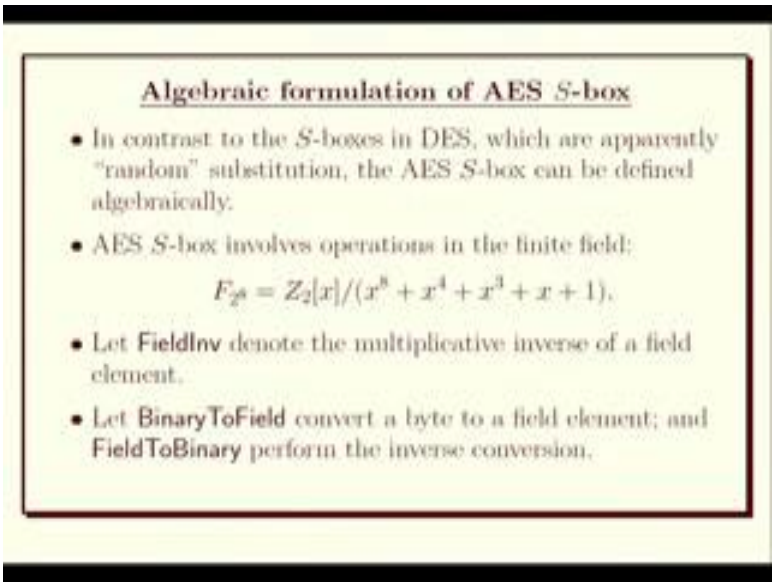
Internetwork Security
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture - 22
Advanced Encryption Standards (AES)
(Contd.)

We will continue AES, so we have seen the few operations; we have to perform on AES. So, we will talk about the algebraic formulation of AES S-box; sub byte operation. So, can you go to the slide please?

(Refer Slide Time: 00:39)



For DES there was no explanation of the S-box, how S-box is coming, but AES there is a proper algebraic structure for AES. So, there is a algorithm to find the AES S-box output, now this is based on finite field output operations.

(Refer Slide Time: 01:03)

We are taking a field which is F 2 to the power of 8 which is basically Z 2 x given by this irreducible polynomial x to the power 8 plus x to power 4 plus x cube plus x plus 1. So, this is a 8 degree irreducible polynomial. So, this Z to the power Z; z to x means it is set of all polynomial whose coefficient are 0 or 2. So, it is like a 0 plus a 1 x plus a 2 x square. So, this is basically Z 2 x dot, dot, dot, of any degree. So, a n x to the power n could be anything. So, where a i's are coming from Z 2, Z 2 is basically 0 and 1. So, this is a polynomial whose coefficients are basically binary 0 1.

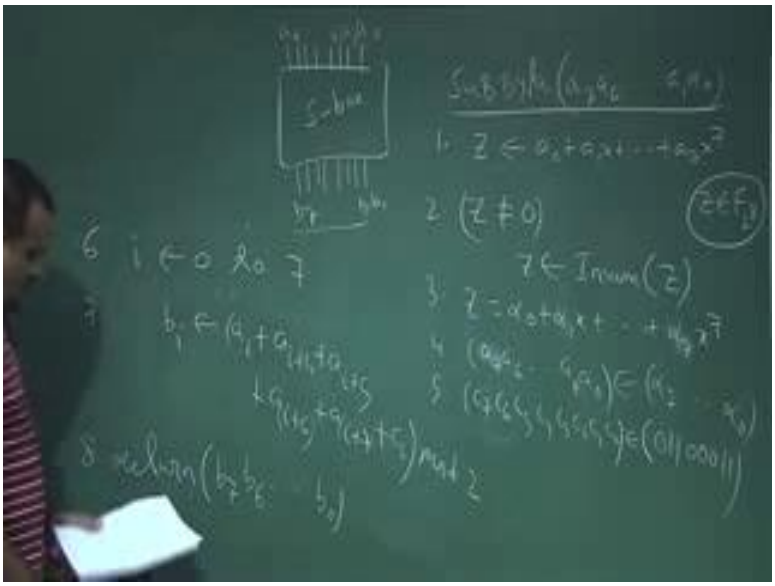
Now, if we take this polynomial P x which is irreducible polynomial; if we take this polynomial and if you take this mod operation I mean we take any polynomial in Z 2 and we try to multiply this with this polynomial then we will get a remainder polynomial and that remainder polynomial, so, is basically, so, if you take any polynomial a x; a x can be written as some b x into P x plus r x. So, this r x is basically of degree 7 because this is polynomial, we are trying to divide by degree 8.

So, the r x is degree 7 so that collection of this r x is called; it will form a field. So, it is degree 7 polynomial. So, it is basically some polynomial like a 1 a 0 a 1 x a 2 x square dot, dot, dot, a 7 x to the power of 7. So, all these collection and each of these a i's are 0 and 1 so that is why, so, there are 8; 8 coefficients. So, that is why it is size of this set is 2 to the power of 8. So, this is basically F, this is called Galois field, this is F 2, this is finite field. So, this is of size F 2 to the power of 8, so this collection is this set and this is

this will form a field. So, this is finite field or it called Galois field. So, this polynomial we will use for this field, we will use for our AES S-box.

If it is a field, we can perform the inverse operation in this. So, inverse exists for a given x. So, basically what we are doing?

(Refer Slide Time: 04:05)



We want to get the S-box for AES. So, this is AES S-box. So, it has 4 8 inputs. So, this we are writing by a 0, a 1, a 7, like this and the output we are denoting by say b 0, b 1, b 7. So, this, how we get b 0s from a 0s, so that we will discuss, so that is basically the AES S-box operation.

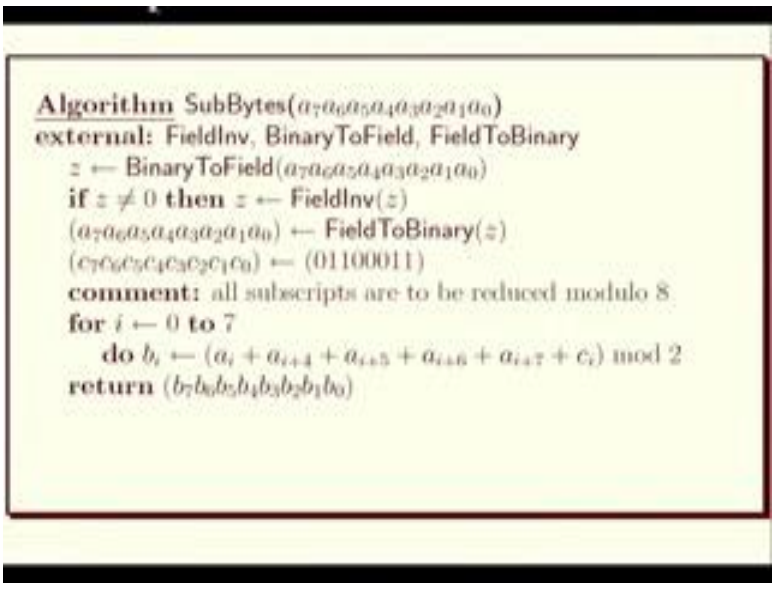
What we do? So, this can; you go to the slide please. So, this is basically the sub byte operation for AES. So, sub byte so, it is taking input say input a 0 and it should give us the output like b 0, b 2, and b 7. So, what we are doing? We are converting this into the a field so, how we can convert 8 bit number into field? So, just we will just denote this by a 1 plus a 0 plus a 1 x plus. So, this is basically a 0 plus a 1 x plus a 7 x to the power of 7. So, this is a field; field element Z, so, Z belongs to F 2 to the power of 7. So, this is a. So, Z is a field element.

