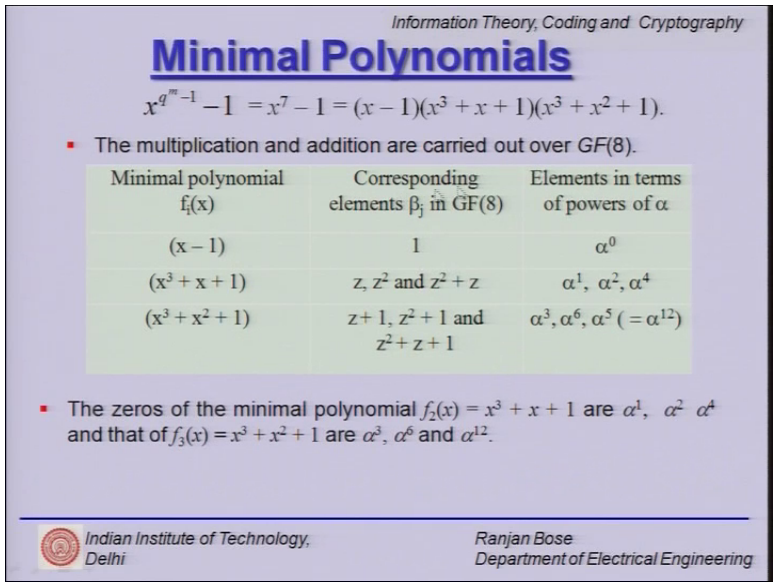
Now, over GF 8 I can always factorise it if I my elements were x minus 1 x minus 2 into x minus 7, but we can also write the elements of GF 8 using polynomials and we have r polynomials. If you construct and we have done an example before so we can write x minus 1 x minus second element is z x minus z squared x minus z square minus 1 because z squared plus 1 is a polynomial and we have again product x minus z square minus z, then we have x minus z square minus z minus 1 and then you can have 1 2 3 4 5 6, 6 of them are there and then 0 is not being included.

So, what is important is to observe that x minus z x minus z square and x minus z square minus z right these are together when you multiply them out they form x cubed plus x plus 1 right. And if you look at x minus z minus 1 so that is 1 element is missing alright; so if you look at x minus z minus 1 and then you look at x minus z square minus 1 and if you look at all the x minus z square minus z minus 1 these 2 if you multiply them out they form x cubed plus x square plus 1 and you already have x minus 1 as coming from here at this is your x raise power 7 minus 1. So, it is a matter of visualising that these linear factors could be written immediately without even thinking, but when you start

clubbing together you can get them in terms of the factors with coefficients in the base field.

So please note: that each of this polynomials have a coefficient either a 0 or a 1, so they belong to GF 2 this is important. So the coefficients of this polynomial look at the number of z's so if I take x minus z into x minus z square into x minus z square minus z; so many z's, but you multiply them out magical all the z's would cancel out because z belongs to the extension field, but here the coefficients are only in the base field GF 2 same with this kind. So it is very easy to factorise this x raise power 7 minus 1 like this ok. So we come back to our slides and we say time for some observations. So if you look at x raise for 7 minus 1 in the slide you have as we have written out x minus 1 into x cubed plus x plus 1 into x cube plus x squared plus 1.

(Refer Slide Time: 29:05)



