Cryptography and Network Security

Prof. D. Mukhopadhyay

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

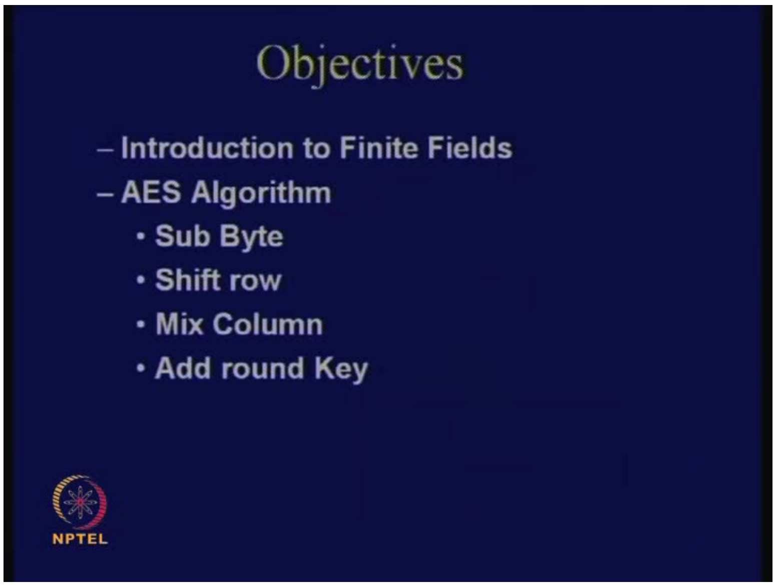
Module No. # 01

Lecture No. # 12

Block Cipher Standards (AES)

Welcome to today's class. Today, we will discuss about Modern Block Cipher Standards. In last day's class, we have discussed about the DES algorithm. So, today, we will take up a new standard, which is followed worldwide. It was adopted as a standard around 2000 – 2001. The algorithm is known as the Advanced Encryption Standard, but the name of the algorithm is Rijndael, given by the names of the designers of the algorithm.

(Refer Slide Time: 00:48)

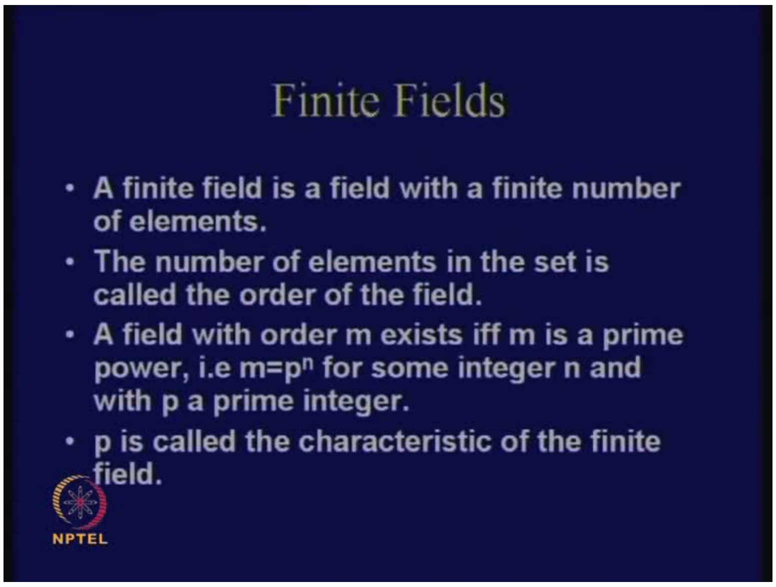


Objectives

- Introduction to Finite Fields
- AES Algorithm
  - Sub Byte
  - Shift row
  - Mix Column
  - Add round Key

Today's objectives are as follows. We shall discuss about finite fields because finite fields form a very important component of the AES algorithm. It is important to understand the field theory or finite field theory in order to understand the intricacies or the internal workings of the algorithm. After introduction to finite fields, we shall discuss about the internal algorithms of the algorithm steps of the AES algorithm. Essentially, AES has got four steps like the sub byte step, the shift row, mix column and followed by add round key.

(Refer Slide Time: 01:30)



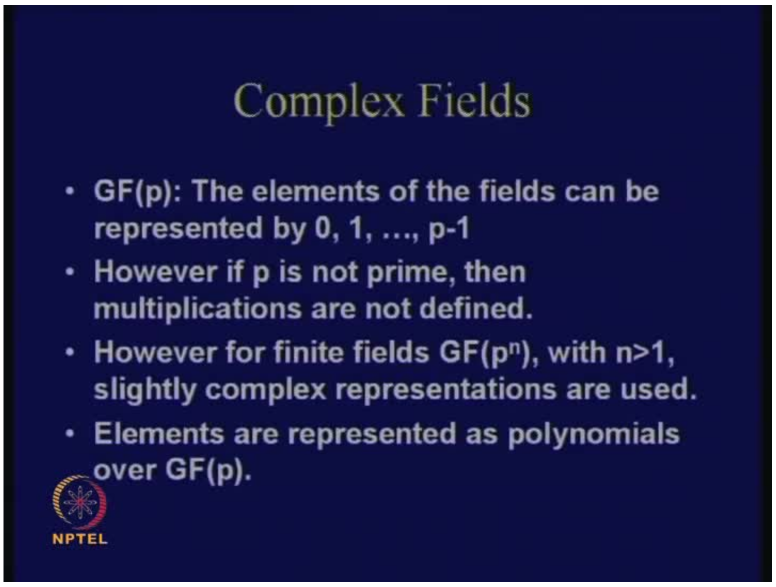
Finite Fields

- A finite field is a field with a finite number of elements.
- The number of elements in the set is called the order of the field.
- A field with order m exists iff m is a prime power, i.e $m=p^n$ for some integer n and with p a prime integer.
- p is called the characteristic of the finite field.

Talking about finite fields, a finite field is a field with a finite number of elements. The number of elements in the set is called the order of the field. Therefore, essentially what we are talking about here is that we know that we talk about integer numbers for example. In case of integers, we do operations like addition, multiplication, but for example, the number of elements that we have in the integer set is infinite. However, on the contrary, in case of finite fields, we shall have finite number of elements. That means we have to do multiplication operation, addition operation in a set, which has got finite number of elements. So, that has got natural consequences as we shall see in this class itself.

A field with order m exists if and only if m is a prime power, that is, m is equal to p to the power n for some integer n and where, p is a prime integer. So, in case of finite fields, we essentially refer this variable p as the characteristic of the field. Therefore, as we shall see in our discussion on the AES algorithm, the characteristic of the finite field of the AES algorithm is p equal to 2.

(Refer Slide Time: 03:10)



Complex Fields

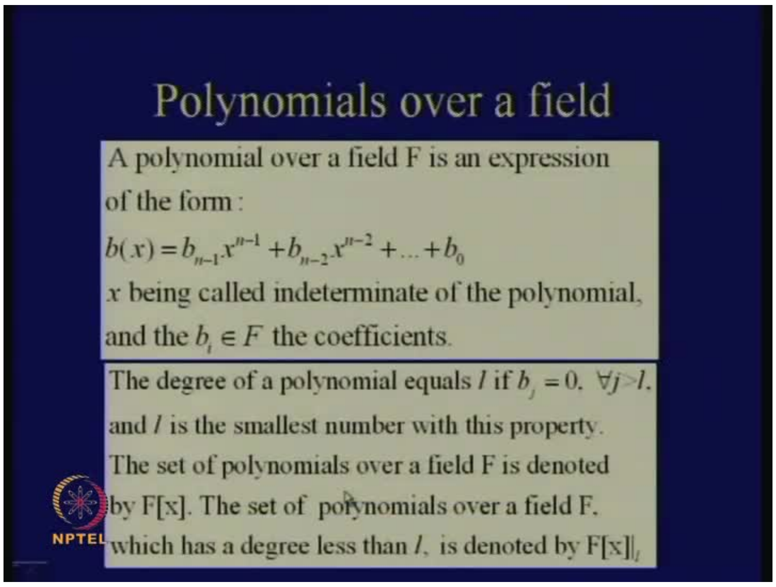
- GF(p): The elements of the fields can be represented by 0, 1, …, p-1
- However if p is not prime, then multiplications are not defined.
- However for finite fields GF(p^n), with n>1, slightly complex representations are used.
- Elements are represented as polynomials over GF(p).

Therefore, we shall find that the internal field of the Rijndael algorithm or the AES algorithm is GF 2 to the power 8. Finite fields are often refer to as the Galois field also and it is denoted by the letter GF. Therefore, GF p means that we have got the number of elements that are inside the GF p are from 0 to p minus 1; that is, 0,1 to p minus 1. So, the elements of the fields can be represented as 0, 1, p minus 1. In this case, the point to be noted is that if p is not a prime number, then multiplications are not defined. This follows from a natural result from number theory, which says that the multiplicative inverse of an element a modulo m exists if and only if the gcd of a and the modulo m is equal to 1. What I mean to say is that when I am considering the numbers inside say modulo m, I am considering the inverse of a modulo m; that is, a inverse modulo m. Then, a inverse modulo m exists if and only if the gcd of a and m is equal to 1; that is, a and m are mutually co-prime.

This result follows from the equilibrium theorem, which we shall see in our future classes, but right now let us accept this as a result. Therefore, the idea is that if p is a prime number, then all the elements starting from 1 to p minus one are essentially co-prime with p. Hence, every element has got a multiplicative inverse. Thus, that fulfills the criteria or definition of a field.

However, if p is not a prime number, then multiplications are not defined. Therefore, the inverse of an element; that is, a multiplicative inverse may not exist. However, when we are considering say finite fields like GF p power n, with n greater than 1, if you use slightly complex representations, then actually we can define the idea of a field as well. In that case, we shall see that the elements have to be represented in a form, which is a polynomial notation. In that case, we can do multiplications as well as addition operation. So, how does a polynomial over a GF p look like?

(Refer Slide Time: 05:35)



Polynomials over a field

A polynomial over a field F is an expression of the form:

$b(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \ldots + b_0$

x being called indeterminate of the polynomial, and the $b_i \in F$ the coefficients.

The degree of a polynomial equals l if $b_j = 0, \forall j > l$, and l is the smallest number with this property. The set of polynomials over a field F is denoted by F[x]. The set of polynomials over a field F, which has a degree less than l, is denoted by $F[x]|_l$

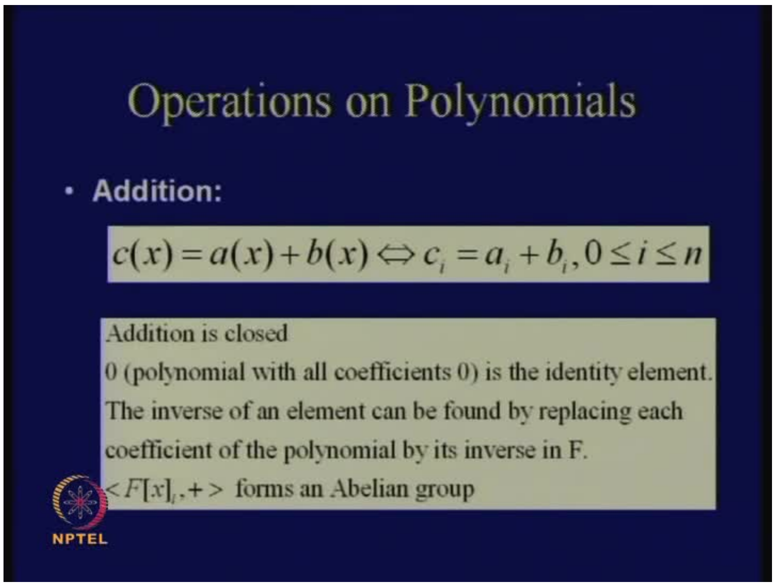
It looks like this. Therefore, consider a polynomial over a field F. It can be denoted by an expression of the form b x is equal to b n minus 1 x to the power of n minus 1 plus b n minus 2 x to the power of n minus 2 plus so on plus b 0. x being called indeterminate of polynomial and the b i; that is, the coefficient of this polynomial are belonging to the underlined field. Therefore, for example, when you consider a polynomial in GF 2 to the power 8 or GF 2 to the power n, then each of these coefficients would belong to the field GF 2. That means each of these elements that GF 2 has got are 0s and 1s. So, this means that when I am considering a GF 2 to the power n polynomial, then the coefficients are either 0 or 1. Here often x is referred to as the indeterminate of the polynomial.

There is a term called degree, which we know for example, for polynomials. We say that a polynomial has got a degree l if b j; that is, the coefficient b j is equal to 0 for all j, which is greater than l. Therefore, l is the smallest number with this property. That means the set of these polynomials over a field is often referred by this notation (Refer Slide Time: 07:03) F x. The set of polynomials over a field F, which has got a degree which is less than l, which is denoted by F x suffix l. F x suffix l means that I am referring to the set in entire set of the polynomials over a field F whose degree is less than l.

Another importance of this particular notation (Refer Slide Time: 07:27) when I am talking about Galois field or finite field is that this polynomial can be stored in the form of a register. Therefore, if we would like to represent this in a computer, then we can store this in an array, where each of these elements would have belonged to this underlined field, F. What happens when we are referring to the field GF 2 to the power n?

The entire polynomial can be stored in an n bit register, where each of the bits are either 0s or 1s. Therefore, we see that this representation (Refer Slide Time: 08:01) in GF 2 to the power n has got a very natural linkage or connotation with the normal computer arithmetic. Therefore, that is a precise reason why GF 2 to the power n arithmetic is very easily available to nice computer architectures. So, we can have nice multipliers, nice inverse architectures, which leads to highly efficient architectures for cryptography for communication. That is a precise reason why this field GF 2 to the power n has got lot of interesting applications in both communications as well as cryptography.