Now this is a finite field. So, Z is having a inverse so that is basically so, if Z is not 0 then Z has a inverse. So, we denote that we store that inverse in Z inverse of Z. So, then we just again store into this inverse into this, again this is a field. So, this again we

convert into this is again basically some sort of a 0 plus a 1 x like this, this inverse is also a 7 degree polynomial not a 0, it must be some say alpha 0, alpha 1, alpha 7. So, basically what we have? We have this a 0 plus a 1 x plus a 7 x to the power of 7 into alpha 0 plus alpha 1 x, x to the power 7, this mod that irreducible polynomial should give us 1 then this is the inverse of this vice versa provided Z is not 0, it has the inverse. So, this is the inverse.

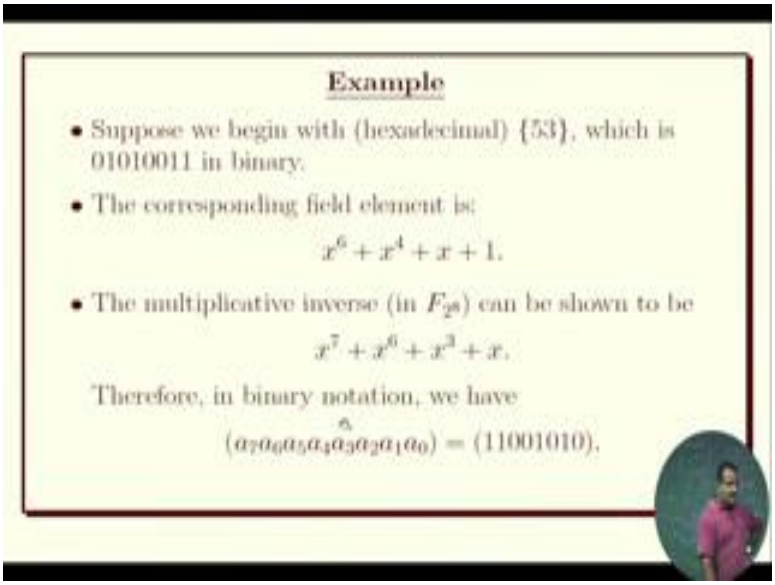
Now we store this alpha value here in a; so a 0, so, a 7, a 6, a 1, a 0, we store into the; so now, we have some constant. So, those constant are basically c; c value c 7; c 6, c 5, c 4, c 3, c 2, c 1, c 0, we have 7 constant and then this constant is given by the designer, so, 01100011, so, this is the constant given by the designer. So, this constant we are going to add with this a i's to get the this b i's, this is a fine transformation, we will come to that. So, now, how to get b i's then? So, this is 6, so from 0 to 7 so, basically these are b i's is basically we have this formula a i plus a i plus 4 plus a i plus 5 plus a i plus 6 plus a i plus 7 plus c i mod 2 because everything is mod 2 and this is also these are also has to be mod 8 because these indices have up to 7 0 to 7. So, this is the formula to get the b i's then finally, we return these b i, b 0. So, this is the algorithm this is the code for algorithm for this is the pseudo code for finding the b i's. So, can you go to the slide please?

(Refer Slide Time: 09:30)



This is basically the code we have discussed. So, we are taking the 7 bit which are basically a 0, a 1, a 7; a 7 so, this is 8 bit so, what we are doing? We are performing the binary to field. So, this is binary bit, it is field, we have basically a 0 then plus a 1 x. So, that just the corresponding filed element, now once we got the field element and if it is not 0 then it as a inverse. So, now, we store that inverse; the field inverse of Z into Z again and then we take the coefficient again in a and then we have this constant this constant is given by the designer and then we have this b i‘s operation to get the output of the S-box. So, we are basically on this each of this index must be mod 8 because we only have a 0 to a 7. And then finally this should be mod 2 because this b i’s should be 0 or 1 then we return this.

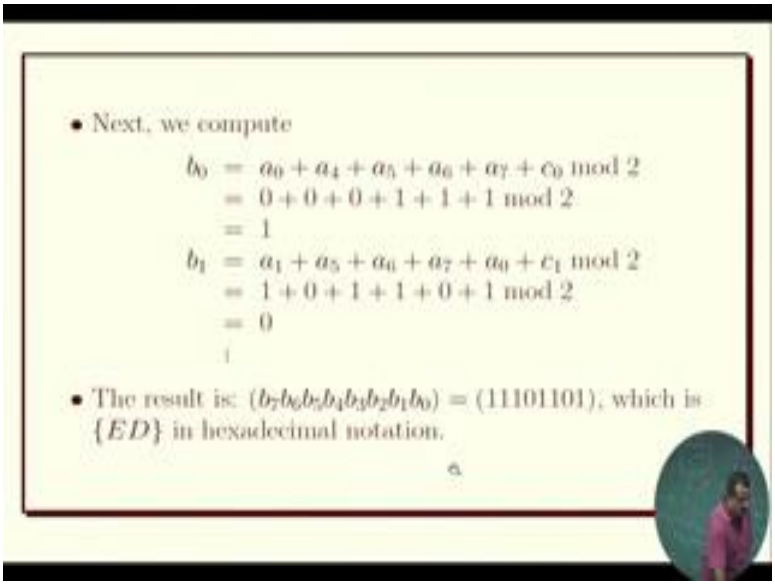
(Refer Slide Time: 10:35)



So, let us take an example. So, suppose we want to have the output for 5 3. So, this is the 8 bit. So, 8 bit can be represented in 4 bit 4 bit. So, this 5 is basically in hex form and 3 is also in hex form. So, 5 it is basically 0110 and 3 is basically 0010. So, this is our a i’s. So, now, if we convert into field element it is basically 1 plus x plus x square plus x to the power of 6. So, this is the field element corresponding to these binary bits now these we have to invert. So, this is non 0 elements in field in the Galois field. So, this as the inverse and we have to calculate the inverse and this is the inverse for this. So, you can check this by multiplying these 2 modulo that polynomial that irreducible polynomial and we can check this is 1. So, this is the inverse now we store this inverse. So, this is

also a this is again a 7 degree polynomial. So, these are 7 8 bits. So, this we will store into this again we will store into this a i's variables, so, a 0, a 1, a 2, a 7.