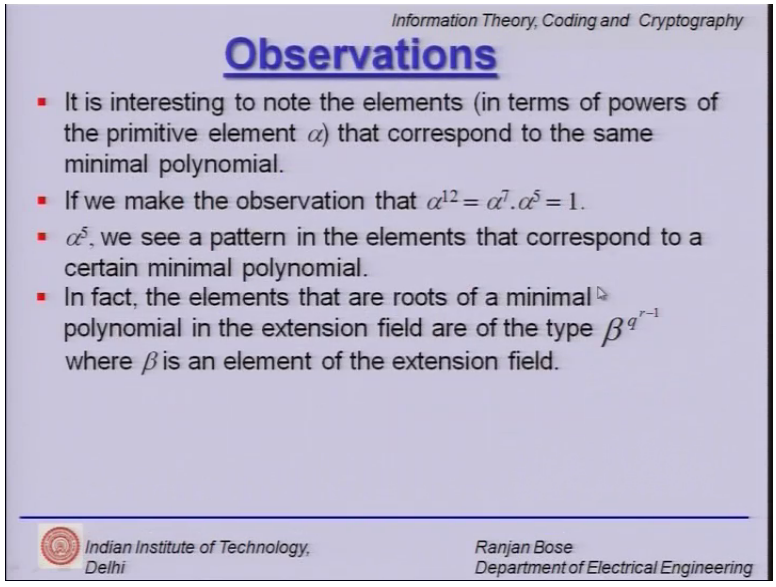
Now, we can write out the table and we see that the multiplication and additions are carried out over GF 8 and we talk about these 3 minimal polynomials; please note that minimal polynomials, if you if you look at the definition the smallest degree polynomial with coefficients in the base field. So this is what we highlighted each and every minimal polynomial has the coefficients in the base field, but at the sometimes 0 in the extension field which is obvious because it is a product of the linear factors like this. So 0's in the extension field and you have the elements which corresponds to 0's are given by this b j's. So for the first case it is product of x minus z x minus z square x minus z square

minus z so these are the 3 elements in the extension field and the other 3 elements again on the in the extension field correspond to this minimal polynomial.

Now, we make a startling observation; if you look at elements being represented as powers of alpha we note a pattern here alpha 1 alpha raise power 2 and then alpha raise power 4 here alpha cubed alpha 6 alpha 12 well alpha 12 will go back into alpha 7 into alpha 5 and alpha 7 is always 1 alpha raise power 7 is always 1 for GF 8 so it is alpha 5 here.

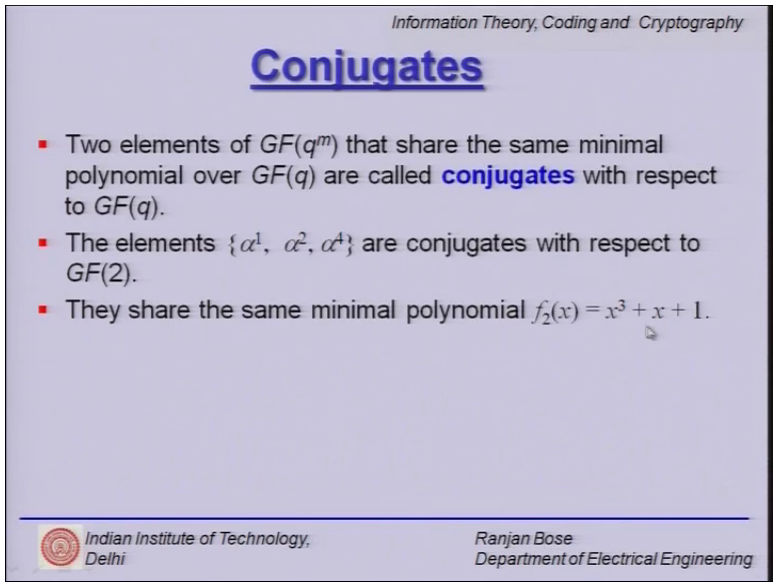
So, the observation is in the pattern that the minimal polynomial x cubed plus x plus 1 corresponds to alpha; alpha squared alpha 4 where as x cube plus x squared plus 1 corresponds to cubed 6 and alpha 12. So many things are going to be much easier I do not even have to worry which 1 to take if I can just observe that the pattern; please note the element and powers of alpha have a one to one correspondence because alpha is the primitive element.

