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

Lecture - 51
Cryptonalysis and Stream Cipher

(Refer Slide Time: 00:36)

So, we talk about cryptanalysis on stream cipher at around stream cipher, specially we will talk about algebraic at around stream cipher. So, let us just recap what is a stream cipher. We have seen the LFSR stream cipher; basically in stream cipher, we have a key stream generator, this is basically a pseudo random bit generator because the bit it is. So, this is basically taking initialized by the initial key, so the key secret key k which is shared between Alice and Bob. So, you have two party Alice and Bob. So, they have a common so this is symmetric k encryption, they have a common key k which is some bit.

Now, if it is LFSR based stream cipher then basically what we have we have a LFSR say for example, if we have four bit LFSR, and this is four bit, and we have some connections over so depending on the polynomial we have some connections. And this is the state transition met this is the way it change the state. So, we have a feedback from this bit, and this bit and this bit and so. So, if k is four bit, so k 0, k 1 or the other way round, so k 3, k 4, k 2, k 1, k 0. So, we initialize this state by this way and we run the LFSR. So, it will keep on generating the k 0 for first come out then it will be k 1, k 2, k 3 then it will be XOR with this k 1 XOR k 2 XOR k 1 like this so then k 1. So, like this. So, this is the key stream. And this key stream we are going to XOR with the plaintext bit p 0, p 1 like this and we got the c 0, c 1 like this and this c 0, c 1 is going to Bob. So, this is the encryption.

Now, how Bob will decrypt it? So, Bob is having the same LFSR that is same bit generator; and Bob is having the same key, which is the secret key shared between Alice and Bob. So, Bob will just initialize the state by this key and it is the same LFSR. So, this is one example. So, Bob will generate the same key stream bit k 0, k 1 like this and Bob will XOR this with the c 0. So, Bob is receiving c 0, c 1, and Bob will get the c 0. So, k 1 XOR with the c 1, so Bob is getting the plaintext bit. So, this is the decryption which is doing by Bob and this is the encryption which is done by Alice.

Now, we have to think of what Oscar can do. So, now, if Oscar knows some knows some plaintext and ciphertext bit, this is (Refer Time: 03:56) known plaintext set up. If Oscar knows some plaintext and ciphertext, now this cipher text is basically the linear function of the plaintext and the key.

(Refer Slide Time: 04:21)

So, basically what we have, so we have a Z t which this is basically function of k, k is the secret key. So, Z t is the function of secret key, and it is basically what we are doing we are getting the C t as Z p XOR with P t. Now, Z p is the t-th key from the keystream and this is the t-th ciphertext bit and this is the t-th plaintext bit. Now Z t is basically C t

XOR with P t. Now, Z t is basically function of f function of k, k is the secret key shared between Alice and Bob. So, that is the initial source we are loading the secret key and then we are running the key stream generator. So, this is basically C t of this.

Now, if this function is a linear function then basically we have some system of equation linear equation, then we can solve this to get the this key once we get the k, then everything is gone, because k is the secret shared between Alice and Bob, so that is the attack on LFSR simple LFSR based stream cipher. So, to avoid that, you need to put some non-linear functions in the LFSR baser stream cipher.

(Refer Slide Time: 06:25)

So, some non-linear key we need to introduce, so that is why we say L LFSR, L 1, L 2 like this, so n many LFSR. And these we are parsing function f non-linear function. So, this is our dot dot dot, this is our key stream, this is our key stream. So, what we are doing. So, this LFSR is having the same length. So, this is Alice is doing. So, this is the encryption Alice is doing. So, what Alice is doing? Alice and Bob they have shared with a common key. So, they are loading this secret key in each of this LFSR and they are running this, and they are getting some output foundation let us say this is x 1, x 2, x 3 like sorry x n like this. So, this is basically a function of x 1, x 2 this is a Boolean function. So, it is basically a function f is a function of 0, 1 to the power n to 0, 1. So, this is a Boolean function.