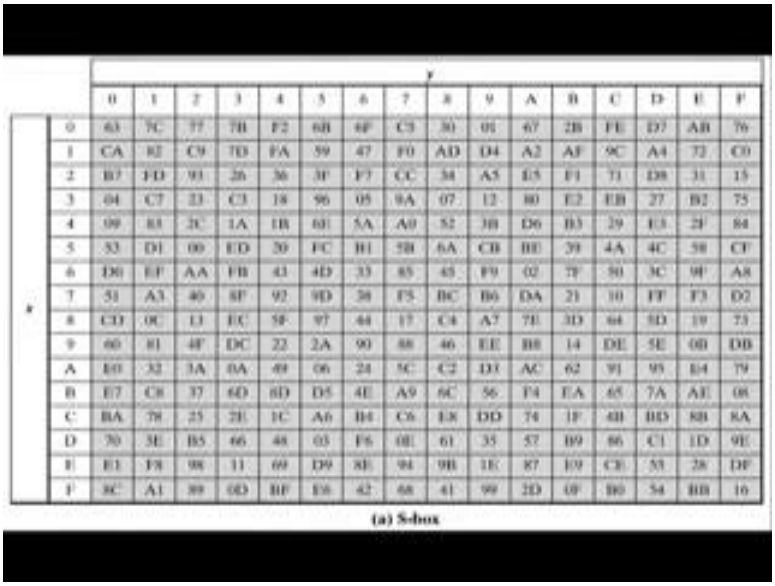
(Refer Slide Time: 11:55)



Now, to get the b i's, we have to get these c 0s constant and b 0 is basically a 0 plus this that formula we are going to use and put these values and we got 1.

Similarly, we can calculate b 1 b 2, so on, so, this is basically giving us this b i's are basically this which is in hex form e d. So, if we have input 5 3 then the output is e d. So, if this is this is x this is y if x is 5, y is 3 then the output is basically e d.

(Refer Slide Time: 12:43)



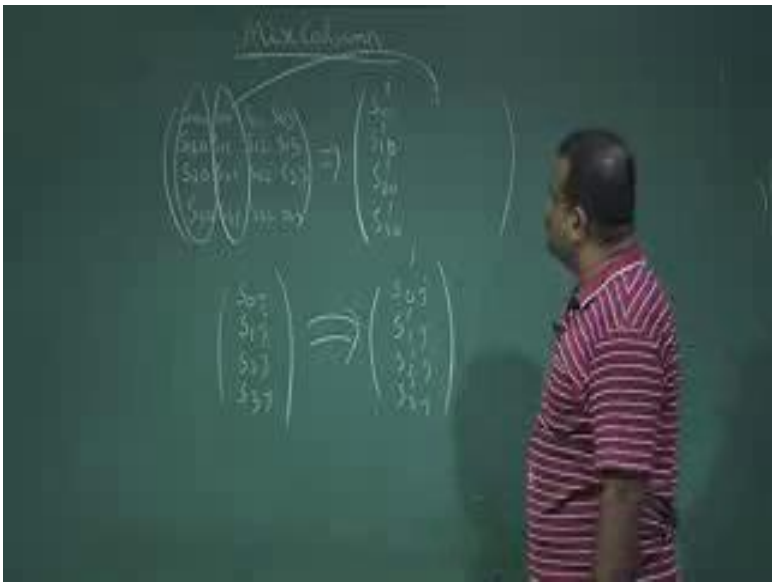
x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	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

(a) S-box

These was a we can verify form this table. So, if the input is x is 5 and y is 3 then we can check these output is e d. So, this is the table, but AES S-box is not just a table lookup.

We have proper mathematical algebraic construction for finding the value for AES S-box. So, after getting the value we can store into the table. So, that we can just do a table lookup if it is not expensive then computing this executing this algorithm then we will just look at the table and we go the corresponding position and we get the output, but it as a mathematical explanation for finding the AES S-box, now we will go for the how to get the mix? What is the mix column operation for AES? This is also involves field operation.

(Refer Slide Time: 13:57)



Mix column operation, so this is also field operation. So, it is basically; it is taking a column say S 0 j. So, AES operation is state. So, this is the state of AES 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 3 0, S 3 1, S 3 2, S 3 3. Now for mix column operation we are taking a column and we are converting into another column which is the output of this mix column operation S one. So, this is the 0th column S 3 0.

Similarly we take this column and we are converting into. So, if it J eth column J eth column means S 0 J, S 1 J, S 2 J, S 3 J. So, this is going to S 0 J prime, S 1 J prime, S 2 J prime, S 3 J prime. So, what are these elements? We will just come to know. So, this is the operation, now what is this primes or this new column, this we will come know. So, basically so, this is basically the field operation on this.

(Refer Slide Time: 15:54)

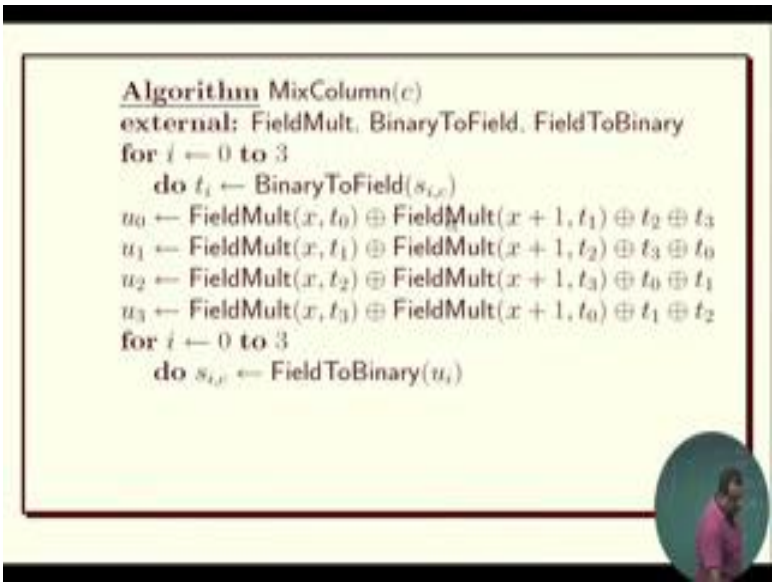
Basically what we are doing? This is S 0 J prime is basically 2 into 2 means x; x into in to means this is the field operation x into S 0 J x or with x plus 1 into x 1 J then x or with S 2 J x or with S 3 J. So, this is basically our first S 0 J prime.

Now, S 1 J prime is basically so, this is symmetry. So, S 0 J will be remain unchanged then we multiply x with S 1 J then x plus 1 with S 2 J and then S 3 J will remain unchanged. So, this is symmetric. So, S 2 J prime so, these 2 will be remained unchanged x or with S 2 J then we will perform this x into this into means field operation, these are all 8 bits. So, this can be converted into field like summation of a i x i then we can multiply this field operation under that mod that irreducible polynomial. So, this is basically S 2 J then x or with x plus 1 into S 3 J. So, then again the last 1 is so, this is x plus 1 into this is symmetric S 0 J x or with S 1 J x or with S 2 J x or with x into S 3 J. So, this is the formula, how we get this column mix column operation.

We have basically S 1, S 0 J, S 1 J, S 2 J, S 3 J, this is transferring to S 0 J prime, S 1 J prime, S 2 J prime, S 3 J prime and this is the operation we are doing. So, each operation is a field operation. So, this is a, this is x, this is a 8 bit. So, it can convert into say this is 8 bit. So, this is a; a 7, a 6, a 0. So, this can convert into a 0 plus a 1 x plus dot, dot, dot a 7 x to the power of 7. So, this we multiply with x field multiplication and field multiplication means under that Galois field F 2 to the power of 8. So, it has to be mod that irreducible polynomial.