(Refer Slide Time: 31:19)



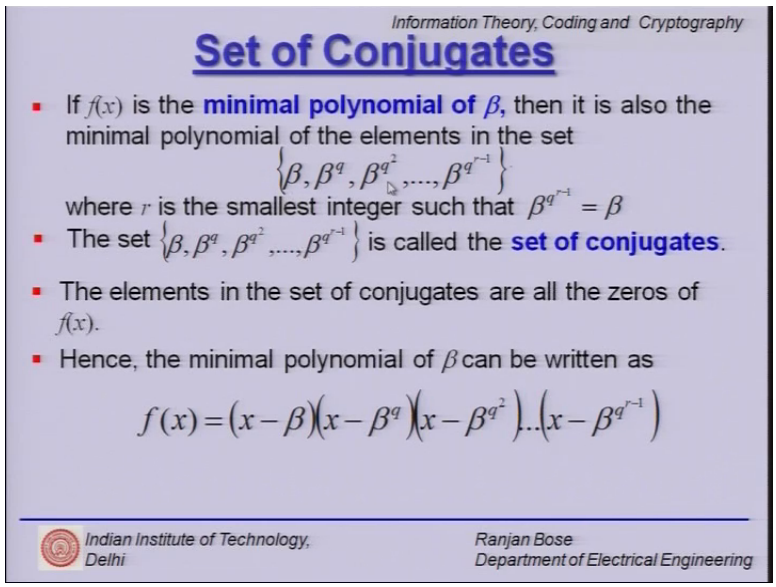
So, we have made this observation that the elements are the roots of a minimal polynomial in the extension field which of the type beta raise power q r minus 1; where beta is a an element of the extension field right. So if alpha would if you want to write alpha as a primitive element it is alpha raise power q r minus 1; what is q q was 2 r was 3 so we can always right in terms of alpha raise power 2 raise power 3 minus 1.

(Refer Slide Time: 31:58)



Now, 2 elements of GF q m that share the same minimal polynomial over GF q are called the conjugates with respect to GF q for example, you have seen that alpha raise power 1 alpha raise power 2 alpha raise power 4 have the same minimal polynomial right. Similarly we had seen in the previous slide that alpha cubed alpha 6 alpha 12 share the same minimal polynomials; so they are also conjugates.

(Refer Slide Time: 32:36)

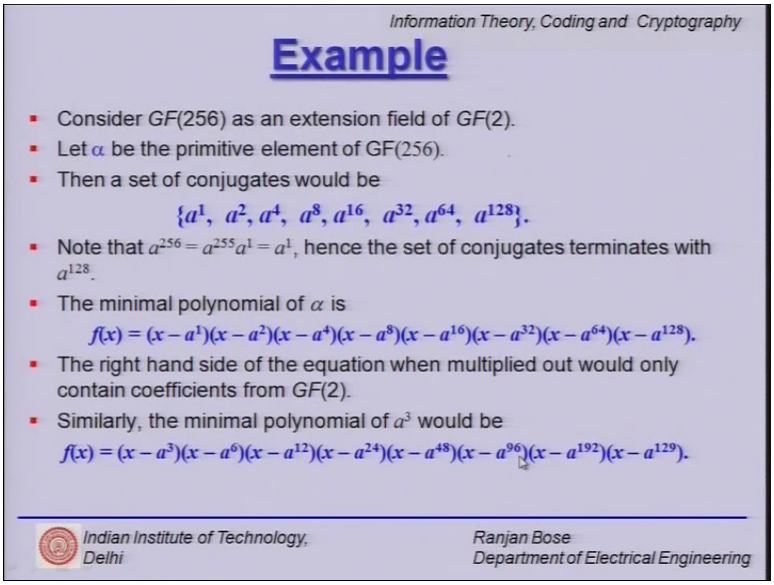


So, f x if f x is a minimal polynomial of beta then it is also the minimal polynomial of the elements of the set beta raise power q. So since beta was earlier there then beta square

and then beta 4 and so and so forth up to beta raise power q r minus 1, where r is the smallest integer such that beta q raise power r minus 1 is equal to beta and this is called the set of conjugates. So in the earlier example alpha, alpha squared, alpha raise power 4 was the set of conjugates.

So, now we make our lives even more simpler, the minimal polynomial of beta can simply be written as x minus beta x minus beta q x minus beta q squared and so and so forth after x minus beta q raise power r minus 1.