(Refer Slide Time: 08:39)



Operations on Polynomials

- **Addition:**

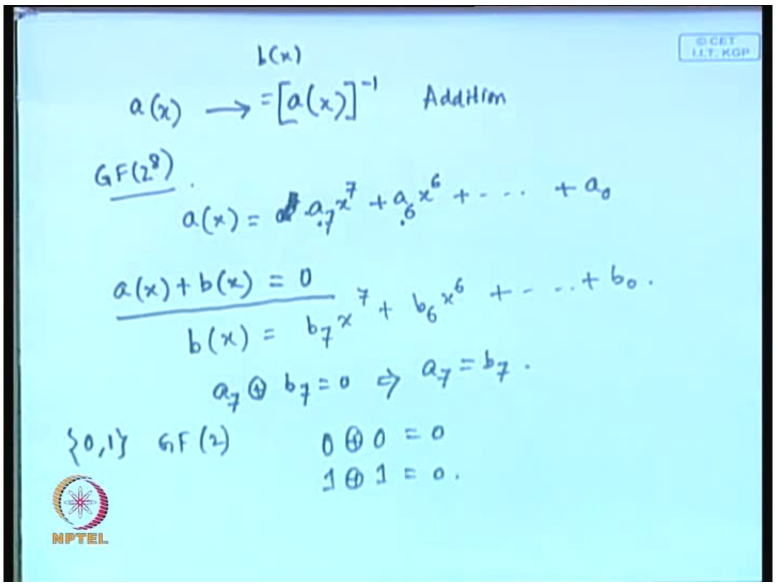
$$c(x) = a(x) + b(x) \Leftrightarrow c_i = a_i + b_i, 0 \le i \le n$$

Addition is closed
0 (polynomial with all coefficients 0) is the identity element.
The inverse of an element can be found by replacing each coefficient of the polynomial by its inverse in F.
$< F[x]_l, + >$ forms an Abelian group

The moment we have these polynomials, what are the polynomials? The polynomials are elements of a field. What is our objective? We would like to define addition and multiplication operations on these polynomials. How does an addition works? It is quite simple.

If you would like to do an addition over the polynomial a x and b x, then how do we obtain c x? It is simple addition. However, that means for every i, what we are doing is that we are adding coefficients. So, take a i and b i; add them and store them in the form of c i. That you do for all from 0 to n minus 1; that is, for all the values of i. Here, the point to be noted is that addition is quite closed. That is, for example, consider the degree of a x is greater than the degree of b x. Then, the degree of a x plus b x cannot be greater than that of the highest degree between a x and b x. That means the degree of c x is always lesser than the higher degrees between a x and b x. So, that means c x is always closed. That means addition is essentially closed. Also, we note that for example, if we have one of the polynomials with all their coefficients as 0 and that we often refer to that as the element 0, then 0 forms the identity element under the addition operation.

(Refer Slide Time: 10:24)



$b(x)$

$a(x) \rightarrow = [a(x)]^{-1}$ Addition

$\underline{GF(2^8)}$

$a(x) = a_7 x^7 + a_6 x^6 + \dots + a_0$

$\underline{a(x) + b(x) = 0}$

$b(x) = b_7 x^7 + b_6 x^6 + \dots + b_0$.

$a_7 \oplus b_7 = 0 \Rightarrow a_7 = b_7$.

$\{0, 1\}$ $GF(2)$

$0 \oplus 0 = 0$

$1 \oplus 1 = 0$.

The inverse of an element can be found by replacing each coefficient of the polynomial by its inverse in F. Therefore, for example, if you take the element a x, I would like to form the inverse of a x; that is, I would like to compute the inverse of a x under the addition operation. How do we do that?

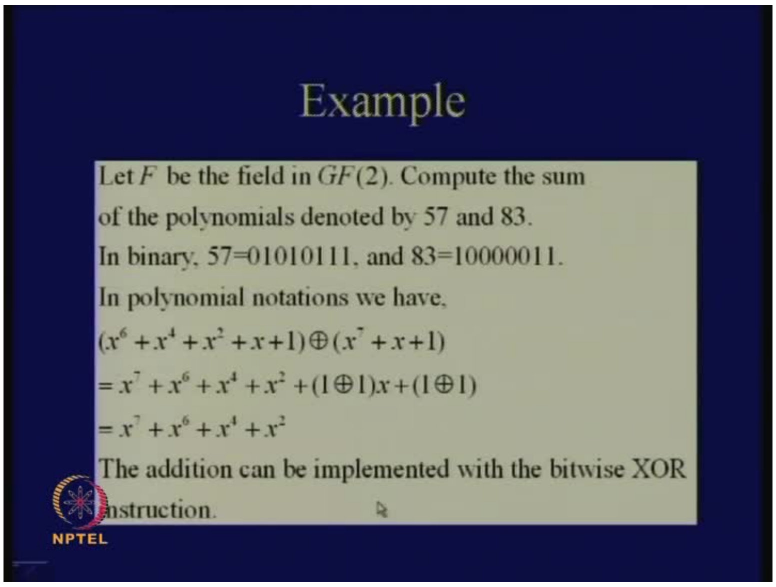
What we do is that for example, let us consider the example of GF 2 to the power 8. In GF 2 to the power 8, an element a x can be written as a 7 x to the power of 7 plus a 6 x to the power of 6 plus so on till a 0. I need to form the inverse of a x, which means that for example, say let this be equal to b x. Then, a x plus b x should go to 0. When I am forming b x, I can also denote b x like b 7 x to the power 7 plus b 6 x to the power 6 plus so on till b naught.

If I would like to satisfy this equation, (Refer Slide Time: 11:43) a x plus b x is equal to 0, then automatically it means that I have the identity that a 7 XORed with b 7 is equal to 0. This implies that a 7 and b 7 are the same things. What does that mean?

That means when I am trying to form the inverse of a x; that is, the additive inverse of a x, then in order to do that I replace each coefficient of the original polynomial a x by its additive inverse in the field GF 2. In GF 2, for example, every elements is self inverse. What I mean to say is that in GF 2, we have two elements: 0 and 1; 0 is the self-additive inverse of 0 and 1 is the self-additive inverse of 1. Why? Because 0 XORed with 0 is equal to 0 (Refer Slide Time: 12:34) and 1 XORed with 1 is also equal to 0.

Coming back to our operation on polynomials, (Refer Slide Time: 12:42) we can see that when I am talking about forming the inverse of an element, the inverse of an element can be found by replacing each coefficient of the polynomial by its inverse in F. Then we say that F x l; I have already defined what F x suffix l means. F x suffix l; these set of polynomials with their degree l under the addition operation forms an Abelian group. Why an Abelian group? That is because it is also commutative; that is, a x plus b x is equal to b x plus a x. So, it is as well as it is also commutative. What about multiplication?

(Refer Slide Time: 13:24)



Example

Let F be the field in $GF(2)$. Compute the sum of the polynomials denoted by 57 and 83.

In binary, 57=01010111, and 83=10000011.

In polynomial notations we have,

$(x^6+x^4+x^2+x+1)\oplus(x^7+x+1)$

$=x^7+x^6+x^4+x^2+(1\oplus1)x+(1\oplus1)$

$=x^7+x^6+x^4+x^2$

The addition can be implemented with the bitwise XOR instruction.

It is not so straight forward in the case of multiplication. However, before going to multiplication, let us consider one example for addition. Let F be the field in GF 2 and we would like to compute the sum of the polynomials, which are denoted by 57 and 83. First of all, let us denote 57 and 83 in binary notation. This is how it looks like. In polynomial notations, I can express 57 as this.

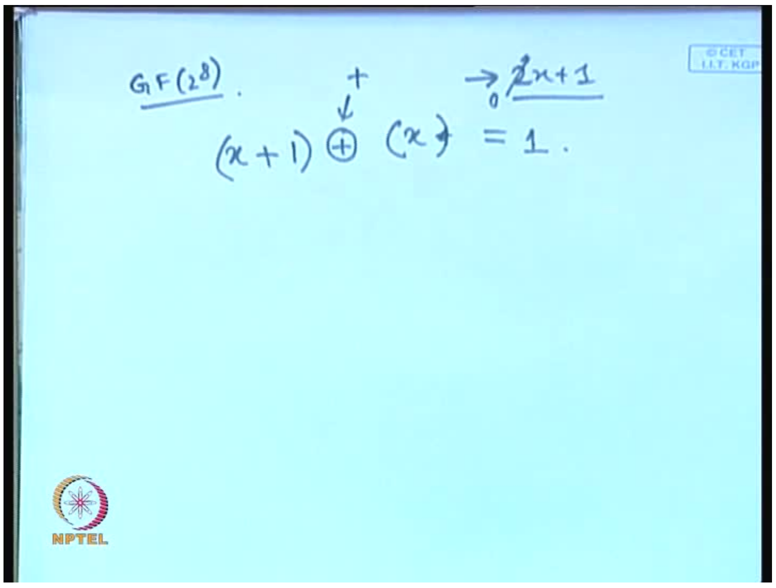
I think all of us have convinced because you see that this 1 (Refer Slide Time: 13:51) forms the coefficient of your constant. This 1 forms the coefficient of the variable x. This 1 forms the coefficient of x square. Then, the coefficient of x to the power 3 is 0. Therefore, you do not have an x to the power 3 term here (Refer Slide Time: 14:07). However, here you have 1. So, this 1 forms x to the power 4 and this 1 forms x to the power 6. So, you can see x to the power 7 and x to the power 8; x power 7 and x power 5 are the coefficients of them and still are missing because we have 0 as the coefficients. So, the polynomial becomes the polynomial notation. Number 57 can be represented as this (Refer Slide Time: 14:34) and your 83 can be represented as x to the power 7 plus x plus 1.

I need to do the addition operation among these polynomials. What I do is that I arrange them considering their degrees. Then, for example, this is called straight forward; (Refer Slide Time: 14:50) that you get x to the power7 because we do not have an x to the power 7 term here. You have x to the power 6 and you do not have an x to the power 6 term here. So, you get x to the power 6. Similarly, x to the power 4; there is no x to the power 4 here (Refer Slide Time: 15:00). So, you get x to the power 4. What about x to the power 2? There is no x power 2 also here. So, straight away you get x to the power 2 here.

Consider the variable x. There is an x here and there is an x here (Refer Slide Time: 15:13). When I am considering x, I am doing an addition over GF 2, which is the underlying field over the coefficients of x in both the terms that I am adding. Therefore, there is 1 here and there is a 1 here. If do an XOR between them, I get 0. So, x does not appear in the sum; similarly, for the constant term. So, finally, the result is x to the power 7 plus x to the power 6 plus x to the power 4 plus x to the power 2.

Note that the addition can be implemented with the simple bitwise XOR instruction. So, the addition in the field GF 2 to the power n can be implemented with the simple bitwise XOR instruction. So, addition is quite easy if you are able to remember that when you are adding terms, you also need to the modulo 2 operation. This may sometimes seem to be quite strange to us. So, for example, if I would like to compute the sum of say x plus 1 whole square, then that would result in?

(Refer Slide Time: 16:16)

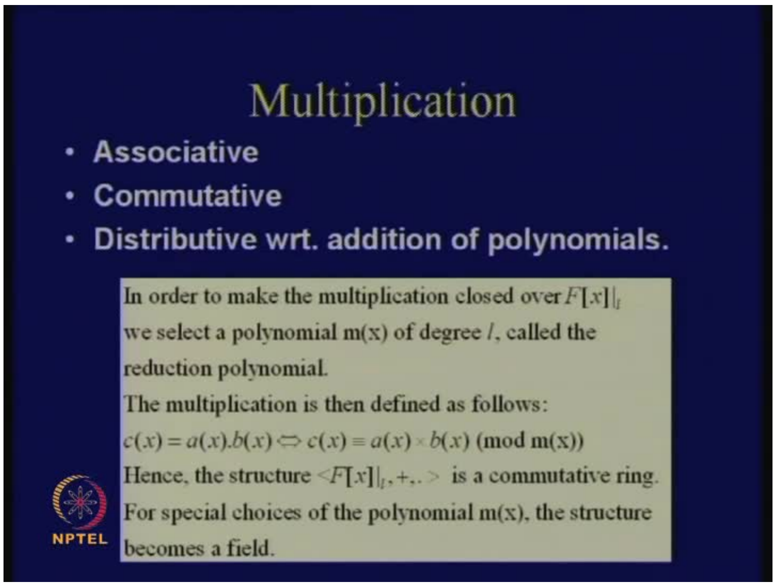


$\underline{GF(2^8)}$.

$$(x+1) \overset{+}{\oplus} (x) = 1. \quad \rightarrow \underline{\underset{0}{\cancel{2}}x+1}$$

Considering addition, GF 2 to the power 8; I would like to form the sums x plus 1 XORed with x. If this would have been a normal addition, then we would have got 2 x plus 1. However, when you are doing an addition over the GF 2 to the power 8 field, what you need to do is that do a simple addition, but over the coefficients, do a modulo 2 operation. So, if you do modulo 2 operation here, 2 gets reduced to 0 and you get back 1. Therefore, that is how you can easily obtain the corresponding sum.

(Refer Slide Time: 17:04)



Multiplication

- Associative
- Commutative
- Distributive wrt. addition of polynomials.

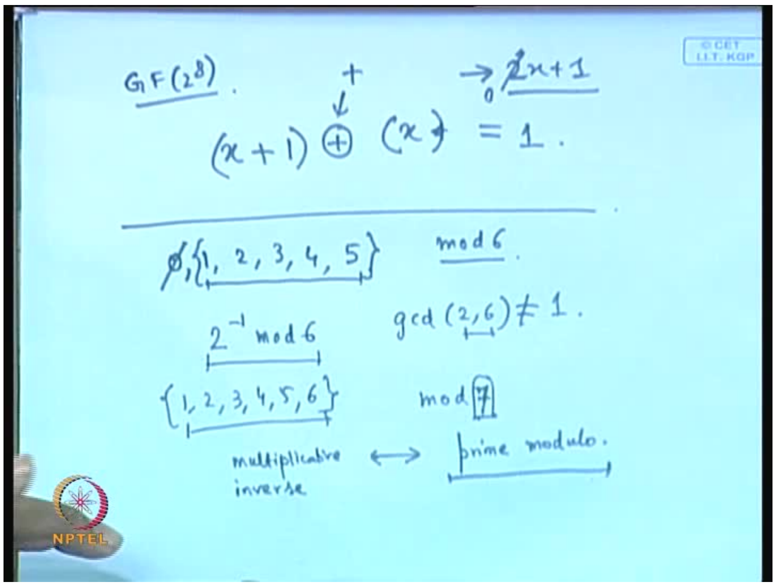
In order to make the multiplication closed over $F[x]|_l$ we select a polynomial m(x) of degree l, called the reduction polynomial.
The multiplication is then defined as follows:
$c(x) = a(x).b(x) \Leftrightarrow c(x) \equiv a(x) \times b(x)$ (mod m(x))
Hence, the structure $\langle F[x]|_l, +, . \rangle$ is a commutative ring.
For special choices of the polynomial m(x), the structure becomes a field.