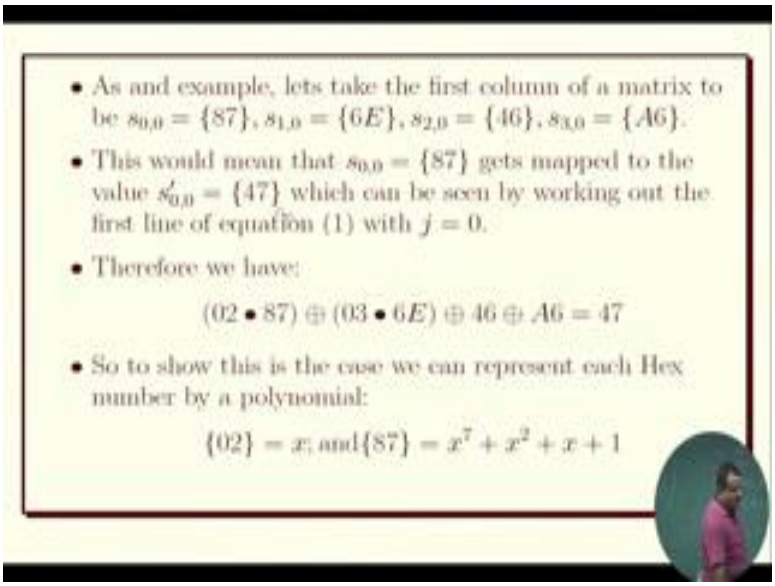
Again it is after this multiplication again it will be element in F 2 to the power of 8. So, again we can convert into that a 8 bit. So, this is also same. So, all are 8 bits. So, we are doing bitwise x or this is bitwise XORing once we got 8 bit; 8 bit we can do the bitwise XORing these are all 8 bits. So, like this. So, this is mixture of filed operation and the bit operation. So, bitwise XORing we are doing. So, this is the way how we do this mix column operation.

(Refer Slide Time: 19:58)



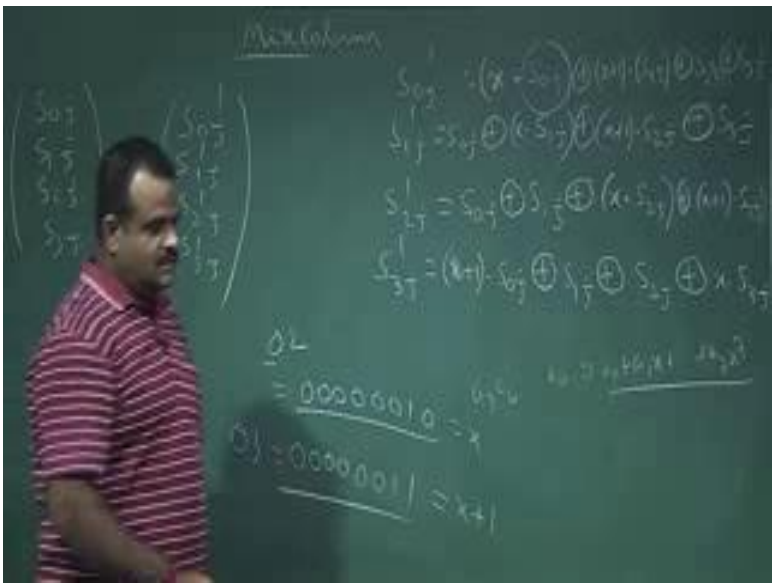
So this is basically our algorithm for mixed column operation, we are taking a column the c eth column then we are converting into this operation we are doing this field operation and then we are bitwise XORing and we are getting the new column.

(Refer Slide Time: 20:20)



This is we can take an example suppose our S 0, we had a column like this, this is the first element. So, each element is the 8 bit, this is the first element of the column, this is the second element, this is the third element. So, this is the 0th column, now we want to see where it is going. So, this is 0 into 2 means it is just x 0 into sorry, 0 2 means 0 2 means all are 0es this is in hex; hex notation.

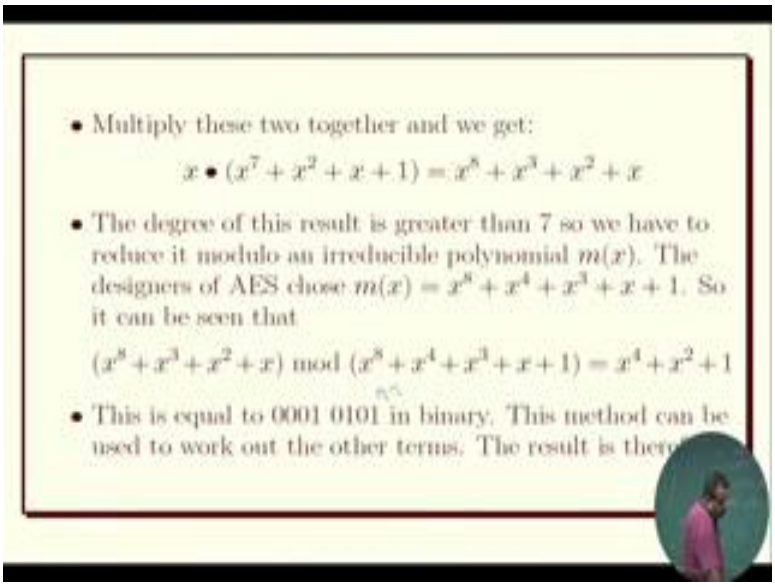
(Refer Slide Time: 20:51)



And 2 means 0010 so, this is basically x and 0 3 means or only 3 means all 0s; 0011. So, this is basically x plus 1 in this system.

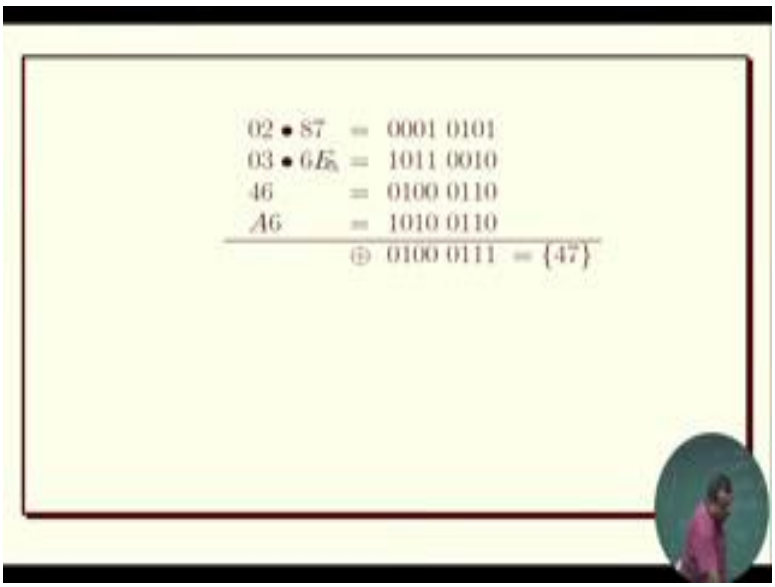
This is basically 2 into this is basically x into this, this is the field operation, we need to perform then we need to perform, this is x plus 1 basically into this and then these all are basically bitwise XORing. So, if you do that we will be getting 47 for the first column of this so, how we can check that so this is 87; 87 can be written as this.