(Refer Slide Time: 33:42)

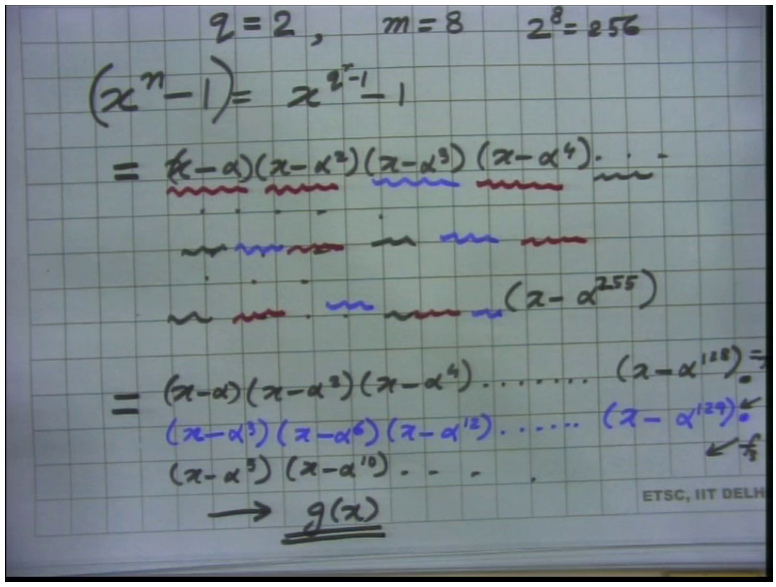


So, let us look at simple example consider GF 256; now if you try to do this by the standard technique you may have to work a little harder, but GF 256 as an extension field of 2 is can be easily factorized. So suppose I want to look at alpha being the primitive element of GF 256. So without even worrying too much I can say oh sure the first set of conjugates I can write as alpha alpha squared alpha 4 alpha 8 16 up to alpha 128 ok. So you can do a check that alpha 256 could have been the next element, but 256 is equal to alpha q minus 1 now alpha raise power q minus 1 is always unity so alpha 255 is 1 into 1 so alpha 1 so then it will start repeating cyclically; so we terminate our search beyond this point.

So, what is the minimal polynomial of alpha? Well straight away without any sweat x minus alpha x minus alpha squared x minus alpha 4 x minus alpha 8 so and so forth up to x minus alpha raise power 128.

Now, what about alpha and alpha squared or covered alpha cubed is not covered we went to alpha alpha squared alpha cubed is not covered. So what will be the minimal polynomial of alpha cubed well alpha cube alpha 6 alpha 12 alpha 24 so and so forth till alpha 129 and so and so forth and please note that if we have to factorise x raise power m minus 1 where m is a primitive block length then I need to have all of the alpha's in business. So let us look at what we are trying to say here.

(Refer Slide Time: 35:55)



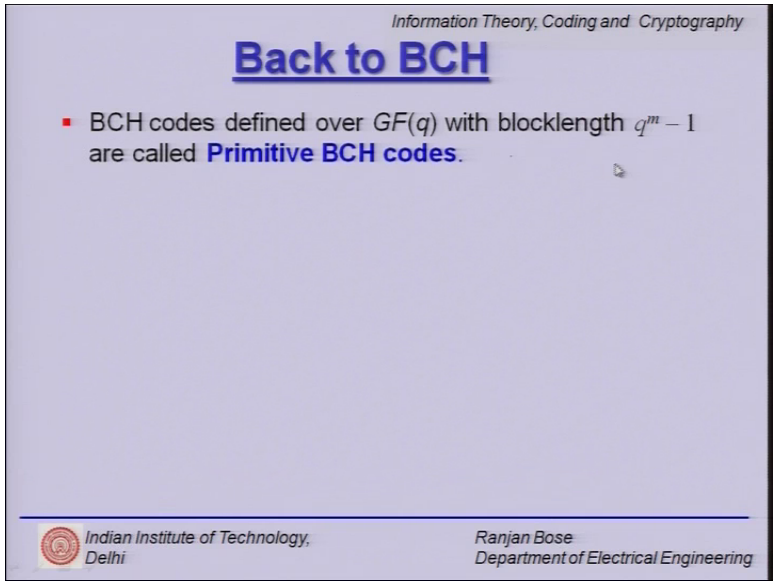
So, if you look at factorising x raise power n minus 1 right; in this example q is equal to 2 right and so we have m so m is equal to suppose 8; so we have 2 raise power 8 has 256. So if you have we have to factorise this all we have to do is you can keep writing till you have x raise power alpha raise power 1 256 so 255 and that is it; we have immediately written all the factors. Now if you have to write it out in terms of minimal polynomial all you have to do is choose the right candidates so alpha alpha square alpha 4 alpha 8 somewhere alpha 16 and so and so forth you choose the right candidate and you get the first set of to get x minus alpha x minus alpha squared x minus alpha 4 x minus alpha 128.