However, multiplication is not so straight forward. In case of multiplication, the properties of associativity and commutativity are quite easily understood; also, distributivity with respect to the addition of polynomials. So, I think we all of us know these points. These things are quite self-evident, but what about multiplication? The difference of multiplication from addition is that in case of multiplication, there is a problem of over flow. When you are multiplying say for example, 2 n n degree polynomials, the resultant output, that is, the product has got a degree of more than n. Therefore, if the field is restricted, then there is a problem of over flow, which means what?

It means that in order to impose the property of closure, we require to do a modulo operation. Therefore, in order to do a modulo operation, we introduce the concept of a reduction polynomial and that is denoted by m x, whose degree is l. I call that a reduction polynomial. That means when I am doing a multiplication, I define my multiplication as follows. I take a x and b x. If I want to do a multiplication and obtain c x, it implies that I do a normal multiplication with a x and b x; a normal polynomial multiplication, but I do a modulo with m x. The idea is that it is required to bring the corresponding product back to the field. So, if you are able to do that, then F x l or F x suffix l under the operation of both plus and dot, is a commutative ring. So, what you have done is that you have imposed the property of closure and obtained what is known as a commutative ring. However, for special choices of the polynomial, this m x (Refer Slide Time: 18:56) leads to the structure of a field.

What is that property for a polynomial m x? We need a property, which is known as irreducibility. So, what is the extra criterion that we require when we are talking about say field? We require the idea of a multiplicative inverse.

(Refer Slide Time: 19:43)



GF(2^8).

$(x+1) \oplus (x) = 1$. $\rightarrow \underset{0}{\cancel{2}}x+1$

$\cancel{0}, \{1, 2, 3, 4, 5\}$ mod 6.

2^{-1} mod 6 $\gcd(2,6) \neq 1$.

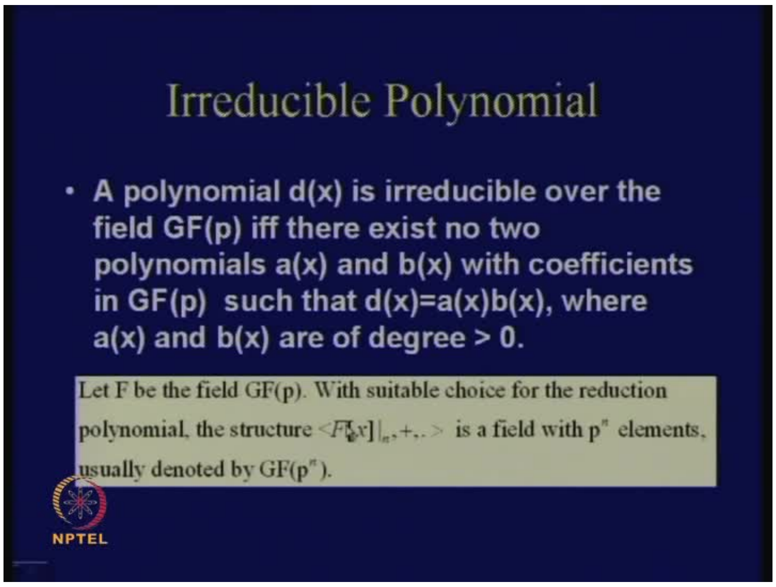
$\{1, 2, 3, 4, 5, 6\}$ mod $\boxed{7}$

multiplicative inverse $\longleftrightarrow$ prime modulo.

So, the moment we require something, which is called multiplicative inverse, if you remember the normal number theory, then we had the result that number had to be a prime number. What I mean to say is that if you consider the numbers from say 0, 1, 2, 3, 4, 5; if I exclude 0 and take only the elements say 1, 2, 3, 4 and 5, then I am doing a modulo 6 operation. So, you see that in this case, all the elements that we have already discussed in our previous class do not have a multiplicative inverse. For example, 2 does not have a multiplicative inverse; that is, 2 inverse modulo 6 is not defined. Why? Because the gcd of 2 and 6 is not equal to 1; that is, 2 and 6 are not co-prime to each other.

Similarly, if it is required to define this, then instead of taking modulo6, I would have rather taken modulo7. Why? Because 7 being a prime number, all the elements from 1, 2, 3, 4, 5, 6 are co-prime to the number 7. Therefore, the idea of prime modulo was important to define the concept of multiplicative inverse of all the elements. So, in order to have a multiplicative inverse for all the elements, we require that the modulo had to be a prime number. Similar concept exists for the polynomials also. In case of polynomials, we call the prime modulo by an irreducible polynomial.

(Refer Slide Time: 21:22)



Irreducible Polynomial

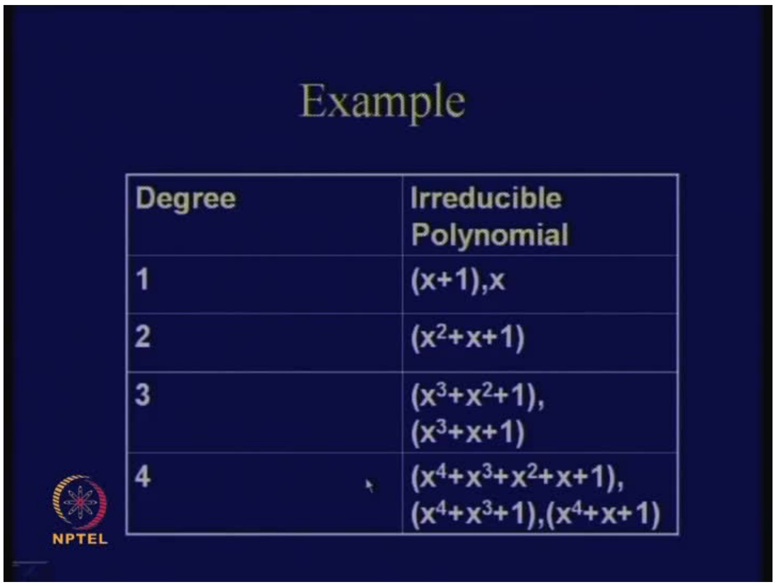
- A polynomial d(x) is irreducible over the field GF(p) iff there exist no two polynomials a(x) and b(x) with coefficients in GF(p) such that d(x)=a(x)b(x), where a(x) and b(x) are of degree > 0.

Let F be the field GF(p). With suitable choice for the reduction polynomial, the structure $<F[x]|_n, +, .>$ is a field with p^n elements, usually denoted by $GF(p^n)$.

Let us go through the definition of irreducible polynomial. It says that a polynomial d x is irreducible over the field GF p if and only if there exist no 2 polynomials a x and b x with coefficients in GF p such that d x is equal to the product of a x and b x. That means a x and b x form the factors of d x and their degree is greater than 0, but it is less than the degree that we are considering; For example, the degree of the field. Therefore, these elements, that is, ax and b x should not be members of the field. Therefore, there should not be any factor of d x, which is in the field. So, such kind of factors is not permissible. So, that forms the definition of an irreducible polynomial.

For example, let F be the field GF p. With suitable choice for the reduction polynomial, the structure F x suffix n under the operation of plus and product is a field with p to the power n elements. It is denoted by the letter GF p to the power n. So, in case of AES, p is equal to 2 and n is equal to 8. Therefore, we are talking about the field GF 2 to the power 8.

(Refer Slide Time: 22:41)



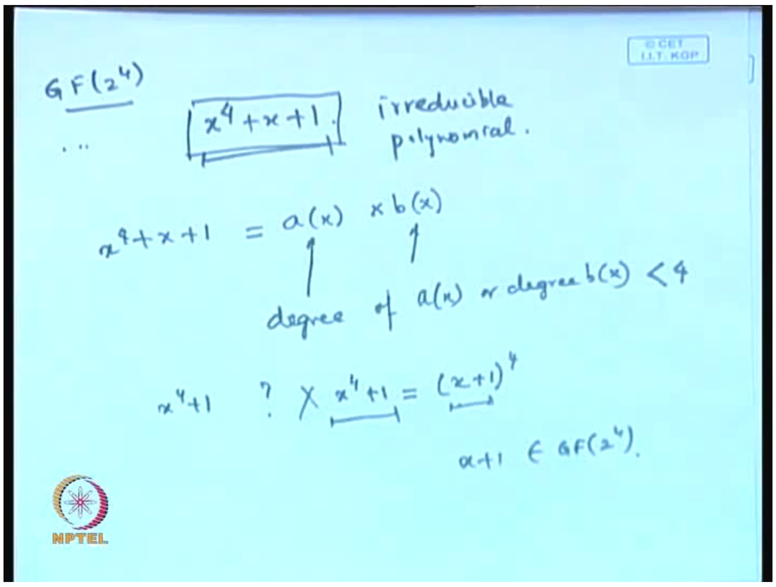
Example

Degree	Irreducible Polynomial
1	$(x+1), x$
2	(x^2+x+1)
3	(x^3+x^2+1), (x^3+x+1)
4	$(x^4+x^3+x^2+x+1)$, (x^4+x^3+1), (x^4+x+1)

There are some examples of irreducible polynomials. Consider degree 1. You see that x plus 1 and x are irreducible in this field because you do not have any factors of x plus 1 and x, which has inside this field.

For example, consider… I think this will be better explained if you take the degree 4. Here are some polynomials GF 2 to the power 4. If you consider GF 2 to the power 4 and take this particular polynomial - x to the power 4 plus x plus 1.

(Refer Slide Time: 23:22)



$\underline{GF(2^4)}$

$\boxed{x^4+x+1}$ irreducible polynomial.

$x^4+x+1 = a(x) \times b(x)$

degree of $a(x)$ or degree $b(x) < 4$

x^4+1 ? $\times$ $x^4+1 = (x+1)^4$

$x+1 \in GF(2^4)$.

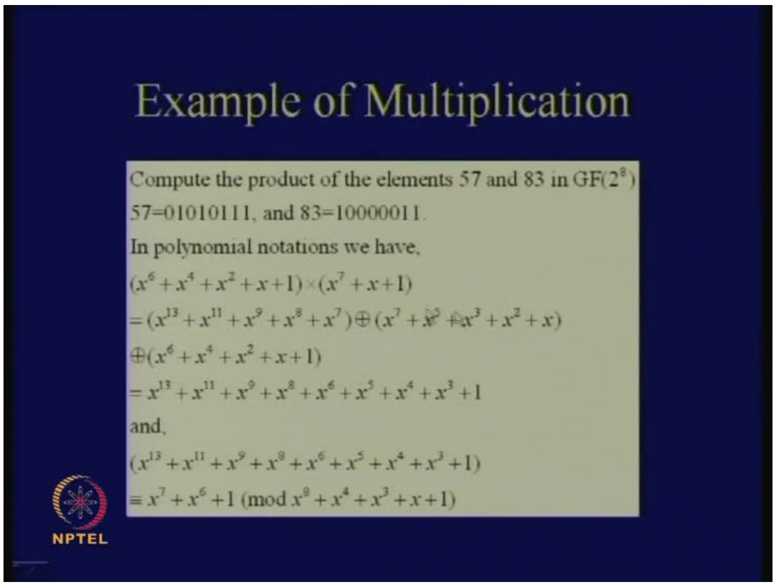
Therefore, if you consider the field GF 2 to the power 4 and consider a polynomial whose degree is 4; x to the power 4 plus x plus 1. If I say that this particular polynomial is an irreducible polynomial, then what we have to show is that x to the power 4 plus x plus 1, cannot be written in this form; a x into b x such that the degree of a x or the degree of b x is less than 4. The degree of a x or the degree of b x should not be less than 4.

Consider another example like x to the power 4 plus 1. The question is - Is this an irreducible polynomial? For example, we can see that x to the power 4 plus 1. I can factor as x plus 1 whole to the power 4. Why? Because if I do a normal multiplication like x plus 1 whole to the power of 4 and apply a modulo2 operation over all the other coefficients, then we will see that only x to the power 4 plus 1 remains; whereas, the other coefficients are even numbers; Therefore, if I take a modulo 2 operation, all of them goes to 0. Therefore, this particular factorization holds in this field (Refer Slide Time: 24:49) GF 2 to the power 4. Therefore, this holds. Therefore, automatically we see that x to the power 4 plus 1 has got a factor x plus 1, which is a member of the field GF 2 to the power 4. That means x to the power 4 plus 1 is not an irreducible polynomial.

Therefore, if I want to define the idea of a field, that means, if I want all the elements except 0 has got a multiplicative inverse in GF 2 to the power of 4, then it is required to have reduction polynomial of this type (Refer Slide Time: 25:35); maybe this, but not this. This is not a good reduction polynomial if we would like to define the characteristic of a field.

Here are (Refer Slide Time: 25:47) some other examples, which you can see that they are really irreducible or not irreducible. You can take it as an exercise.

(Refer Slide Time: 25:55)



Example of Multiplication

Compute the product of the elements 57 and 83 in GF(2^8)

57=01010111, and 83=10000011.

In polynomial notations we have,

$(x^6+x^4+x^2+x+1)\times(x^7+x+1)$

$=(x^{13}+x^{11}+x^9+x^8+x^7)\oplus(x^7+x^5+x^3+x^2+x)$

$\oplus(x^6+x^4+x^2+x+1)$

$=x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1$

and,

$(x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1)$

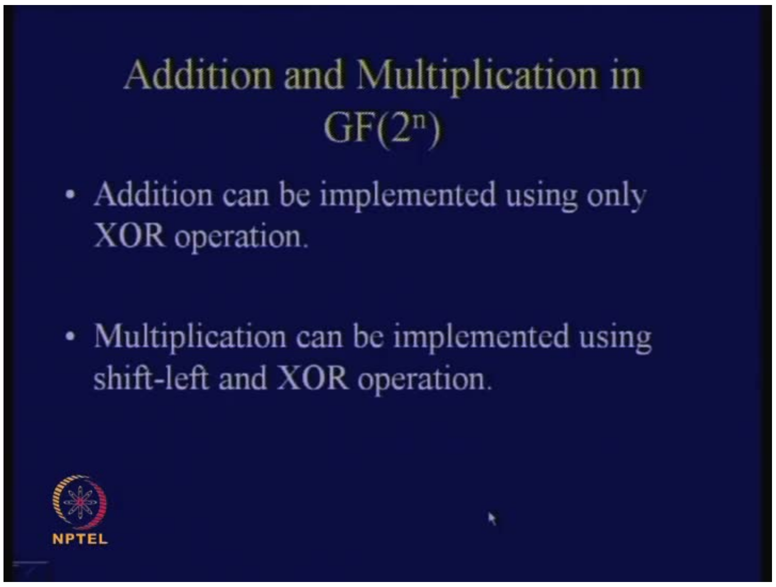
$\equiv x^7+x^6+1 \ (\mathrm{mod}\ x^8+x^4+x^3+x+1)$

Now, let us see how a multiplication works. If I want to compute the product and we take the same examples as we have done for the addition case; we take 57 and 83, which are members of GF 2 to the power 8. We have the same binary notation. I have already described how this polynomial works out and how this can be represented in this form.