(Refer Slide Time: 21:45)



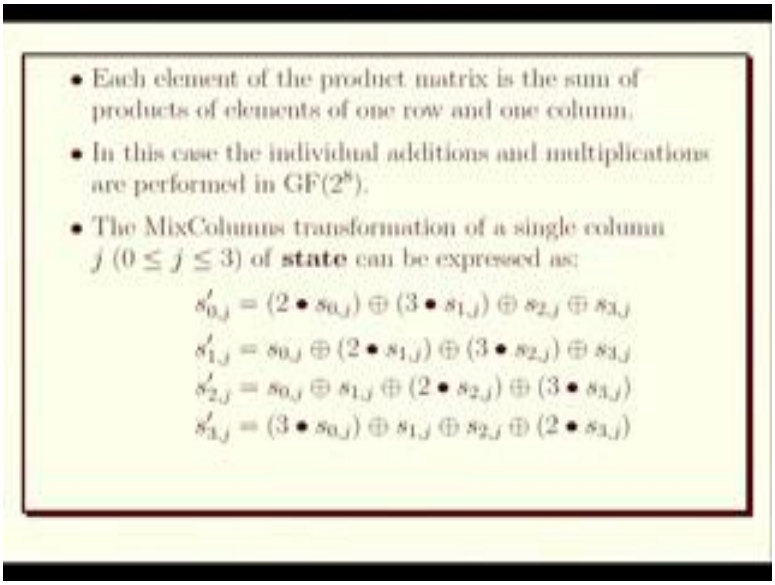
And this is x. So, if we multiply this we will be getting this now this is this as to be take mod because everything has to be in F 2 to the power of 8. So, if you take mod we are getting this; this field element and then this is equal to these binary bits.

(Refer Slide Time: 22:04)

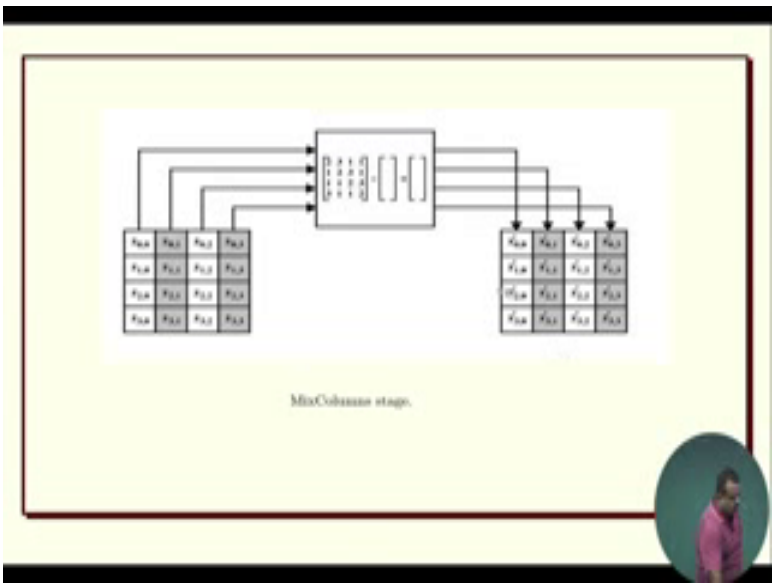


And similarly we can calculate this x plus 1 into that, this is also binary bit then we are doing the then we have this next 8 bit, next 8 bit we are just doing the bitwise XORing with this and we are getting 47, this is the first entry of that new column. So, this is the operation we are doing, this is the mix column operation. So, this is the mix column operation we are doing, can we go to slide please?

(Refer Slide Time: 22:51)



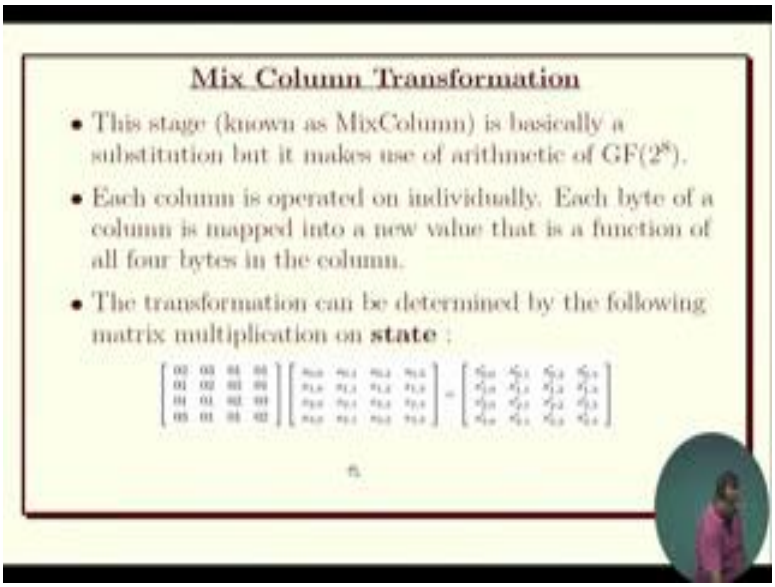
(Refer Slide Time: 22:57)



This is basically; slide please, so this is basically the mix column operation we are doing. So, if we just look at the matrix form. So, we are taking the particular column and we are

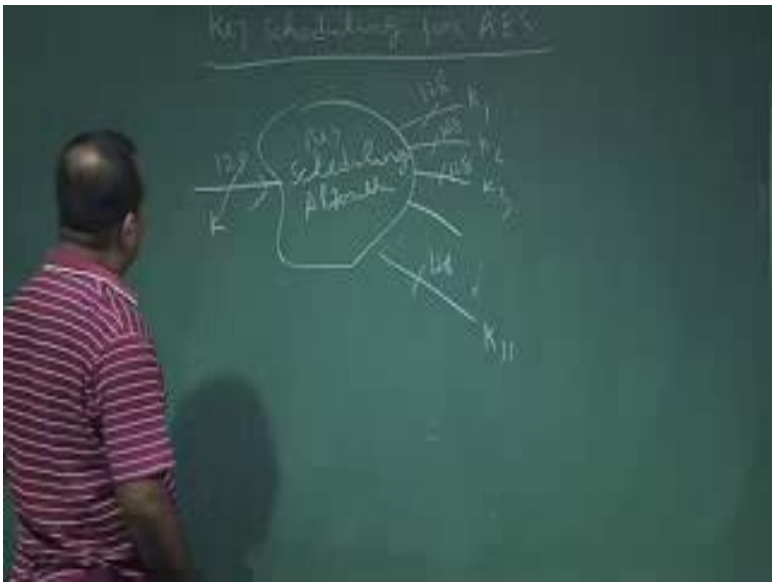
applying this transformation mixed column transformation we are replacing into the new column. So, this is sort of matrix multiplication we need this when we talk about the inversion. So, this is sort of like this.

(Refer Slide Time: 23:18)



This is a state, we have a state and this is a matrix constant matrix and then it will give us because when we talk about inversion of the AES we need this information. So, this is the mix column operation.