Now if you multiply them out all the alphas will disappear, because your base field is GF 2 and you will be left with some big polynomial with coefficients either 0 or 1, but we have just re grouped. Now we are not exhausted so multiplied by and then we have x minus alpha cubed is missing, so alpha cube x minus alpha 6 x minus alpha 12 and keep

going till x minus alpha 129. And then you have actually exhausted some of the other ones right and then you have 1 2 3 4 maybe 5 is not there because 5's are not coming so you have got x minus alpha 5 then x minus alpha 10 and so and so you know you can till we exhaust all of them so 5 will start exhausting.

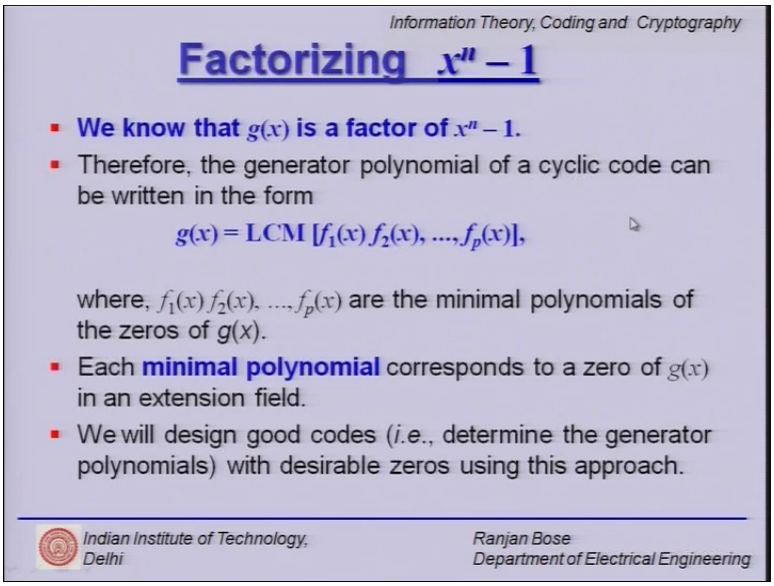
And finally, we will exhaust all of these factors and we will write the factorization in terms of the minimal polynomial. So this first guy this first product will be your f 1 x this will be your f 2 x and so and so forth. So this what are we trying to do we are trying to factorise this x n. And hopefully these guys would lead to me the generator polynomial; so my final aim is to get g of x and that is all we need for cyclic codes so we go back to our slide.

(Refer Slide Time: 39:47)



And we go back to our BCH codes; so BCH codes defined over GF q with block length q raise power m minus 1 are called primitive BCH codes and why this primitive block length is important is because it can help us factorise very easily.

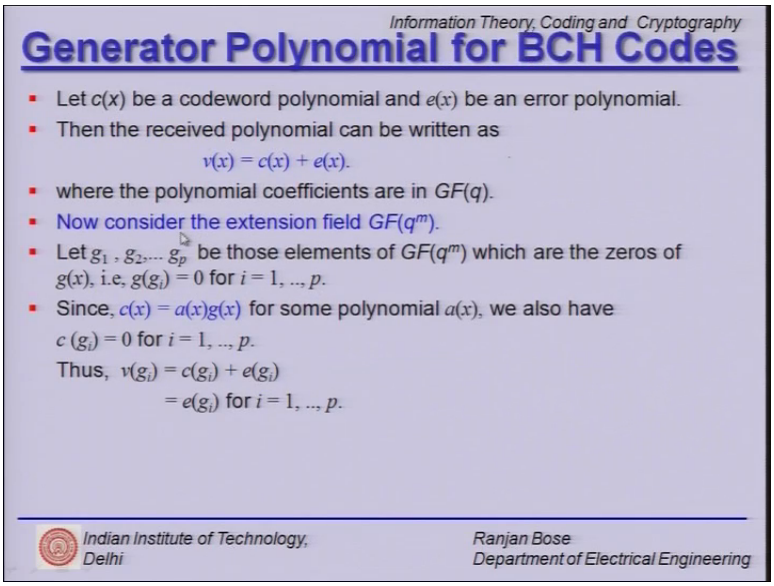
(Refer Slide Time: 40:03)