We can realize this function in algebraic form or we can have a truth table for this function. So, we can have a truth table like say we have these values x 1, x 2, x n and we have this value z. So, this is the z the keys stream. So, we can have all possible combination of these 0 0 0 1. So, we can have the output. So, we can expressively have the truth table for this function to represent or we could have algebraic form, like we can have this function to be say x 1 x 2 XOR with x 2 x 3 x 4 like this. Now we call this function to be balanced if the number of zeros and number of ones are same then we call this function to be balanced function. And this function you want to be non-linear function, because these are the non-linear terms and this will make our cipher to be secure in the terms of the algebraic data, because we are not able to find the linear equation out of this.

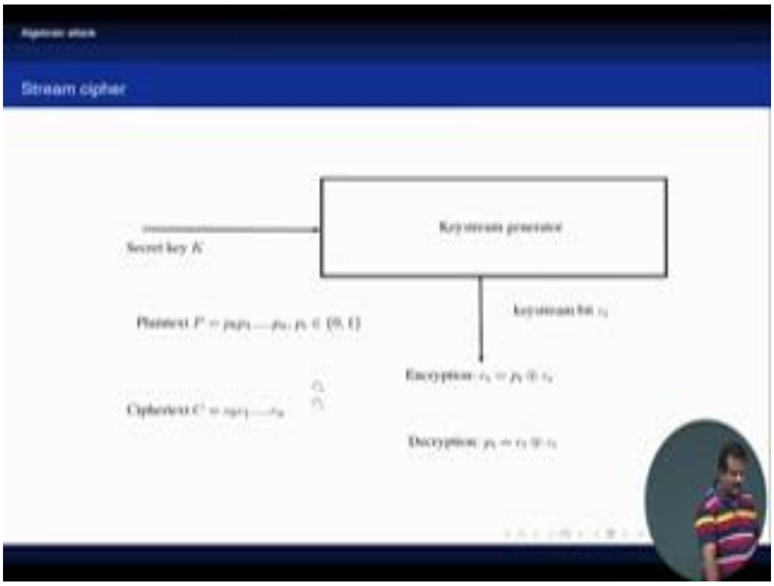
So, we will see how we can try to get a linear equation by linearization of this term, so that we will see. So, this is the general form of non-linear, this is non-linear function which is taking the input from the linear LFSR, and it is generating the output. Now, this is the key stream, we are getting and this key stream we XOR with the plaintext bit to get the cipher text bit. So, this is the encryption. So, decryption Bob is doing the same thing Bob is having this structure. So, what Bob will do? So, Bob will just put the same secret key shared between Alice and Bob; and then Bob will run this key stream and Bob will extort this key stream with the key stream bit with the cipher text bit, and Bob will receive the plaintext bit. So, this is the general structure of a LFSR bit.

(Refer Slide Time: 10:14)

But any stream cipher, so for any general stream, this is synchronized stream cipher. So, any stream cipher, what we have we have a key stream generator or this is called pseudo random number generator. Why pseudo, we know why pseudo because it is cannot give a truly random sequence, so that is why it is a basically it is a finite state machine. So, this is taking input as, we initialize this state as this state we denote by S t, and we initialize this by the secret key k shared between Alice and Bob - S 0. So, these states are initialized by the secret key shared between Alice and Bob. And then we have a state update function g which is taking the current state g of S t and which is generating the next state if S 0 is the K. So, this is the state updating function.