Suppose I would like to compute the product of this polynomial and this polynomial (Refer Slide Time: 26:22). In order to do that, we just do a normal multiplication, that is, as I would have done with finite fields or without finite fields keeping in mind that addition is an XOR operation. If I do that, then this is the final polynomial that I get. The moment I so that my… I need to have a reduction polynomial because we see that the highest degree term in this case is x to the power 13; x to the power 13 does not belong to GF 2 to the power 8. So, in order to bring it back to GF 2 to the power 8, it is required to do a modulo with a reduction polynomial.

The reduction polynomial that we have considered here is x to the power 8 plus x to the power 4 plus x to the power 3 plus x plus 1. Why? Because this is an irreducible polynomial in this field (Refer Slide Time: 27:13). If we do a modulo with this field, then we get back x to the power 7 plus x to the power 6 plus 1. Therefore, we see that we have brought back the result back to the field GF 2 to the power 8. Actually, as a matter of fact that this is the polynomial, this is being used in AES or the description of AES as an irreducible polynomial.

(Refer Slide Time: 27:38)



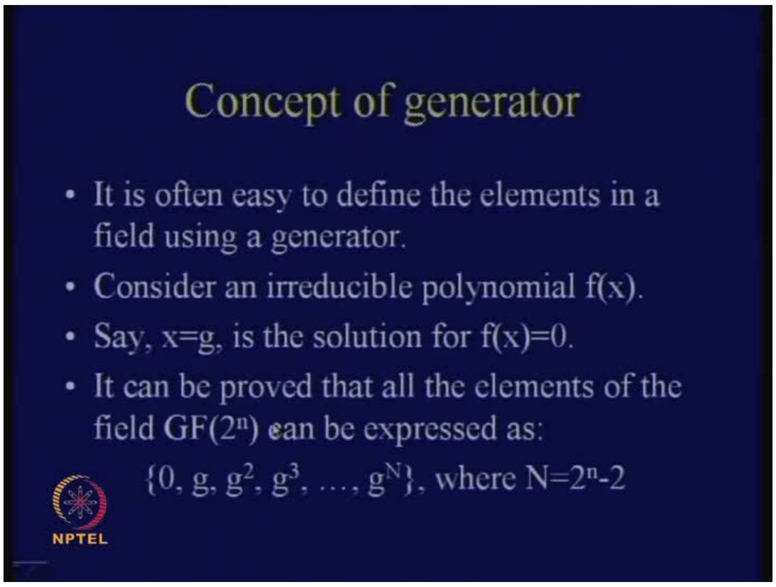
Addition and Multiplication in GF(2^n)

- Addition can be implemented using only XOR operation.
- Multiplication can be implemented using shift-left and XOR operation.

Commenting about the advantage of the addition and multiplication in GF 2 to the power n, we have already reflected upon the fact that addition can be implemented using only the XOR operation.

However, in case of multiplication, multiplication can be implemented, but you cannot do that only with the XOR operation. However, it is required to do another operation, which is the simple shift-left operation. So, in case of normal multiplication, you could have done with normal shift-left. However, since you are doing a finite field multiplication, it is required to do some extra XOR operation. This is because it is required to bring back the finite back to the field; it is required to do modulo operation.

(Refer Slide Time: 28:20)



Concept of generator

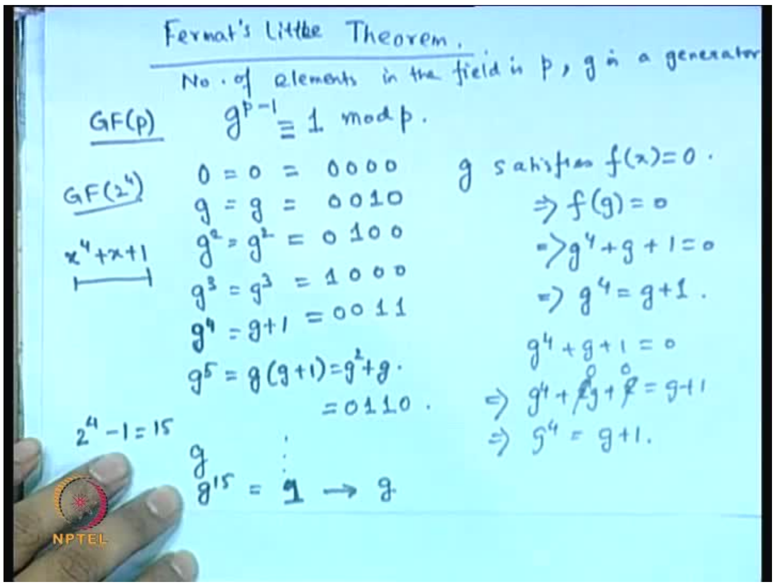
- It is often easy to define the elements in a field using a generator.
- Consider an irreducible polynomial f(x).
- Say, x=g, is the solution for f(x)=0.
- It can be proved that all the elements of the field GF(2^n) can be expressed as:

  $\{0, g, g^2, g^3, \ldots, g^N\}$, where $N=2^n-2$

We shall talk about a concept of something, which we call as generator. It is often easy to define the elements in a field using a generator. What is the idea of a generator? The idea of a generator or why the generator is required is as follows. That is, what we require is that all the elements, which are there in the finite field; I would like to generate them by using a single element. Therefore, I take a single element and I call them as g. Consider the various powers of g. Therefore, for example, I consider g I consider g square, g cube, g to the power of 4 and so on. I would like to generate all the elements in the finite field. Obviously, we understand the 0 cannot be generated. Therefore, all the nonzero elements can be derived in this fashion. So, that is the idea of generators. This particular element g is often referred to as the generator or the primitive element of the field.

In this case, if you consider the irreducible polynomials f x and say I need to compute the value of the generator g, then what I do is that I solve this equation f x equal to 0 and I obtain the generator. It can be actually proved that all the elements of the field GF 2 to the power n can be expressed in this form. Like, I take 0 and the others I generate by g, g square, g to the power 3 and so on till g to the power n; where n is equal to 2 to the power of small n minus 2. So, when I am considering a field of GF 2 to the power n, I have got the elements, which can be represented from 0 to 2 to the power of n minus 1. Therefore, 2 to the power of n elements exist. In our case, we know that we have run in till 2 to the power of n minus 2. Why?

(Refer Slide Time: 30:23)



Fermat's Little Theorem.

No. of elements in the field is p, g is a generator

GF(p) $\quad g^{p-1} \equiv 1 \bmod p.$

GF(2^4)

x^4+x+1

$0 = 0 = 0000$

$g = g = 0010$

$g^2 = g^2 = 0100$

$g^3 = g^3 = 1000$

$g^4 = g+1 = 0011$

$g^5 = g(g+1) = g^2+g. = 0110.$

$2^4-1=15$

g

$\vdots$

$g^{15} = 1 \rightarrow g.$

g satisfies $f(x)=0$.

$\Rightarrow f(g)=0$

$\Rightarrow g^4+g+1=0$

$\Rightarrow g^4=g+1.$

$g^4+g+1=0$

$\Rightarrow g^4+\cancel{g}+\cancel{1}=g+1$

$\Rightarrow g^4 = g+1.$

Because there is a theorem called Fermat's little theorem, which says that if you take an element g and if the number of elements in the field is p, then g to the power p minus 1 is equal to 1, when you are doing a modulo p operation. So, if you consider the field say GF p, all these elements and you take g, which is a generator; I call g as a generator, then g to the power p minus 1 is equal to 1, when you are doing a modulo p operation. That means if you consider all the elements for example, if you just take 0, 0 is 0; you take g equal to g; you consider g square is equal to g square; you take g to the power 3 is equal to g power 3. For example, if I consider the field GF 2 to the power 4 and if I consider the g to the power 4, we have a problem of overflow because g to the power 4 cannot fit in the field GF 2 to the power 4. So, it is required to do a modulo operation.

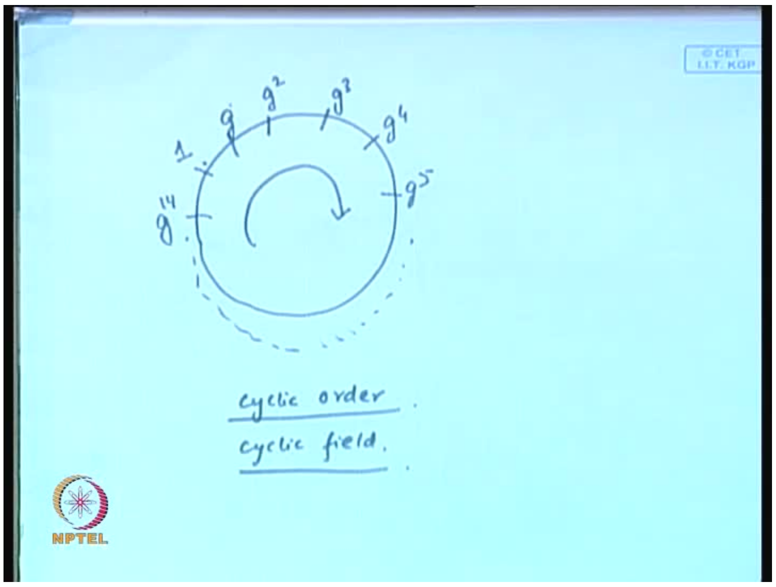
If I take say for example, x to the power 4 plus x plus 1 as the corresponding irreducible polynomial, we have already seen that. In that case, if we know that g satisfies f x equal to 0, then it implies that f g is equal to 0, which means that g to the power 4 plus g plus 1 is equal to 0. Therefore, it means that g to the power 4 is nothing but equal to g plus 1. Why is g to the power 4 is equal to g plus 1? Because the addition in case of GF 2 to the power n field is nothing but a subtraction operation; So, if g to the power 4… I would have transferred this on the other side. If I just take addition and subtraction to be the same, then I obtain this result (Refer Slide Time: 33:08).

Another way of observing this would have been this; that is, g to the power 4 plus g plus 1 is equal to 0. If I add g plus 1 to both sides, then I get g to the power 4 plus 2 g plus 2 equal to g plus 1. If I apply modulo 2 operation over this coefficients, then all of them go to 0. What I obtain is (Refer Slide Time: 33:28) g to the power 4 equal to g plus 1. That means that I could have represented this g to the power 4 (Refer Slide Time: 33:32) as g plus 1. Similarly, if I would like to compute g to the power 5, then g to the power 5 is g into g plus 1. So, that is equal to g square plus g. So, that is still inside the field.

Another way of seeing that it is really in the field is that I would have… We already discussed that in order to represent elements in GF 2 to the power 4, we can represent them in the form of polynomials. Therefore, also the corresponding binary representation would involve 4 bits. That means using a 4 bit representation I can represent all the polynomials in GF 2 to the power 4. Consider 0; It can be expressed as this (Refer Slide Time: 34:15). What about g? I can express this as 0 0 1 0. What about g square? I can represent them as 0 1 0 0; g cube - I can represent them as 1 0 0 0. However I cannot represent g to the power 4 in 4 bits. That means there is an overflow, but if I do this modulo operation, I obtain g plus 1. Therefore, that can be represented as this (Refer Slide Time: 34:37). What about g square plus g? I can represent g square plus g as 0 1 1 and 0. So, I can obtain the other elements also.

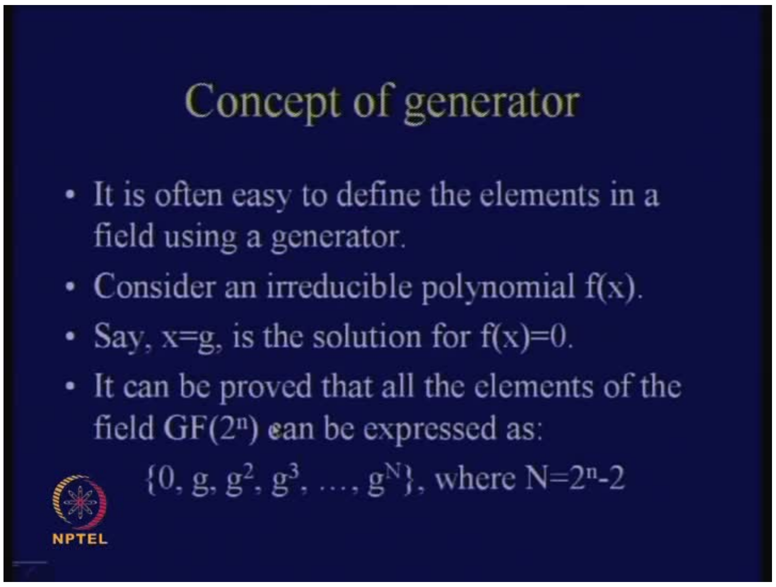
However, the point that I was saying from Fermat's little theorem is that if I consider 2 to the power 4 minus 1, then that is equal to 15. So, I keep on multiplying in this fashion, I will see that g to the power of 15 is actually equal to 1. So, that means if I would like to find out all the elements, then after this, (Refer Slide Time: 35:25) I have got a circular shift because what is the next element after this? I obtain g. This is because I again multiply this with g and I obtain back g.

(Refer Slide Time: 35:36)



That means all the elements in this form essentially form a cycle. Therefore, if I start with g and obtain g square; if I start with say…, then I obtain g to the power 3, g to the power 4, g to the power 5. Similarly, if I continue like this, there is a point after which g again reappear. So, there is 1 which appears at one place. There is g to the power 14; after which, there is a repetition. Therefore, there is a cycle; all these numbers can represent in the form of a cycle. This also called a cyclic order or cyclic field.