So factorising x raise power n minus 1 that has been a job right from the beginning we know that g x is a factor of x raise power n minus 1 therefore, the generator of a cyclic code can be written in the form g x is equal to LCM of f 1 x f 2 x up to f p x where f 1 x f 2 x are the minimal polynomials of the zeros of g of x ok. So now, we have suddenly we have got in this product in terms of the minimal polynomials we did this example just now.

But what is great is each minimal polynomials corresponds to 0 of g of x in the extension field ok. So we will design good codes with desirable zeros using this approach so our approach is in terms of a designing g of x ok. We are now in the business of designing g of x by choosing appropriate minimal polynomials.

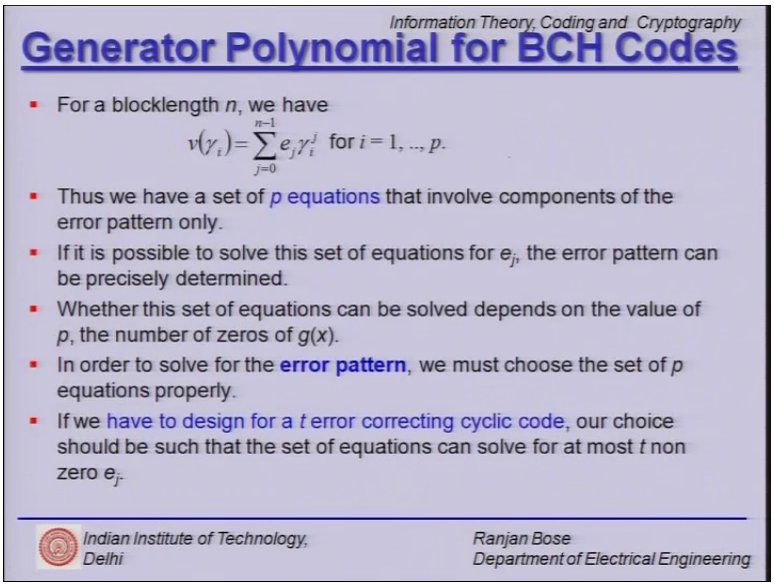
(Refer Slide Time: 41:01)



So, let us look at very quickly how thing will work and you will see the logic very quickly let c of x be a code word polynomial and e x be an error polynomial; then the received vector is always a polynomial representation v x is equal to c x plus e x where the polynomial coefficients are over GF q.

Now, we look at the extension field GF q raise power m let g 1 g 2 up to g p be those elements of GF q raise power m which are the 0's of g of x that is g of g i is 0. Now why do we need that well c x is some polynomial a of x into g of x so if g i is 0 of g like this then g i must be a 0 of c therefore, the received nu g i is nothing, but c g i plus e g i, but c g i is 0 is nothing, but e g i; very similar to the syndrome version that we decided to do even for linear block codes and cyclic codes.