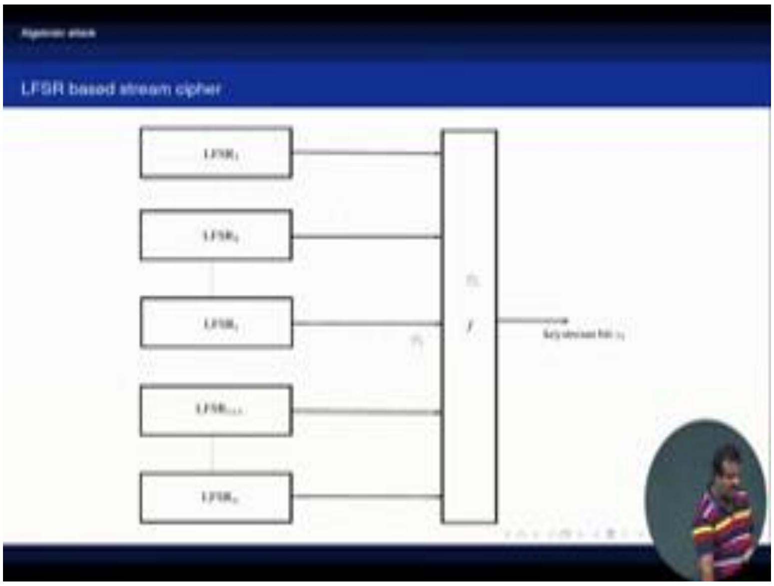
For LFSR it is the feedback function. So, we extort those bits which are participating in the feedbacks and then we are taking the feedback in the last bit and then we have another function which is taking the current state and apply a function h. So, this is giving us the key stream. So, h of S t is basically Z t. So, the key stream bit of the t-th key stream and then we extort this key stream with the plaintext bit P t, and we generate the ciphertext C t. So, this is the general structure of a synchronized stream cipher because here we are not involving the plaintext in this updation function or in this key stream generation that is why it is called synchronized. In the asynchronized will see will see in the next lecture some modern stream cipher. So, there we will see there are some asynchronized stream cipher, where the plaintext is also involved also participating in this is the function of the plaintext also, but here we are taking just a synchronized stream cipher.

(Refer Slide Time: 13:03)



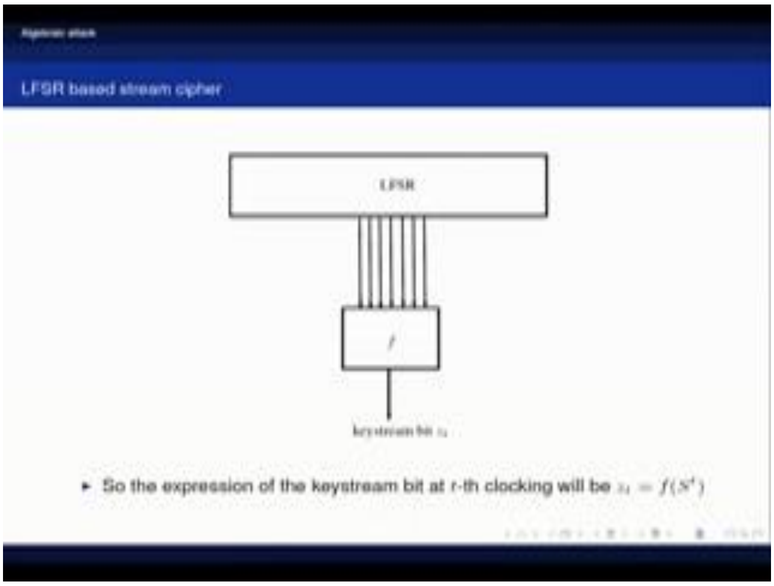
So, can you go to the slide please? So, this is the general structure of a stream cipher we have a secret key and we have a key stream generator and we are generating this key stream generator to get the key stream. And this is the plaintext, plaintext bit p 0, p 1, p n and then we are expiring with the c i key stream bit and getting the ciphertext bit. So, this is the encryption what Alice is doing. And after receiving this ciphertext bit, Bob is just having the same structure same stream cipher and Bob is generating the key stream bit again by taking the initial value of the state as the secret key and Bob is decrypting and getting the plaintext back.