(Refer Slide Time: 36:25)



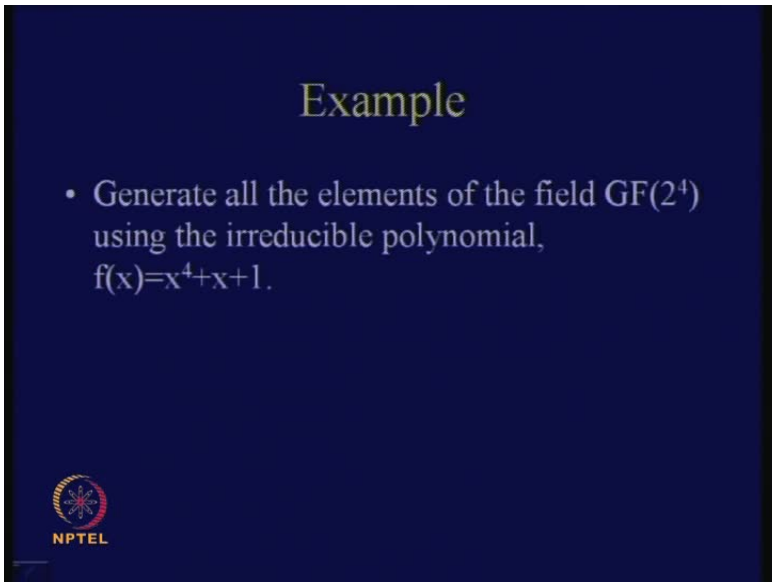
Concept of generator

- It is often easy to define the elements in a field using a generator.
- Consider an irreducible polynomial f(x).
- Say, x=g, is the solution for f(x)=0.
- It can be proved that all the elements of the field GF(2^n) can be expressed as:

  $\{0, g, g^2, g^3, \ldots, g^N\}$, where $N=2^n-2$

Therefore, if I would like to represent all the elements, then I can represent all the elements in this form – 0, g, g square, g to the power 3 and so on till g to the power n; where, n is equal to 2 to the power of n minus 2. If I consider the next element, then I obtain back g because that straight away follows from the Fermat's little theorem.

(Refer Slide Time: 36:47)

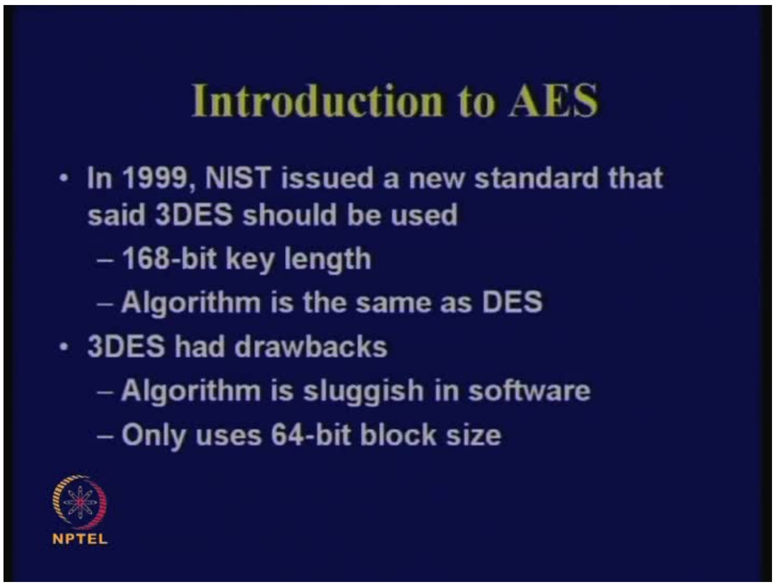


Example

- Generate all the elements of the field GF(2^4) using the irreducible polynomial, $f(x)=x^4+x+1$.

This could be one exercise, which I have solved already. Generate all the elements of the field in GF 2 to the power 4 using the irreducible polynomial; f x equal to x to the power 4 plus x plus 1. If I would like to generate all the elements of the field GF 2 to the power 4 using the irreducible polynomial, then these are the corresponding elements.

(Refer Slide Time: 37:16)

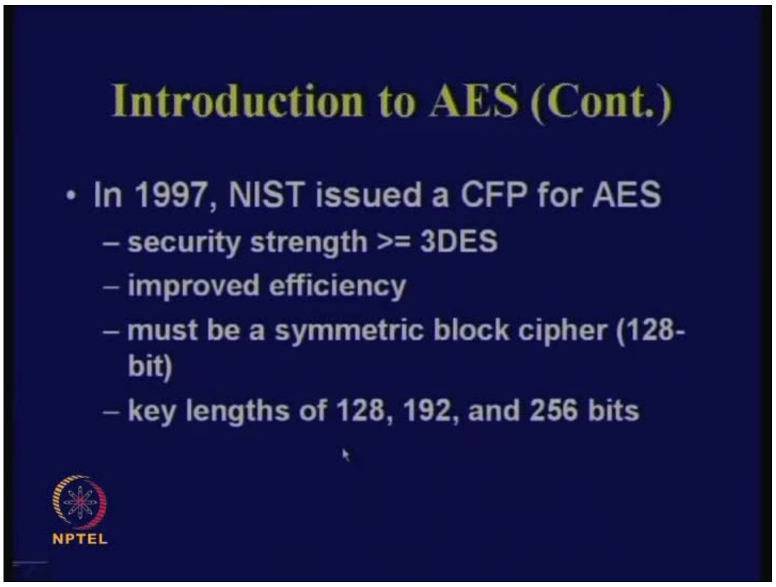


Introduction to AES

- In 1999, NIST issued a new standard that said 3DES should be used
  - 168-bit key length
  - Algorithm is the same as DES
- 3DES had drawbacks
  - Algorithm is sluggish in software
  - Only uses 64-bit block size

Next, we come to the introduction of the real AES algorithm or the Advanced Encryption Standard algorithm. In1999, NIST issued a new standard that said 3DES should be used. We have already seen that in case of 2DES, there were certain vulnerabilities and 2 DES was not able to provide the security of 112 bits of key because of the meet in the middle attack. Therefore, what it was thought is that 3DES will be adopted as a standard. Therefore, 3DES was supposed to provide a security of 168-bit key length and the algorithm was same as DES. However, there are some weaknesses. The weaknesses was that the algorithm was quite sluggish in software and at the same time, it uses only 64-bit block size.

(Refer Slide Time: 38:17)



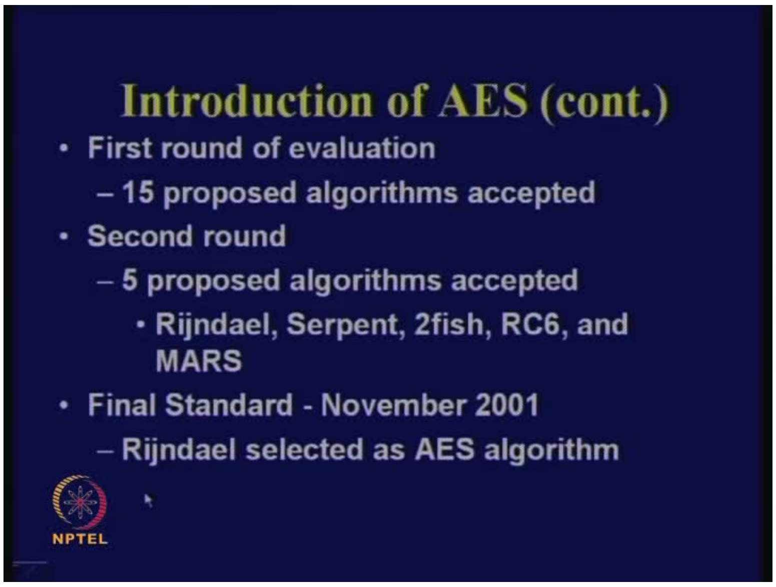
Introduction to AES (Cont.)

- In 1997, NIST issued a CFP for AES
  - security strength >= 3DES
  - improved efficiency
  - must be a symmetric block cipher (128-bit)
  - key lengths of 128, 192, and 256 bits

In case of DES, we have seen that there is something called weak keys and those keys still existed for 3DES. Therefore, 3DES was supposed to be substituted and NIST issued a call for paper for a new cipher. Therefore, people around the world thought that there was necessity of something, which is called an Advanced Encryption Standard.

The criteria for the AES algorithm or cipher, which will be adopted for AES, were as follows: the security strength should be equal to or greater than that of 3DES; the efficiency should be improved and it must be a symmetric block cipher. Therefore, the block length should be either 128 bits or more than 128 bits. Therefore, it was supposed to be a symmetric block cipher of 128 bits and the key lengths that needed to be supported were of 128 bits, 192 bits and 256 bits.

(Refer Slide Time: 39:11)

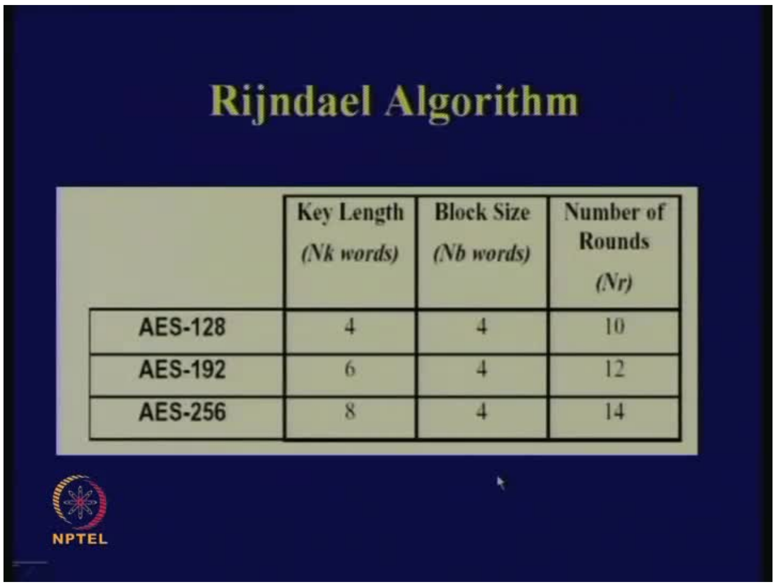


Introduction of AES (cont.)

- First round of evaluation
  - 15 proposed algorithms accepted
- Second round
  - 5 proposed algorithms accepted
    - Rijndael, Serpent, 2fish, RC6, and MARS
- Final Standard - November 2001
  - Rijndael selected as AES algorithm

Initially, in the first round of evaluation, around 15 proposed algorithms were accepted. In the second round, 5 proposed algorithms passed this round of selection and the 5 algorithm were Rijndael, Serpent, 2fish, RC6 and MARS. Finally, in November2001, Rijndael was selected as the AES algorithm.

(Refer Slide Time: 39:32)

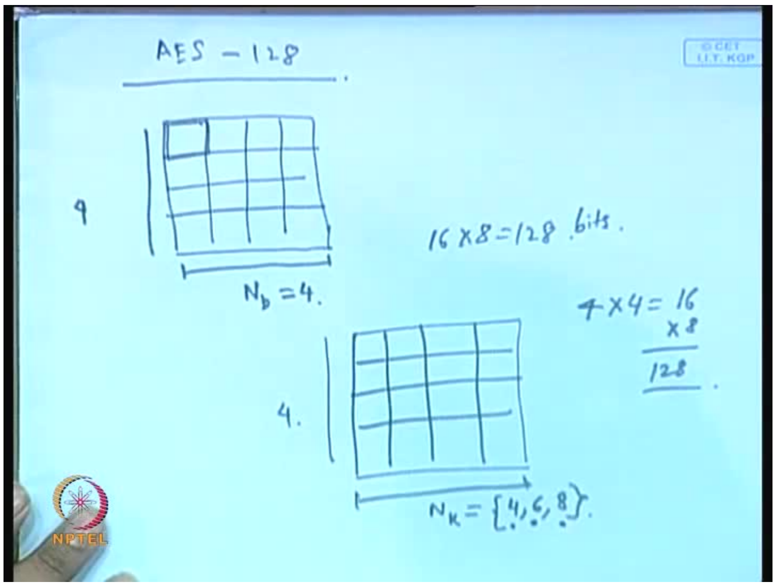


Rijndael Algorithm

	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

In case of AES algorithm, we will see that all the block lengths are taken to be of 128 bits. So, we have got AES-128, AES-192 and AES-256. The basic data of AES is often represented in the form of something which we called as a state matrix.

(Refer Slide Time: 39:59)

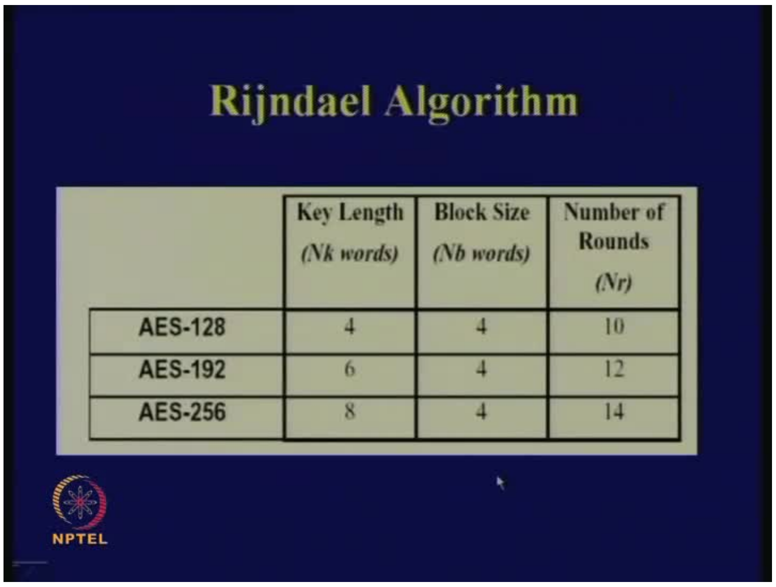


How does a state matrix looks is as follows. We can represent them in the form of this matrix. For example, let us consider AES-128. Each element of this state matrix is a byte. The number of rows is always equal to 4, but the number of columns varies. We say that the number of columns is represented by the variable N b. For AES-128, the value of N b is equal to 4, which means that we have got 16 bytes. So, 16 bytes means what? We have got 16 into 8 that is equal to 128 bits.