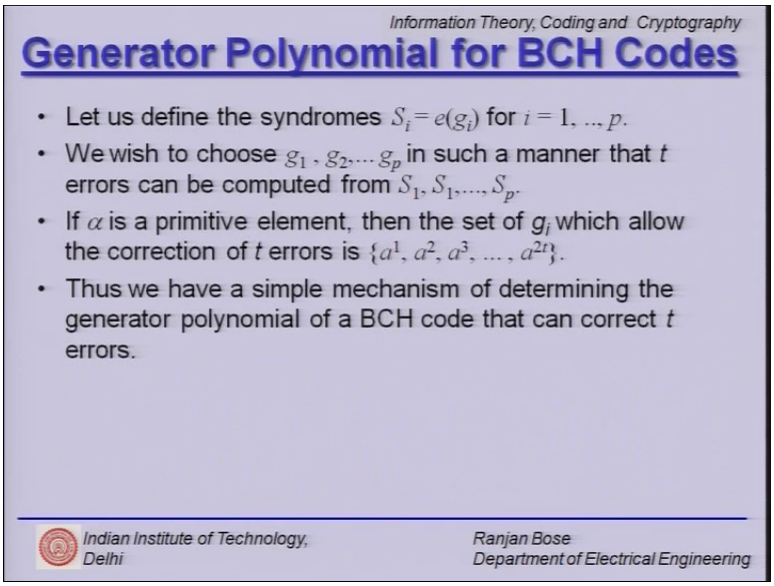
(Refer Slide Time: 42:18)



So, now we have a block length n and we have got now nu gamma i is nothing, but this is a polynomial representation e j gamma i j so eventually we have a set of p equations that involve components of the error patterns only is a syndrome. So now, we have a system of equations if it is possible to solve the set of equations for e j, the error apparent can be precisely determined because we are in the business of error correction.

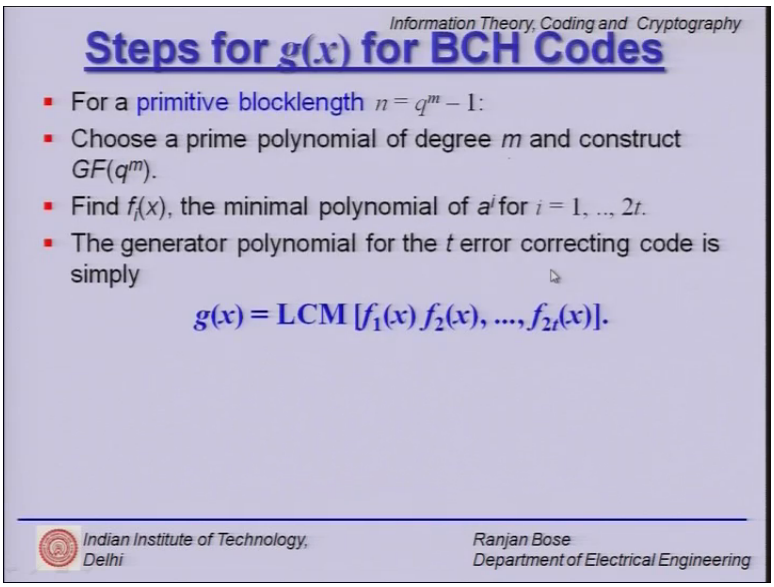
So, whether the set of equations can be solved depends on the value of p that the number of 0's of g of x. So therefore, in order to solve the error pattern we must choose a set of p equations properly. So know let us move to our attention towards trying to design a t error correcting code how many equations do we need to correct t errors.

(Refer Slide Time: 43:26)



So, let us define the syndrome s i is nothing, but error polynomial of g i; so we defined that. So we wish to choose the 0's g 1 g 2 up to g p in such a manner that t errors can be computed from S 1 S 2 up to S p. So if alpha is a primitive element then the sets of g i allow the correction of t errors are simply alpha raise power 1 alpha raise power 2 alpha q up to 2 t ok. So it clearly shows that I have enough equations to now determine the error pattern uniquely. Secondly, we have a handle if we need to correct errors we need to have more equations.