(Refer Slide Time: 13:50)



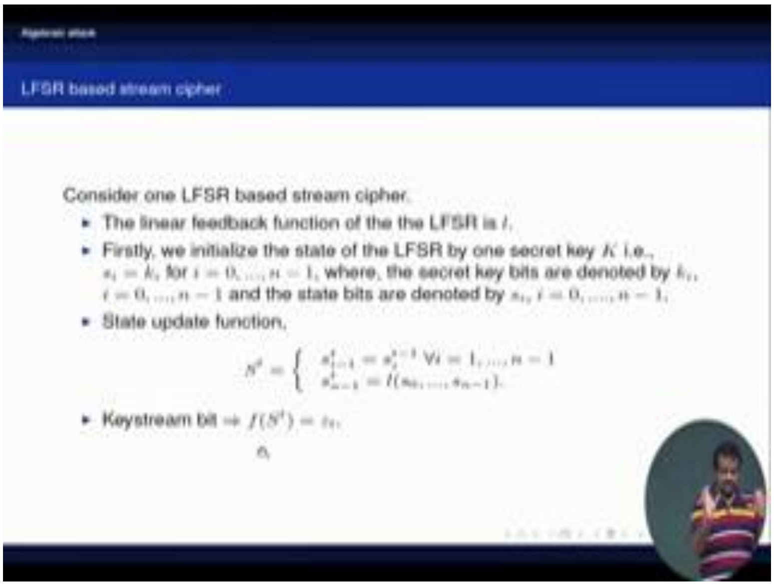
So, this is we have seen if it is LFSR based stream cipher, we have n LFSR which is taking the input to the non-linear function f, and we are generating the key stream.

(Refer Slide Time: 14:05)



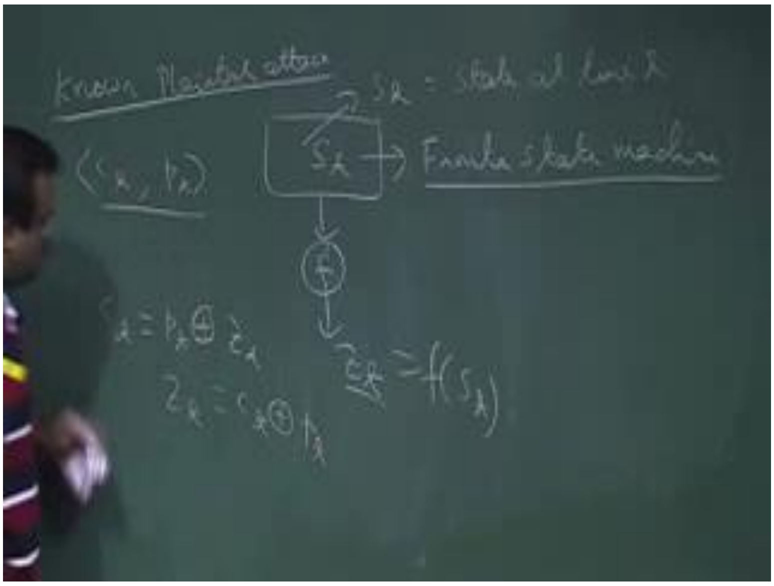
So, this is the structure. So, we denote this Z p as f of S t. So, S t is the state. So, state updation function. So, in our example it was h. So, basically g, so this is basically f we are denoting. So, Z t is basically sorry this is s. So, this is the key stream generator. So, we have taken this f, we have taken this state and we apply this f on it and we will get the key stream.

(Refer Slide Time: 14:42)



So, now this f is for a LFSR based stream cipher, we know this f is basically updating the function and this LFSR has this updation function. So, for all the bits, it is shifting the one position except the last bit - nth bit. So, nth bit is basically linear feedback of this bit depending on the polynomial which we are going to use for that.

(Refer Slide Time: 15:20)



So, this is basically and the S t is basically, so we take a state so basically what we have we have this is S t this is the finite state machine. So, it could be LFSR or it could be collection of LFSR, then we have f it could be anything. So, basically we have finite state machine and this machine this is the state at time t, so this is our S t. Now, it is applying a function f and this is giving us Z t. So, Z t is basically f of S t. So, this is the way we think that.