Therefore, the AES data block in case of AES-128 is equal to 128 bits. What about AES-192 and AES-256? In the case of AES-192 and AES-256, (Refer Slide Time: 40:59) the block size is equal to 4 and 4. This means that in all the three cases - AES-128, 192 and 256, the data word is actually equal to 128. Therefore, you are always processing a 128 bit of data; the block size is always 128 bits, but what about the keys? Note that the key in case of AES-128 is actually 128 bits. Here, (Refer Slide Time: 41:27) it is192 bits and here, it is 256 bits.

The key also can be represented analogously by another state matrix, (Refer Slide Time: 41:35) where the number of rows is still 4. You can again represent this in the form of a matrix and the number of columns is denoted by N k. N k is actually not only equal to 4, but it is equal to 4 comma 6 comma 8. So, in case of AES-128, it is 4; in case of AES-192, it is 6 and in case of AES-256, it is 8. That means the total size of the key is either equal to 128, that is, 4 into 4 is equal to 16. Therefore, we have got 16 bytes; means we have got 16 into 8. Therefore, total number of bit works out to 128 bits. However, in case of AES-192, the number of bits is 192 and in case of AES-256, it is 256 bits.

(Refer Slide Time: 40:30)



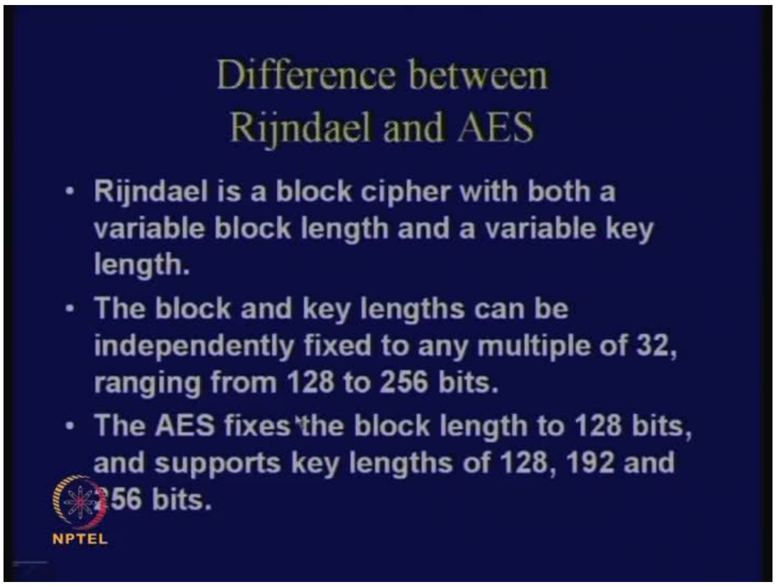
Rijndael Algorithm

	Key Length (*Nk words*)	Block Size (*Nb words*)	Number of Rounds (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Another point to be noted is that the number of rounds also varies; that is, in case of AES-128, the number of rounds is 10; in case of AES-192, the number of rounds is 12; in case of AES-256, the number of rounds is 14.

There are specific requirements by the number of rounds, were fixed like this. However, that follows from deep cryptanalytic results. We shall again discuss about this point in our future class.

(Refer Slide Time: 42:58)



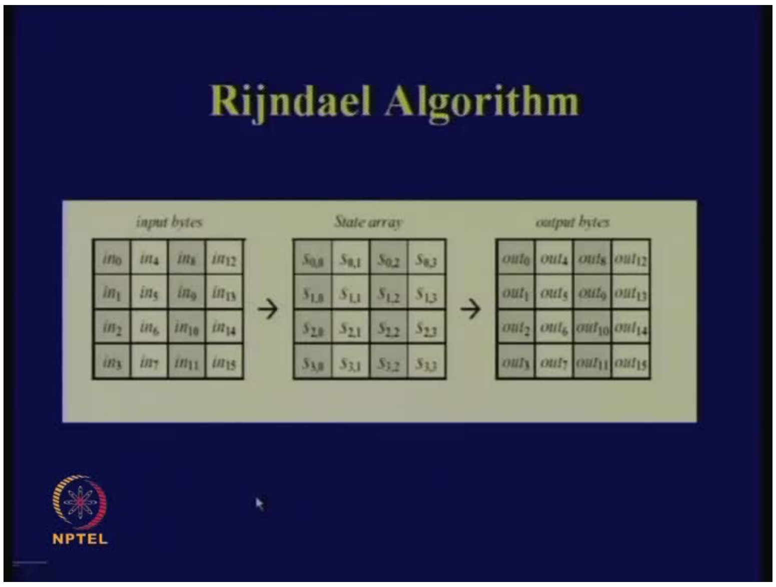
Difference between Rijndael and AES

- Rijndael is a block cipher with both a variable block length and a variable key length.
- The block and key lengths can be independently fixed to any multiple of 32, ranging from 128 to 256 bits.
- The AES fixes the block length to 128 bits, and supports key lengths of 128, 192 and 256 bits.

The difference between AES and Rijndael are not essentially the same. Rijndael is a block cipher, where both the block length as well as the key length can vary. However, when we talk about the AES structure, in case of AES, the block length is actually fixed to 128 bits, but the key length can vary. It can be independently varied from 128 to 256. Therefore, you see that in case of Rijndael, it is a block cipher with both the variable block length and the variable key length. Therefore, Rijndael is a block cipher, where the block and key lengths can be independently fixed to any multiples of 32 ranging from 128 to 256 bits.

However, in case of AES, the block length is fixed to 128 bits. Therefore, it supports key lengths of 128, 192 and 256 bits. That means that the difference between Rijndael and AES is that in case of Rijndael, you can also vary the block length, but in case of AES, the block length is fixed to 128 bits and the key is only 128, 192 and 256 bits. However, in case of Rijndael, the key lengths also could be fixed to any multiples of 32; ranging from 128 to 256 bits. So, you can intermediate values as well in case of the Rijndael algorithm. Rijndael is a more generalized cipher, but out of them, only those configurations, which we have tested for the AES computation from what we know as today’s standard or today’s Advanced Encryption Standard.

(Refer Slide Time: 44:46)



Rijndael Algorithm

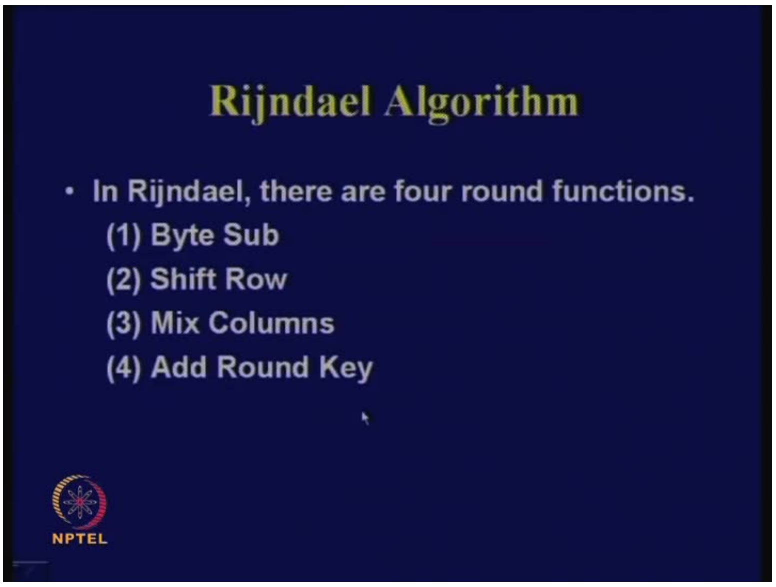
$$\begin{array}{c}\text{input bytes}\\ \begin{array}{|c|c|c|c|}\hline in_0 & in_4 & in_8 & in_{12}\\\hline in_1 & in_5 & in_9 & in_{13}\\\hline in_2 & in_6 & in_{10} & in_{14}\\\hline in_3 & in_7 & in_{11} & in_{15}\\\hline\end{array}\end{array} \rightarrow \begin{array}{c}\text{State array}\\ \begin{array}{|c|c|c|c|}\hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3}\\\hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3}\\\hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3}\\\hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3}\\\hline\end{array}\end{array} \rightarrow \begin{array}{c}\text{output bytes}\\ \begin{array}{|c|c|c|c|}\hline out_0 & out_4 & out_8 & out_{12}\\\hline out_1 & out_5 & out_9 & out_{13}\\\hline out_2 & out_6 & out_{10} & out_{14}\\\hline out_3 & out_7 & out_{11} & out_{15}\\\hline\end{array}\end{array}$$

This is how the Rijndael algorithm works. You have got your input bytes and you transform by certain algorithm to your output bytes. The intermediate steps you store in the form of something, which we refer to as the state array. So, you see that in this case, we are in fact talking about the AES and you have got 16 bytes. So, we denote them as in 0, in 1, in 2, in 3. So, you store them in the form of a through the columns. So, you have got in 0 to in 15 bytes. Similarly, you have got out 0 to out 15 bytes.

However, in between the corresponding operations that you do in the Rijndael algorithm transforms these input bytes to the output bytes, but the intermediate values are stored in the form of a state matrix. All the rounds of the Rijndael algorithm transfer these state matrices. Therefore, they perform operations on the state matrices. One of the prime reasons why Rijndael became the AES or rather the AES was won by the Rijndael designers was that this particular algorithm has got a nice byte-wise representation. The nice byte-wise representation led to good implementation features on even 8-bit processes.

We will discuss in our next class that in case of AES, we actually derived very good performance over various range of platforms; like 8-bit platforms, 16-bit platforms, 32-bit platforms, and even higher bit platforms. The AES algorithm works quite well; it derives good performance. One of the reasons is that the transformations can be exposed through byte-wise operations.

(Refer Slide Time: 46:46)

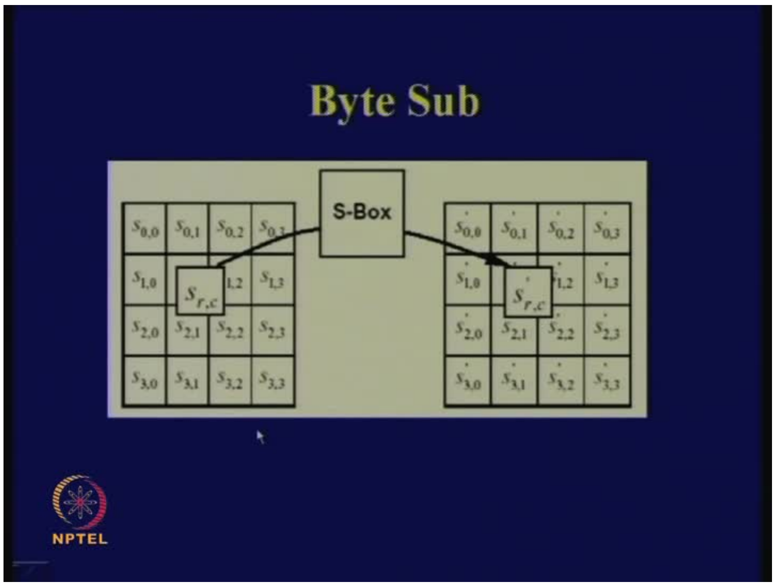


Rijndael Algorithm

- In Rijndael, there are four round functions.
  - (1) Byte Sub
  - (2) Shift Row
  - (3) Mix Columns
  - (4) Add Round Key

Looking inside the Rijndael algorithm, in Rijndael, each round has got four functions. So, there are four round functions. Let us discuss one by one.

(Refer Slide Time: 47:02)

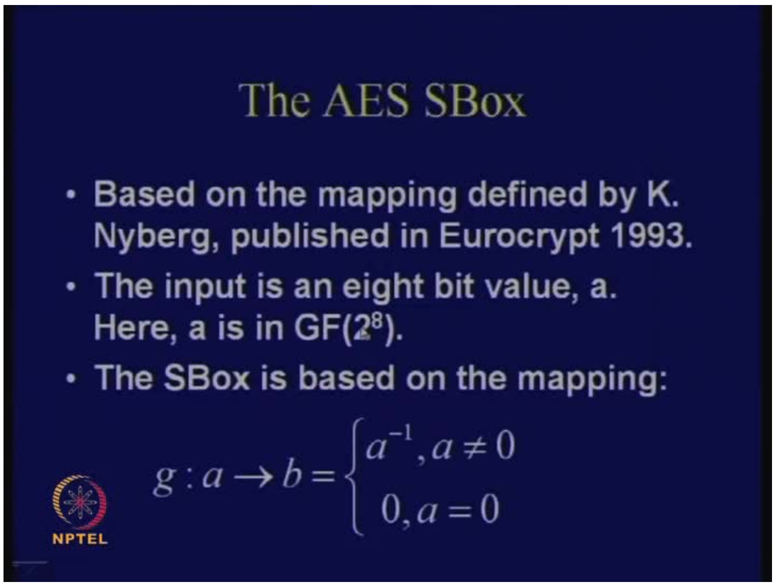


Byte Sub

S-Box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{r,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{r,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

First - byte sub; byte sub forms what is known as the S-Box. Therefore, this is the only non-linear step in the Rijndael-AES round function. What you do is that you take each byte and pass that through an S-Box. This S-Box is bijective transformation, which means it is a one-to-one map. It takes an 8-bit input and produces an 8-bit output. For example, it takes S r,c and converts that into another byte. So, it takes a byte and produces a byte. That is essentially exported to the output of this step.