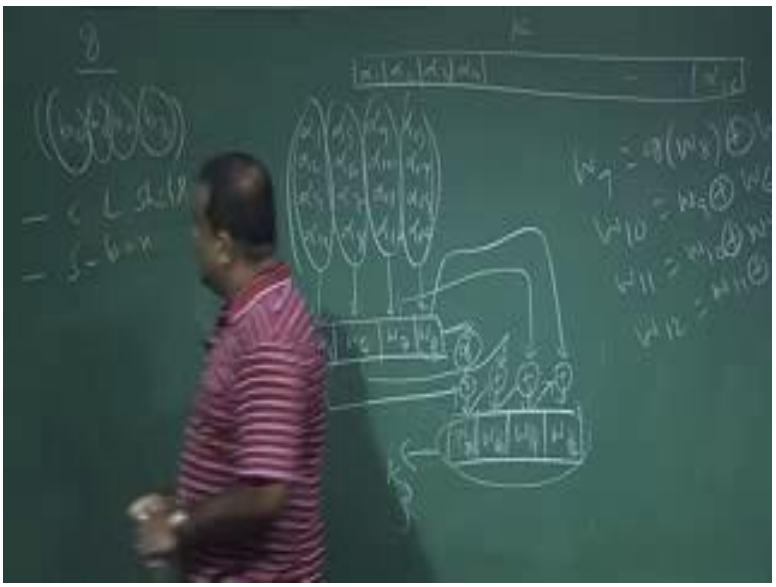
(Refer Slide Time: 23:55)



Now, what is remaining the remaining is the so, we have already seen the sub byte add round key we have seen the sub byte we have seen the shift row we have seen the mix column now we want to discuss what is the key scheduling algorithm for AES key scheduling algorithm for key scheduling for AES. So, this is basically what. So, it is basically give us the round keys the key scheduling algorithm this Algo is public, only secret is the round key's key scheduling algorithm. So, it is taking a that secret keys say at between Alice and Bob and this is the secret keys and this key size is 128 bit because we are talking about AES 128. So, this key size is 128 bit and it should give us how many round keys we need to have 11 round keys. So, it should able to give us K 1, K 2, K 3, dot, dot, do, t K 11 and each of these again a 128 bit. So, we should get a 128 bit round keys from this secret key.

How we can get that? That is called key scheduling algorithm that we have to discuss. So, let us talk about how to get this K 1, K 2, and K 11. So, for this, what we will do?

(Refer Slide Time: 25:37)



We have the key which is 128 bit. So, say we can again brake into because we are doing the; we can again brake into bytes. So, 128 bits means 16 bytes. So, this is the key. So, 16 byte means we can just denote alpha, 0 alpha, alpha 1, alpha 2, alpha 3, alpha 4, like that alpha; alpha 16.

Now, we take a, this state wise. So, this is basically we store this in a state wise like this alpha 1, alpha 2, alpha 3, alpha 4, alpha 5, alpha 6, alpha 7, alpha 8, alpha 9, alpha 10,

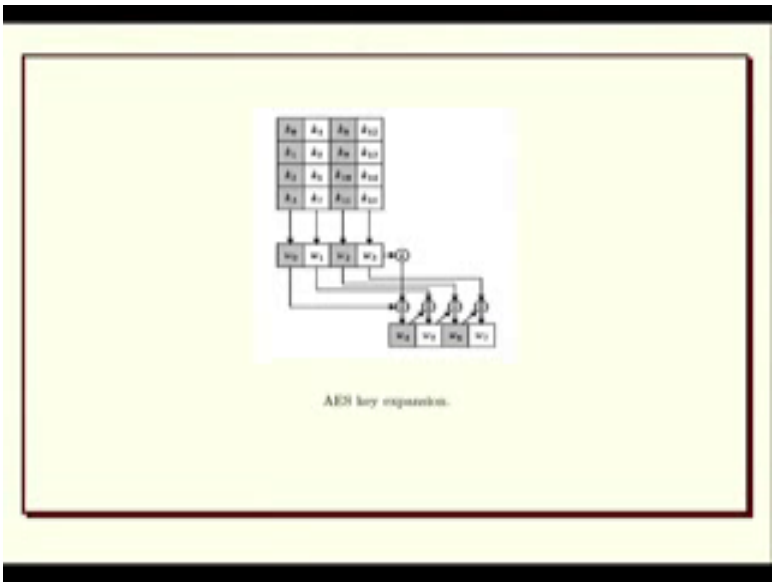
alpha 11, alpha 12, alpha 13, alpha 14, alpha 15, alpha 16. So, this way, we are storing this and then we what we do we just so, these are so what we do? We just take this. So, these are basically we can denote it by K, K 1, K 2, but K we used for the round keys that is why we are denoting by this alphas.

We just copy it into so, these are all 8 bits, 1 byte. So, if we can copy it into 32 so, 4 mean 32. So, this is a W 1, W 2, W 3, and W 4. So, we just copy this, this is 8 bit, 8 bit, 8 bit. So, total is 32 bit. So, we copy this into this 1, our each word is 32 bit and we copied it here and we copied it here and we copied it here and this is our K 1, the first round key this is our K 1, the first round key, now this is 128 bit basically first round key is the original key; the secret key and that we are adding in the first sort of the round of AES and first sort of AES we are adding the first round key. So, this key we are mixing with the plaintext this is the first 1.

Now, how to get the K 2? To get the K 2, what we do? We just apply, so, to get the K 2, we have to get the W 5, W 6, W 7, and W 8. So, what we do? We take this W 4 and we apply the g function we will discuss what is this G function and this G function and we will take this W 1 and we will XOR with this and this is basically W 5. So, this is basically W 5. So, we take this W 4, which is 32 bit, we apply this G function, we will come to that what is G function then again it will give a 32 bit, we XOR with this W 1 and we get the W 5 then how to get W 6 we take this W 5 and we take this W 2 we XOR with this and we get W 6.

Basically W 5 is basically G of W 4 XOR with W 1 and W 6 is basically now we have to W 5 after this W 5 we XOR with W 2 and then we take this W, once W 6 is computed we take this W 6 and we XOR with this W 3 and we got W 7. So, W 7 is basically W 6 XOR with W 3 and similarly once we got this W 7 we XOR with this with W 4 and we get W 8. So, W 8 is basically W 7 XOR with W 4. So, this is the way we calculate this W and this is our once we got this W 5 to W 8 this is our K 2 can you please slide please.

(Refer Slide Time: 30:22)



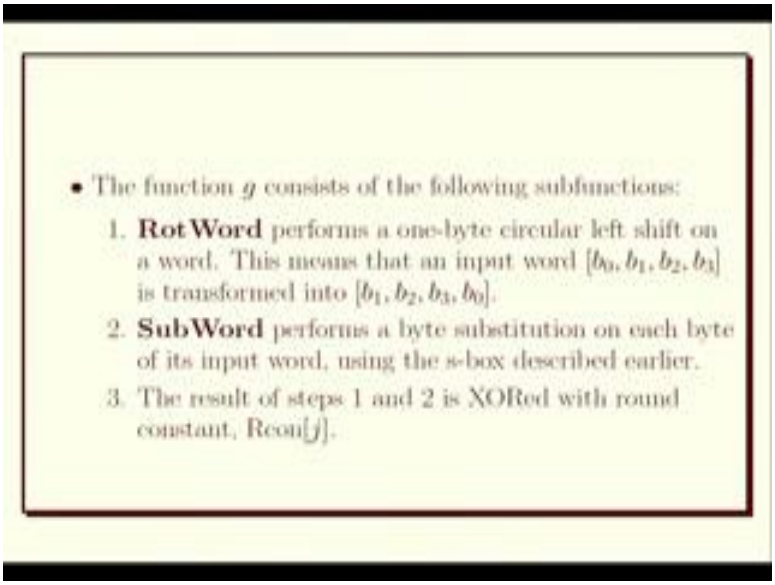
This is the way we are calculating. So, these are all our key original key, this is we are denoting by alphas anyway. So, this is our original keys and this we have copied into these Ws. So, this is the first round key and to get the next round key what we are doing we are just applying this, this with G and then we got this and then we XOR with this like this.