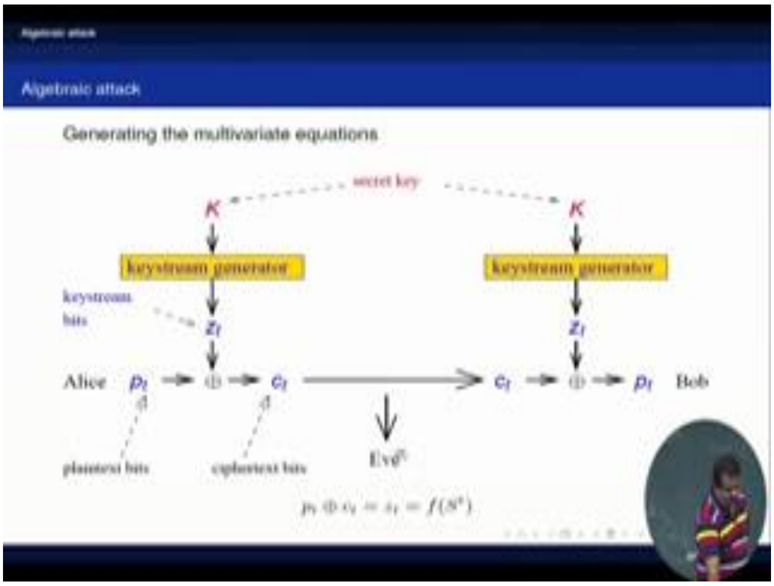
(Refer Slide Time: 16:12)

So, then for algebraic attack what we need to do we need to write this Z p in terms of this f in terms of algebraic equation. So, expressing the whole cryptosystem as a large system of multivariate algebraic equations, which can be solved to recover the secret key, so algebraic attack based on two basic strategies; they are basically we construct a system of algebraic equations involving the secret key using some known plaintext and ciphertext because this is a known plaintext attack. We know some plaintext bits and ciphertext bits. So, basically what we are doing this is the t. So, our ciphertext bit is basically t-th ciphertext is basically p t XOR with Z t. So, now if you know this ciphertext, if you know this c t and p t for some p, so this is a known plaintext attack. So, if you know this then what we basically will do? We then, Z t is basically c t XOR p t.

(Refer Slide Time: 17:42)

So, now, this is known, if this is some value, then what we have, we have basically some functions like. So, we know this Z t is basically f of S t is basically some c t XOR p t, now this value is known for some t. Now, from where we basically have a system of equation then we try to solve this system of equation. So, this is the construct the system of algebraic equations involving the secret key, because f 0 is the secret key, s 0 which is state is neutralized by the secret key.

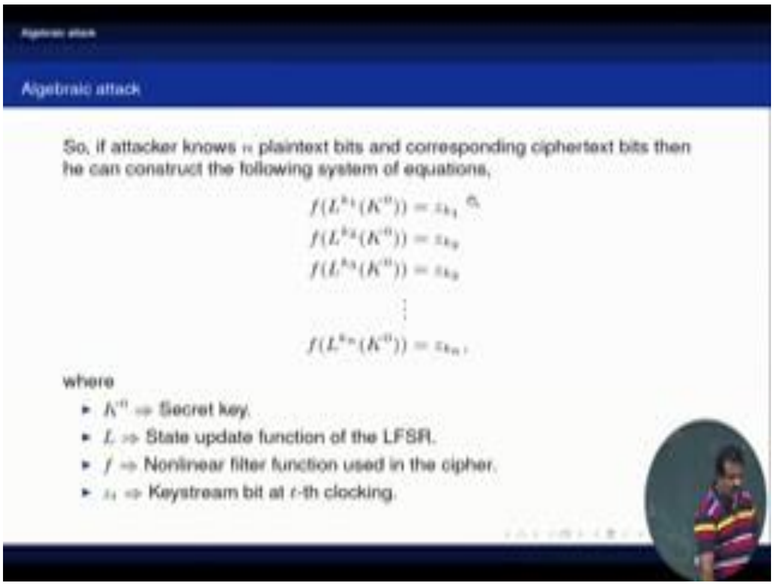
(Refer Slide Time: 18:27)