(Refer Slide Time: 47:36)



The AES SBox

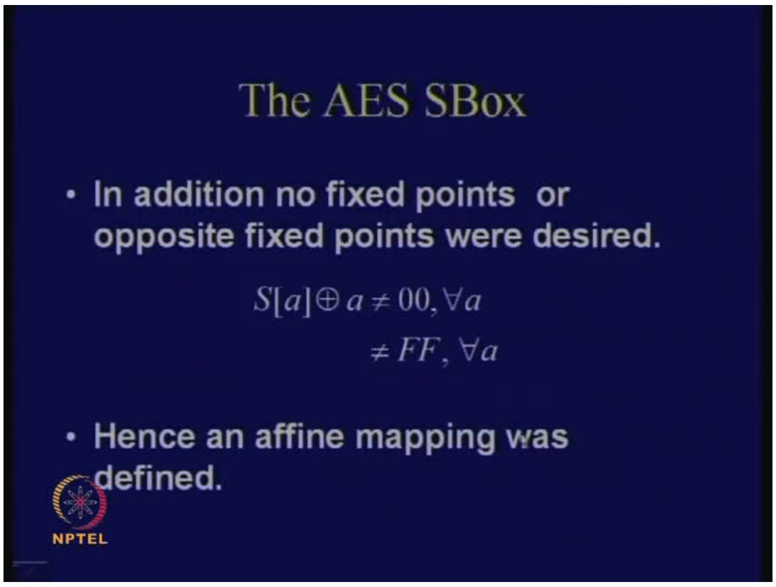
- Based on the mapping defined by K. Nyberg, published in Eurocrypt 1993.
- The input is an eight bit value, a. Here, a is in GF(2^8).
- The SBox is based on the mapping:

$$g : a \rightarrow b = \begin{cases} a^{-1}, a \neq 0 \\ 0, a = 0 \end{cases}$$

This byte sub is based on the mapping, which was defined by Nyberg and published in Eurocrypt in 1993. It is based upon a paper, which is called differential uniform mappings. Therefore, the input in this case is an 8-bit value, a. Here, a is in GF 2 to the power 8. The SBox is based on the mapping that if you take a and produce b. What you do is that if a is equal to 0, you take 0 as the output; that is, as we have already discussed that all the elements apart from 0 have got multiplicative inverse. This means that if you take a and a forms one of the elements, then the output is denoted by a inverse. However, if a is equal to 0 since inverse is not defined, in order to define at that point, we say that the output is also 0. So, we define that 0 inverse is 0. Therefore, through this definition, from a, you can obtain b. Therefore, this is a mapping that we have taken from a to b.

There are nice properties of this mapping, which we shall discuss in our cryptanalysis classes. However, this mapping has got a property, which is called a very uniform kind of nature. So, there is a very uniform transformation of these elements.

(Refer Slide Time: 48:53)



The AES SBox

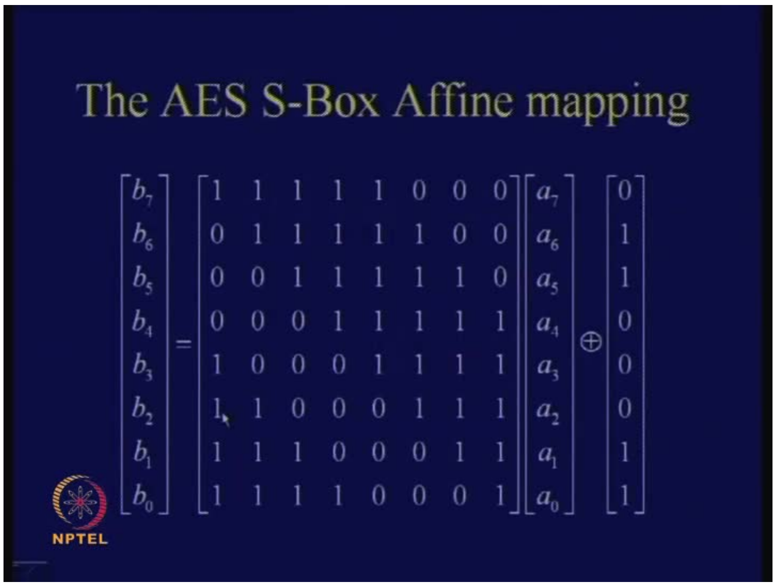
- In addition no fixed points or opposite fixed points were desired.

$$S[a] \oplus a \neq 00, \forall a$$
$$\neq FF, \forall a$$

- Hence an affine mapping was defined.

In addition, the designers also wanted that there should not be any fixed points or opposite fixed points. That means what was desired is that if you take a as the corresponding input, apply the Rijndael S-Box and then XOR that with a, then you should not get either all 0s or FF; that is, all 1s for all values of a. So, we call this as the fixed point and this as the opposite fixed points. In order to achieve this, an affine mapping was defined. So, all of us have seen what is an affine cipher. You will see that similar concepts appear in the case of modern cipher as well.

(Refer Slide Time: 49:35)

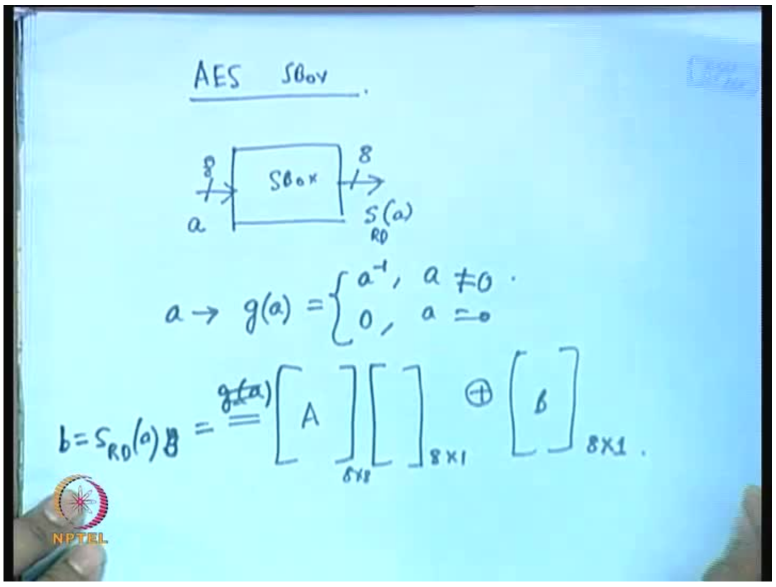


The AES S-Box Affine mapping

$$\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

How does the affine mapping look like? What it does is that it takes the corresponding output of the inverse function and passes that through a matrix, which I referred that as big matrix a and this is a vector called b. So, note that if you denote this in this form and if you observe this corresponding mapping, then you will see that each of these rows has got almost equal number of 0s and 1s. So, we can make this observation; whereas, the other rows are just cyclic permutations. So, you see that this particular matrix has got the property that all these rows are linearly independent and therefore, this matrix has got a full rank. Why is that required? Because we require the fact that this particular matrix should also have its multiplicative inverse. Because if its multiplicative inverse does not exist, then we cannot obtain back the vector a from the vector b. Where is that required? Because we require to decrypt what we encrypt.

(Refer Slide Time: 50:45)



AES SBox.

$$a \rightarrow g(a) = \begin{cases} a^{-1}, & a \neq 0 \\ 0, & a = 0 \end{cases}$$

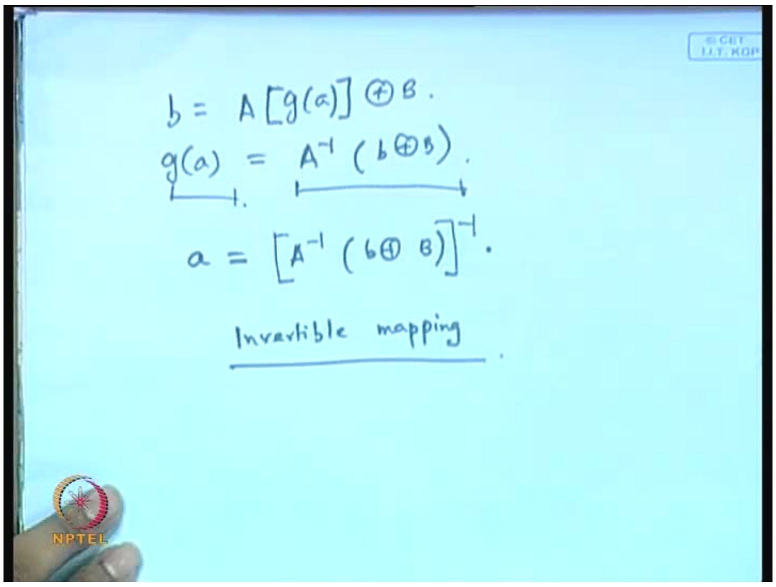
$$b = S_{RD}(a) = \overset{g(a)}{=} \begin{bmatrix} A \end{bmatrix}_{8\times 8} \begin{bmatrix} \; \end{bmatrix}_{8\times 1} \oplus \begin{bmatrix} b \end{bmatrix}_{8\times 1}.$$

If I combine these two steps, the AES S-Box looks like this. AES S-Box works on 8 bits and it produces another 8 bits of output. So, what you do is that for example, you take a and produce S a; I call that S RD a. How is S RD a obtained from a?

First, what you do is that you take a and pass it through mapping g. We have already defined how g a works; you represent that, obtain either a inverse if a is not equal to 0 or straight away pass 0 if a is equal to 0. Then, what you do is that you take this value of g a (Refer Slide Time: 51:29). Since this is also an 8 bit number because it belongs to GF 2 to the power of 8, I can represent this 8 bit number in the form of a vector of 8 rows and

1 column. Then, multiply that with the matrix, which I call as A. Therefore, what we do is that you multiply this with a matrix (Refer Slide Time: 51:49) of 8 cross 8 dimension and that is called A. Then, you add that to another vector, which you call as B and also, this has got a dimension of 8 cross 1. This output is essentially what we call as S RD a or for example, we call that as b.

(Refer Slide Time: 52:24)



$$b = A[g(a)] \oplus B.$$

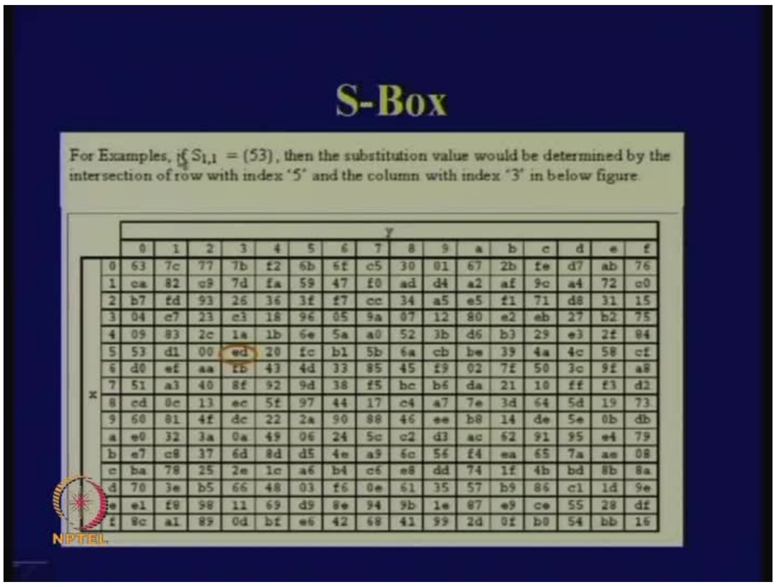
$$\underbrace{g(a)} = \underbrace{A^{-1}(b \oplus B)}.$$

$$a = [A^{-1}(b \oplus B)]^{-1}.$$

Invertible mapping

Now, given b, how can we obtain back a? We can easily see that we can obtain this because b equal to A g a XORed with B. So, if I want to obtain back a, what I do is that first of all I obtain g a. That I can easily obtain by multiplying A inverse with b XOR B. Since g a is defined by the function inverse of A in the finite field, then I can take the finite field inverse of this. That means I can take this; (Refer Slide Time: 52:56) we have this function g a, which is the self-invertible mapping. That is, if you give it a, it will give you back A inverse. So, it is a self-invertible mapping. Therefore, what I do is that I take the finite field inverse of this. If this value of A inverse b XOR B is equal to 0, then I report back 0; otherwise, I take the finite field inverse, I obtain back the output and I call that a. Therefore, we see that AES S-Box is actually an invertible mapping.

(Refer Slide Time: 53:34)



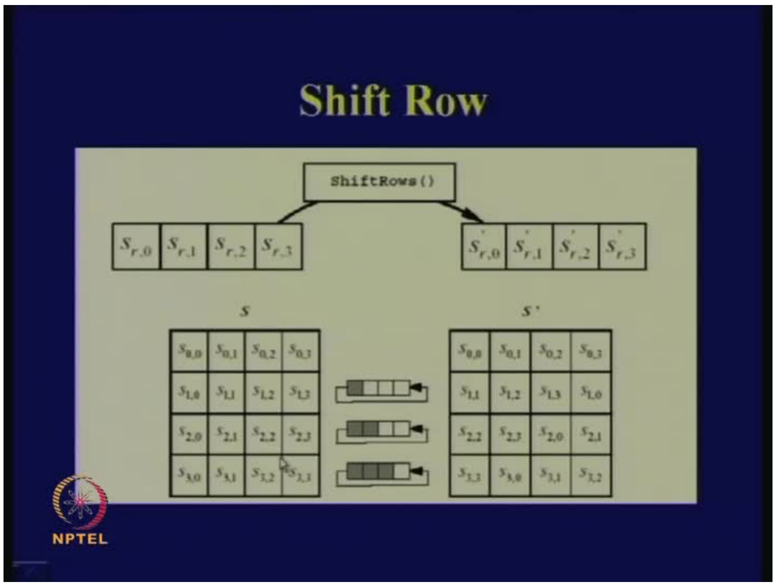
S-Box

For Examples, if $S_{1,1}$ = (53), then the substitution value would be determined by the intersection of row with index '5' and the column with index '3' in below figure

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

In tabular notation, this is how the AES S-Box works. Therefore, for example, if I would like to find out or look up the value of output of 53, then I take the fifth row and the third column and I report the element ed. Therefore, if the input is 53, the output of the S-Box is ed. So, all these elements are denoted in hexadecimal fashion.

(Refer Slide Time: 53:53)

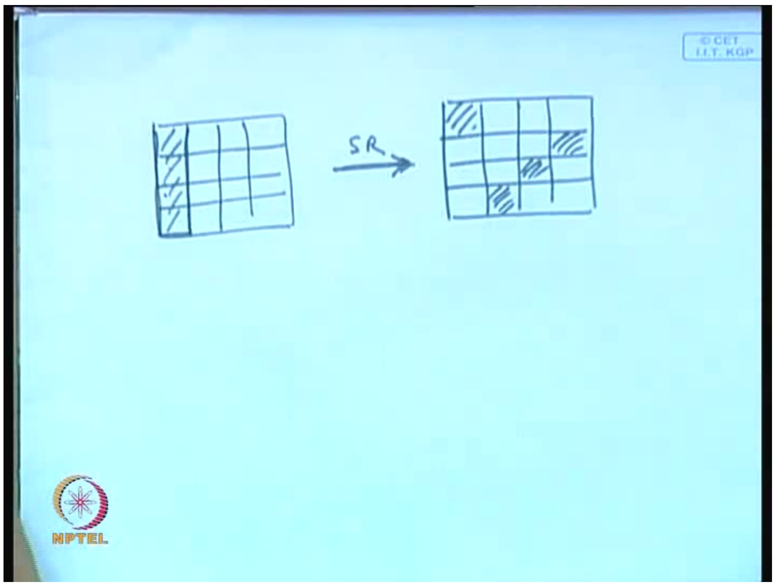


Shift Row

ShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
s
s'
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Shift row is a very simple operation. So, what you do is that you have got four rows: you do not shift the first row; you make a circular shift by one step in the second row; you shift by three steps in the third row and you shift them to the left. Therefore, you see that in this case, the advantage of having such a shift row operation is that we will shall see in our… If you observe the diffusion of AES also, the fact why the shift row is used is that it tries to diffuse a disturbance in one of the bytes to as many columns as possible.