Now, the question is how we get K 3. So, to get K 3 we will do the same thing we just now we have W 5 to W 8 we just copy this here. So, W 5, W 6, W 7, W 8, and now we have to get W 10 sorry, W, W 5, W 6, W 7, W 8, W 9, W 10, W 11, and W 12. So, how to get this? Similar way we have to do. So, we take this g we apply on W 8 and we XOR with this we get. So, basically W 9 is basically we apply G on these W 8 and we XOR with this W 5 and W 10, once we have the W 5 is W 9 is calculated W 9 XOR with W 6 and W 11 is W 10 XOR with W 7 and W 12 is basically W 11 XOR with W 8 like this. So, we got this is basically our K 3 and similarly we continue this process until we get K 11.

Basically these Ws are giving us the key round keys now the question is what is G? G is basically taking a sorry, G is taking a W which is 32 bit; 32 bit we can denote by b 0, b 1, b 2, b 3, each is 8 bit. So, what we are doing? We are just, G consists of this is the G operation. So, G consists of 3 steps, first step we are rotating it. So, this rotation is

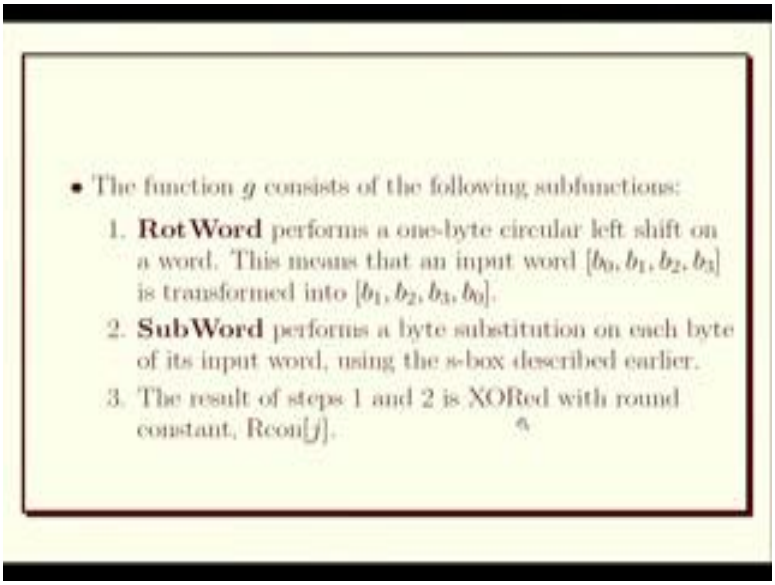
basically so, we are rotating it. So, basically this is going. So, if you denote by this way b 0, b 1, b 2, b 3.

(Refer Slide Time: 33:06)

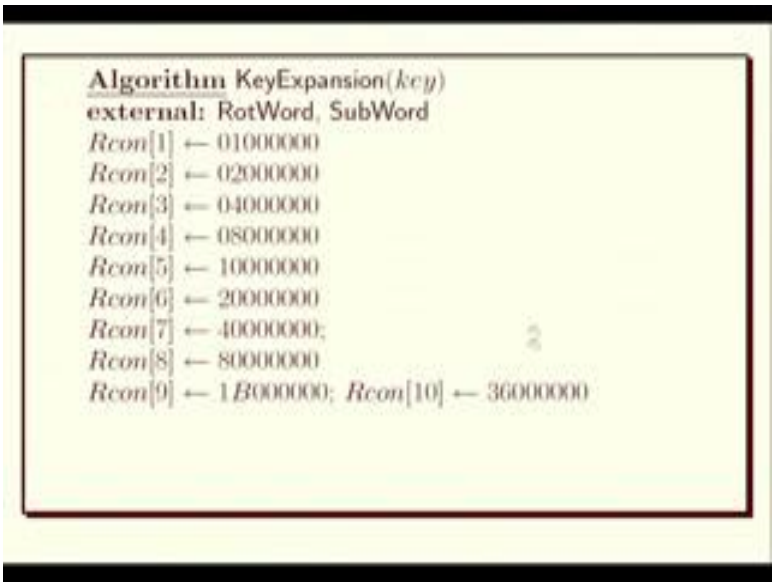


Can you go to the slide please? So, this b 0 is just we are making a rotation. So, a this circular rotation circular left shift, we are doing. So, this is the first operation we are doing. So, circular left shift. So, circular left shift, we are doing and then we are performing the S-box sub byte operation on each of this after this sub byte operation on each of this.

(Refer Slide Time: 33:37)

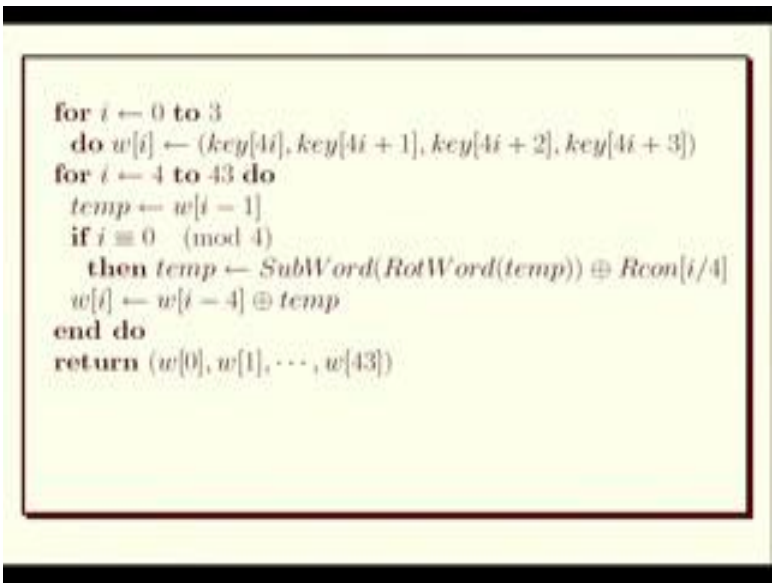


(Refer Slide Time: 33:52)



And then we are slide please then we are performing the sub byte operation on each of that b i's basically we are applying the AES S-box and then we are adding the round keys and we are adding some constant with this to get the this just for these constants are basically this. So, these are the constant we are adding with this b i's; with the b i's to get this g of this.