So, this is the attack model of algebraic attack. So, here what we are doing, we have a key stream generator which is basically the finite state machine, which is a pseudo random bit generator it is a finite state machine. So, it is initialized by the state is initialized by the secret key shared between the Alice and Bob. And it is generating the key stream serially and then this key stream is XOR with the plaintext bit and generates the ciphertext bits, and send it to the Bob over the public channel. Now, after receiving the ciphertext Bob is doing the decryption like this. Bob is again generating the key stream, because Bob is also having the same pseudo random bit generator and Bob is putting the secret key as a initial state, and Bob is running the same thing and then Bob is extorting with the ciphertext with the this key stream bit and getting the plaintext back.

Now, what Alice is doing? So, Alice is receiving this ciphertext, now suppose Alice is having this ciphertext bit and few ciphertext bit and the corresponding plaintext bit. So, this is the known plaintext attack; then basically the attacker is having this system of equation.

(Refer Slide Time: 19:52)

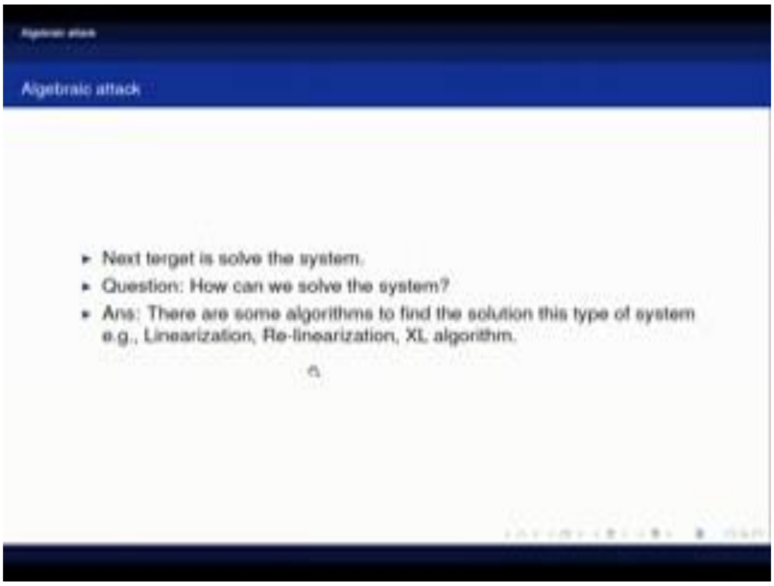


So, basically if the attacker knows plaintext bit, and the corresponding ciphertext bit, then he can construct the following system of equations. So, this basically this is a LFSR based stream cipher because these are the LFSR bit updates. So, every state is initializing by k 0. So, this is basically say for example, for if you know the first k plaintext on the corresponding cipher text.

(Refer Slide Time: 20:36)