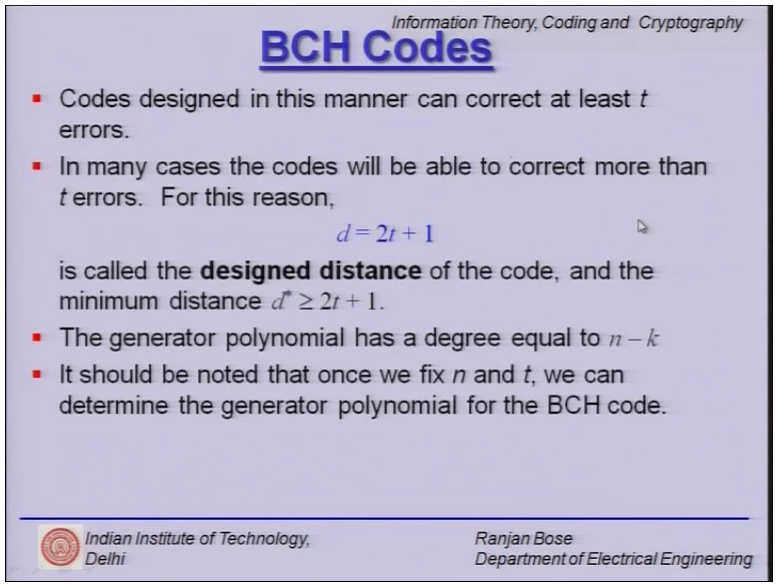
(Refer Slide Time: 44:11)



So, then it tells us a very simple way to get the generator polynomial for BCH codes. So first starting point is that the primitive block length should be honoured, so n must be q raise power m minus 1 then all the theory that have developed holds good. So we choose a prime polynomial of degree m and construct GF q m it is now very easy so the first thing is that you need a prime polynomial which will be the primitive polynomial which will be used to construct the extension field. Then find the minimal polynomials for alpha raise power i, where i is equal to 1 2 up to 2 t. Please note that by now we are very comfortable finding the minimal polynomials, because we know how to make the set of conjugates.

And then the g of x the generator polynomial is simply given by LCM of f 1 x f 2 x and so and so forth; why it LCM because sometimes f 1 x and f 2 x will be the same it depends on your Galois field, but you do not want to repeat it because g g x is of the lowest degree therefore, we take that LCM. So we have now a very simple way for finding a t error correcting code. So starting point is n and t you tell me how many errors you want me to correct and you tell me the primitive block length with that I will give you a BCH code which will guarantee you to correct at least t number of errors.

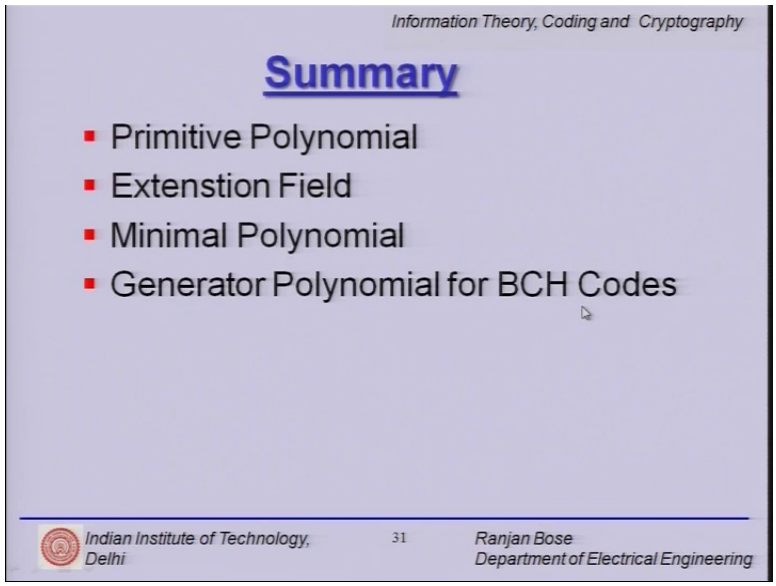
(Refer Slide Time: 45:53)



So please note: code designed in this manner can correct at least t errors therefore, it is called the designed distance d which is equal to 2 t plus 1 and some time it is greater than the minimum distance. And it goes without saying that the generator polynomial has a

degree n minus k so we started with n and t and then we decided how many minimal polynomials to pick up and then k comes automatically. So gets decided later on, first you find out n and then t and then k so once we fix n and t we can determine the generator polynomial of the BCH codes.

(Refer Slide Time: 46:40)



So, we now come to the conclusions for today's lecture what have we studied today; we started off with primitive polynomials and then we learnt how to create an extension field. Then we started minimal polynomials and finally our final result of how to determine the generator polynomials for BCH codes.

With that we come to the end of this lecture.