(Refer Slide Time: 34:04)

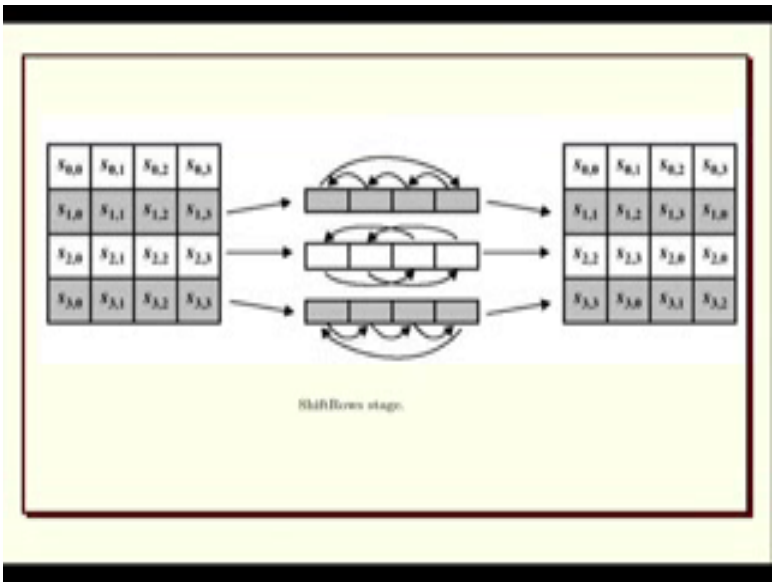


This is the rotation and we adding the constant these constants are given and then we are getting this b, b i's G of G of w. So, this is the operation for g now we will talk about bit

of inverse how we can inverse this AES. So, for that inversion we have to think about how we can invert this operation like add round key shift row S-box and the mix column. So, for add round key inversion just we will again XOR because XOR is the reverse of it is. So, we will XOR again. So, add round key inversion is simple.

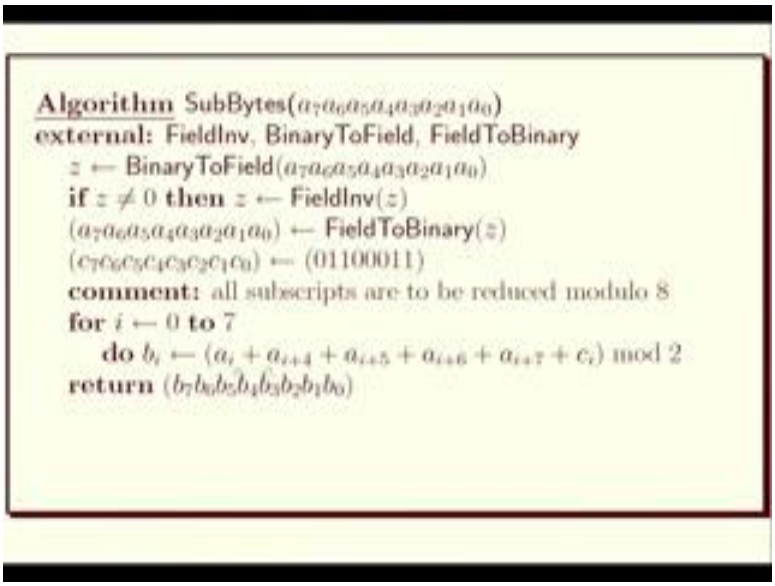
Now, what about this sub byte operation inversion? So, for sub byte operation, so for shift row operation for inversion is basically by reverse.

(Refer Slide Time: 35:01)



This is the original shift row operation. So, inversion shift row operation is just we will do the reverse, now what about the inverse sub byte operation.

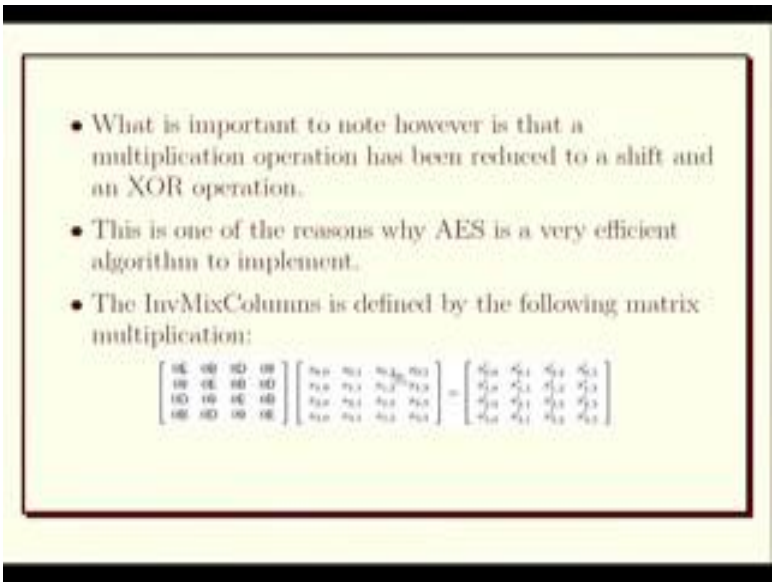
(Refer Slide Time: 35:19)



This is the sub byte operation. So, sub byte operation having this affine transformation. So, this is affine transformation, after this we are doing this additions. So, this is basically affine transformation we have, this is the affine transformation, we have in the sub byte operation.

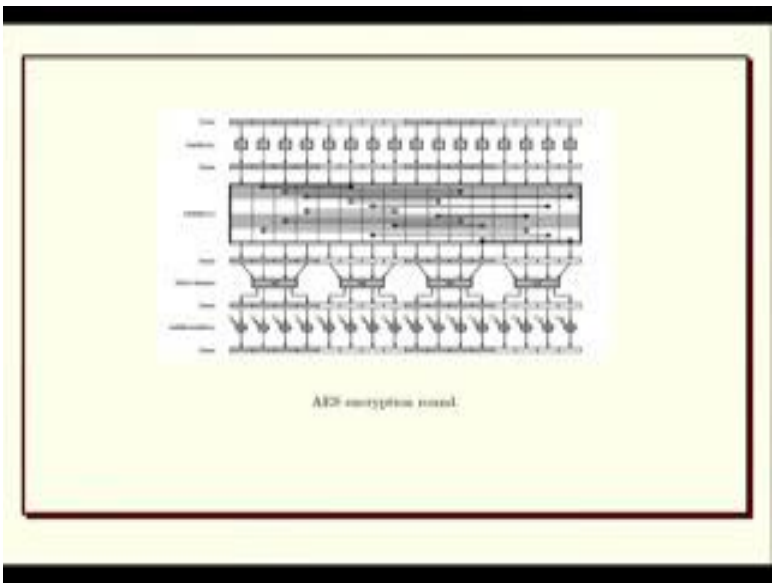
Now to get the inverse so, we need to XOR with these b i's then we will get this a i's and then we have to apply the field inverse again to get back these a i's. So, this is the inverse for the sub byte operation and then we have to talk about the inverse for the mixed column operation. So, mix column operation is basically this. So, we have this columns and this is the matrix we are multiplying. So, we are taking individual columns and then we are converting into a new column. So, for a mixed column operation, it can just have this inverse of this like.

(Refer Slide Time: 36:06)



This is the mix column operation, we have to take the inverse matrix of this and then it will give us the inverse mix column, so this is basically a matrix inversion we are doing.

(Refer Slide Time: 36:20)



This is a typical 1 round AES. So here if we observe, so, we are putting the plaintext into the state and then we are applying the S-box operation this sub byte operation and then again we are storing into the state and then we are doing the mixed shift row operation we are shifting the position of these byte this is byte wise operation. So, we are shuffling

the bytes here and there then again, we are storing into the state and then again we are doing the mix column operation.

So, we are just taking it to the columns and then that matrix we have then again this 1 and then we again adding the; add round key. So, this is the again; the state we are getting. So, there are many operations are involved so that is makes AES secured.

Thank you.