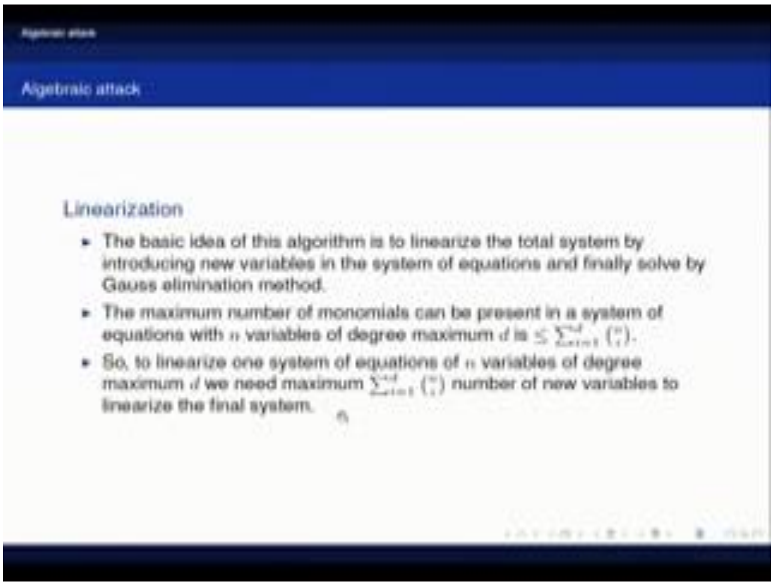
Then what we have, we have this Z of k 1 which is basically f of L of k 1 k 0, this k 0 is the secret key shared between Alice and Bob or this is the private key shared between Alice and Bob. Because this is coming from that, because we know that p 1 XOR with c 1, it is basically z all like this. So, similarly we have s 2 which is basically f of L k 2, so like this. So, these are all state updated functions for LFSR, but it could be general stream cipher. So, this way we have n many this key streams which is corresponding to k 0. So, these are all function in k 0. So, this is a system of equation, where this k zero is the secret key and L is the state update function for the LFSR this L, and f is the non-linear filter function used in the stream cipher, and s t is the key stream bit in the t-th clock. That means, t-th key stream, but it could be other than LFSR also it could be general stream cipher then we have a finite state machine and it is updating.

(Refer Slide Time: 22:14)



So, what next, so our next as so we got some system of equation. So, our next target would be to solve this system of equations, because this system may not be a linear system. If it is a linear system you could directly use the Gauss elimination method or some matrix method to solve it, but this system may not be always linear system. So, we will try to make this system to be linear system, so that is the method called linearization. So, we will talk about that. So, the question is how we can how can we solve this system. So, there are some algorithms to find the solution for this type of system one is linearization we will talk about that, then re-linearization, then XL algorithm to solve this system of equation.

(Refer Slide Time: 23:10)



So, what is the linearization, basic idea of linearization is to this linearization is the total system by introducing new variables in the system of equations and finally, solve. So, this system may not be a linear system. So, we will just introduce some new variable in this system to make it linear system. We will say if we have a variable say x 1, x 2, this is not a linear, but you can put this as y 1, so that way we will solve this. Now, once we have a solution, once we solve this in terms of y 1 then we will try to get the solution for x 1 and x 2. So, this is called linearization.

So, the then the maximum number of monomials can be present in the system of equations of n variables of degree d is must be less than equal to summation of i is equal to 1 to d n to i. So, to linearize one system of equations of n variables of degree maximum d, we need maximum this many number of new variables in the final system. So, this numbers should have controlled because otherwise it will be a huge system then we may not get the solution, so that is also one important thing.

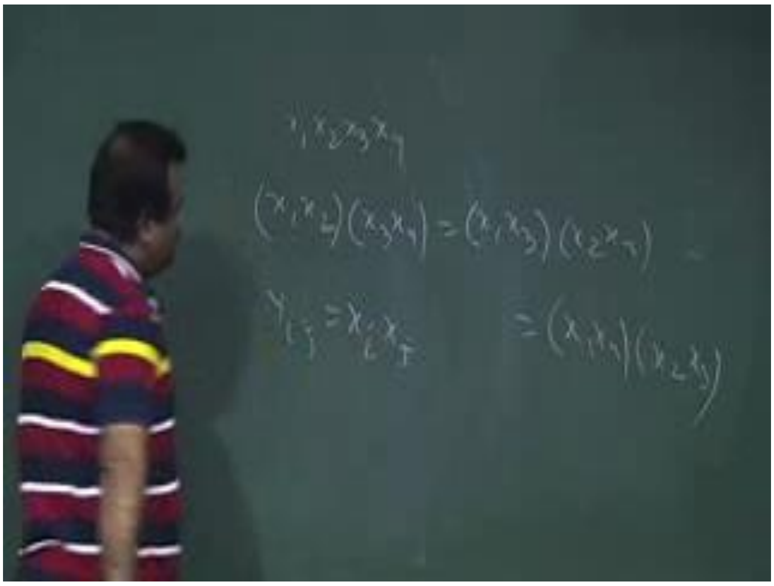
(Refer Slide Time: 24:32)