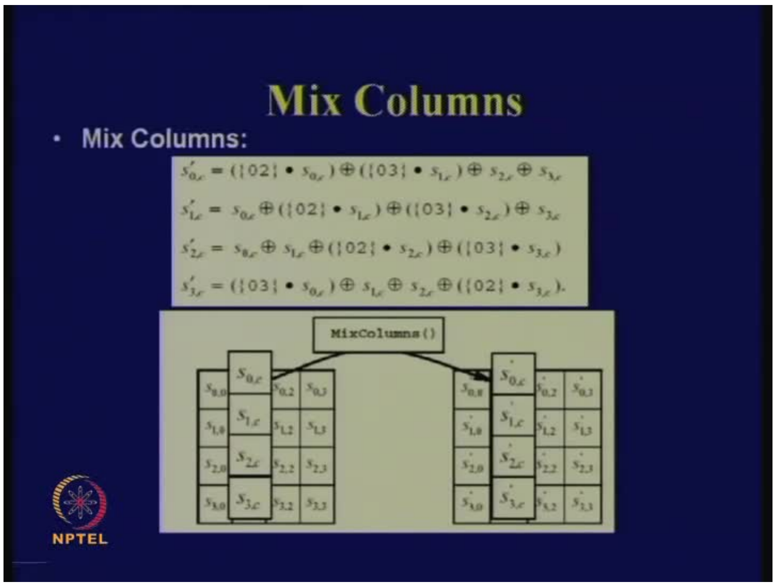
(Refer Slide Time: 54:40)



For example, what you do is this, that is, you can observe this fact from this fact. That is, if you take the AES basic state matrix and disturb one of the bytes and pass it through the shift row operation, then you can see that the first byte does not get shifted. However, the second byte gets shifted to this point; this one gets shifted here (Refer Slide Time: 55:01) and this one gets shifted here.

Now, you see that you had disturbance in one of the columns, but because of the shift row, you have disturbance in all the columns.

(Refer Slide Time: 55:14)



Mix Columns

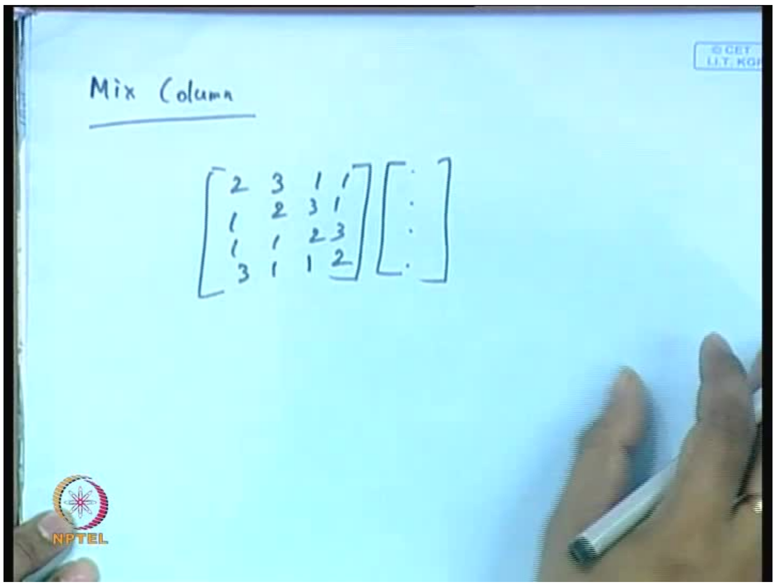
- Mix Columns:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

MixColumns()

That is again taken up by another step, which comes next. It is called the mix columns step. In case of mix columns, what you do is that you take column and transform this by a matrix multiplication and obtain another column. This is how the relation works like. These are the specific things like – if I would like to obtain the value of say S 0 c dash, then what I do is that I multiply the corresponding element of S 0 c with 2; I multiply with 3S 1 c; I take S2 c and S3 c, and XOR them. All these multiplications are in GF 2 to the power 8.

(Refer Slide Time: 55:50)

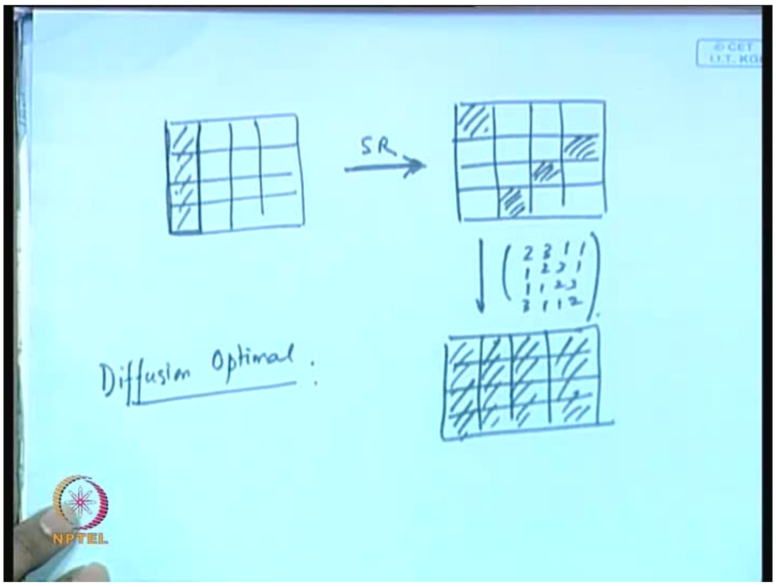


Mix Column

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

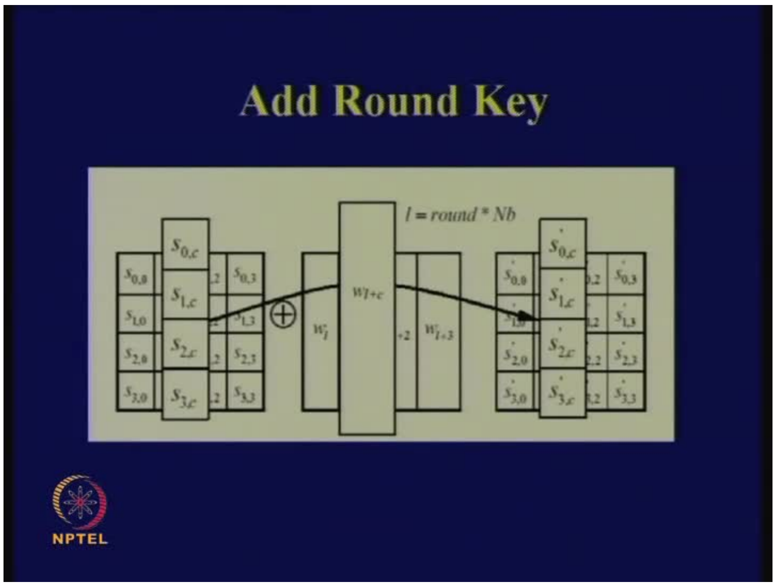
Another way of representing this mix columns operation is this. That is, in case of mix columns, what you do is you consider this matrix 2311, 2311, 2311 and 2311. What you do is that you take this corresponding matrix and multiply each of the column elements. This is how your multiplication works. Why is the mix columns step placed just after the shift row? The advantage comes from the fact that if you note this disturbance; how it propagates.

(Refer Slide Time: 56:23)



Now, if you multiply say any one of the columns by the corresponding matrix of 2311, 2311, 2311 and 2311; that is a mix columns matrix, then you will see that the disturbance, which was there in one byte spreads to the entire column. This particular thing spreads to the entire column. Similarly, the entire state matrix gets disturbed. Therefore, we refer to this combination as diffusion optimality. In a very few number of rounds, we can actually see that the entire state matrix of AES gets disturbed.

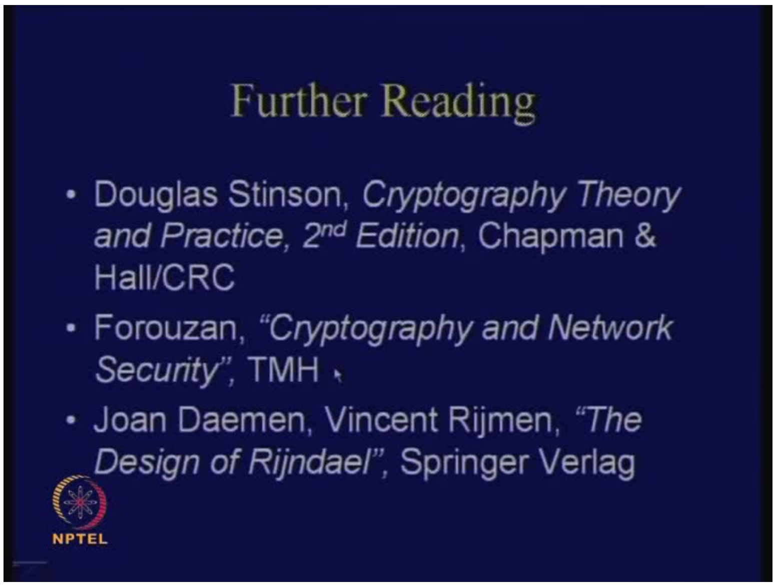
(Refer Slide Time: 57:07)



Add Round Key

The subsequent step is a very simple step, which is called the add round key. So, you take the key and you just XOR that with the state matrix.

(Refer Slide Time: 57:16)

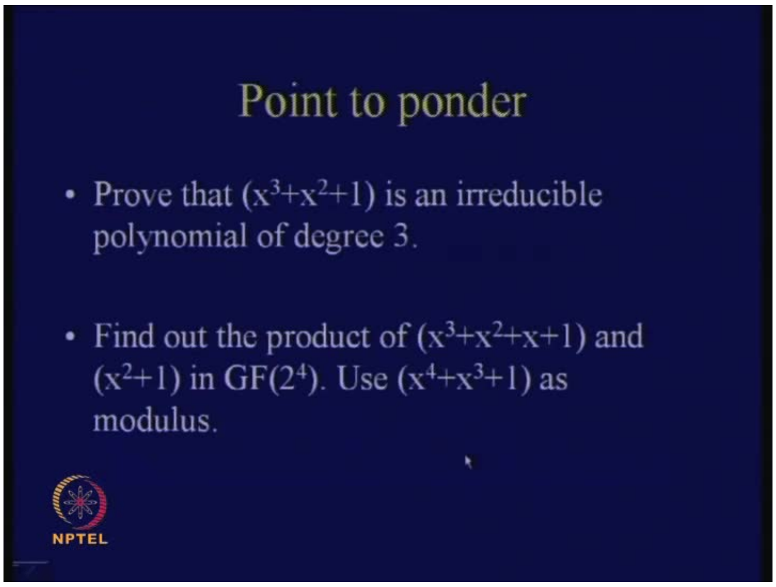


Further Reading

- Douglas Stinson, *Cryptography Theory and Practice*, *2nd Edition*, Chapman & Hall/CRC
- Forouzan, *"Cryptography and Network Security"*, TMH
- Joan Daemen, Vincent Rijmen, *"The Design of Rijndael"*, Springer Verlag

That concludes today’s parts. You can read further things from the book of Douglas Stinson and by Forouzan. Also, I have followed certain things from the original book - Design of Rijndeal written by the authors themselves, Joan Daemen and Vincent Rijmen.

(Refer Slide Time: 57:30)

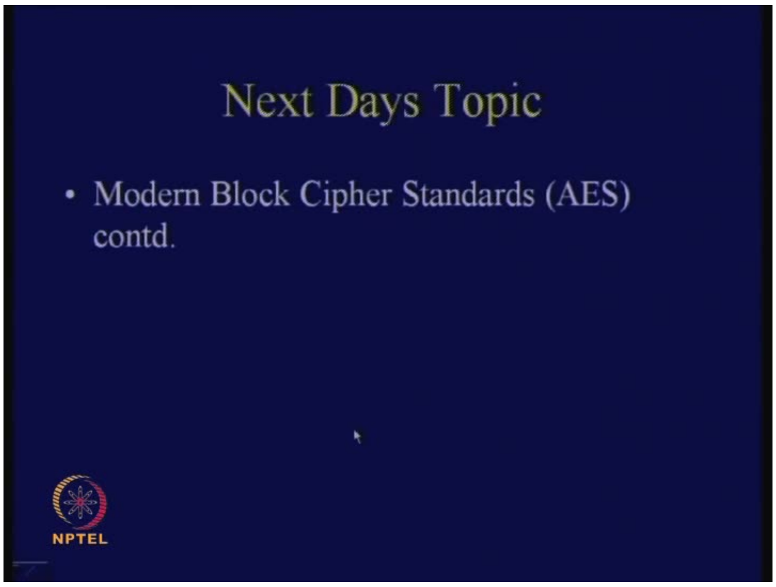


Point to ponder

- Prove that (x^3+x^2+1) is an irreducible polynomial of degree 3.
- Find out the product of (x^3+x^2+x+1) and (x^2+1) in $GF(2^4)$. Use (x^4+x^3+1) as modulus.

Here is an exercise for you; that is, you can prove that x to the power of 3 plus x to the power 2 plus 1 is actually an irreducible polynomial of degree 3. Also, you can work out this, that is, find out the product of multiplying this with polynomial with this polynomial (Refer Slide Time: 57:47) in GF 2 to the power of 4. We already discussed that in order to do a multiplication, we require a reduction polynomial; you can use this polynomial as the reduction polynomial.

(Refer Slide Time: 57:58)



Next Days Topic

- Modern Block Cipher Standards (AES) contd.

In next day’s class, we shall again continue with AES and discuss about certain other things, which is important for understanding this cipher.