So, then final system will have unique solution if the number of independent equation is same as the number of variables present in the final system, so that matrix has to be invertible, but in many cases it is not happening. Then we need to think of generating some extra linearly independent equations.

(Refer Slide Time: 24:56)

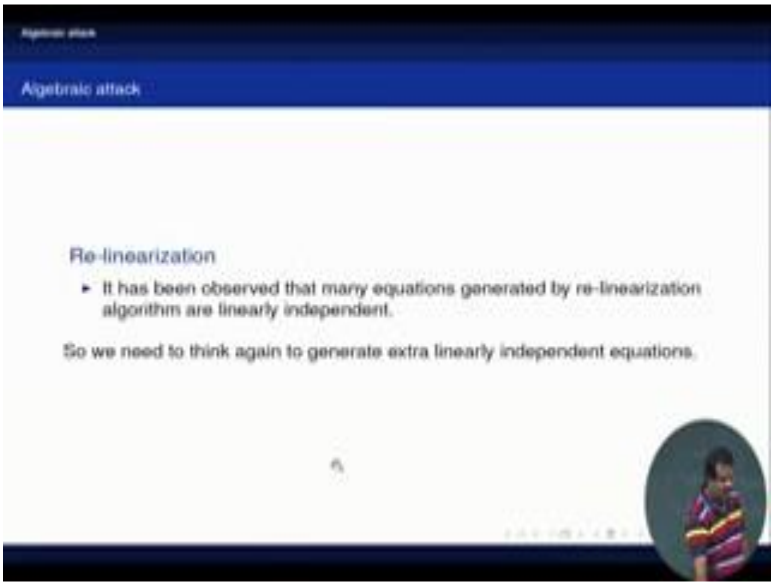
So, suppose we have this example. So, we need to do the re-linearization. So, how we could do that? So, we know that the multiplication of the variables in a commutative operation in this re-linearization method we use with the commutative property to generate the more equations. So, we have to generate the more equation, because of our system of equation what we are getting is not invertible I mean it is not giving the unique solutions. So, consider for example, suppose we have a monomial this terms this non-linear x 1, x 2, x 3, x 4. Now, we can rewrite this monomial in a different way like x 1 x 2 then x 2 x 3 x 4 then x 1 x 3 and x 2 x 4 or x 1 x 4 x 2 x 3. Then we rename each of these different variable as y 1 i because there are, so this is y 1 2 this is y 1 3.

(Refer Slide Time: 26:09)



So, like if you have this term say x 1, x 2, x 3, x 4. So, this can be written as x 1 x 2. Now, we can put these or this is same as x 1, x 3, x 2, x 4 or this is same as x 1, x 4, x 2, x 3. So, now these we are denoting by y i j, y i j means x i x j. So, we have many choices. So, we can have corresponding equation. So, we rename this each of these different variable as y i j is; after substituting this new variable, we will get some new linear depend independent equations of this forms.

(Refer Slide Time: 27:00)



So, then it has been observed that many equations generated by re-linearization algorithm are linearly independent. So, once we got the linearly independent equations then we solve this then we have system is consistent then we can get a solution of this system and we can get the corresponding secret key. So, this is called algebraic attack. We basically generate some equations then we try to solve this equation by some of the method like linearization, XL method, Gauss, so once we get the system of equation, which is consistent then you apply the Gauss elimination method to solve this.

Thank you.