

# **Cryptography and Network Security**

**Prof.D.Mukhopadhyay**

**Department of computer Science and Engineering**

**Indian Institute of Technology, Kharagpur**

**Module No.#01**

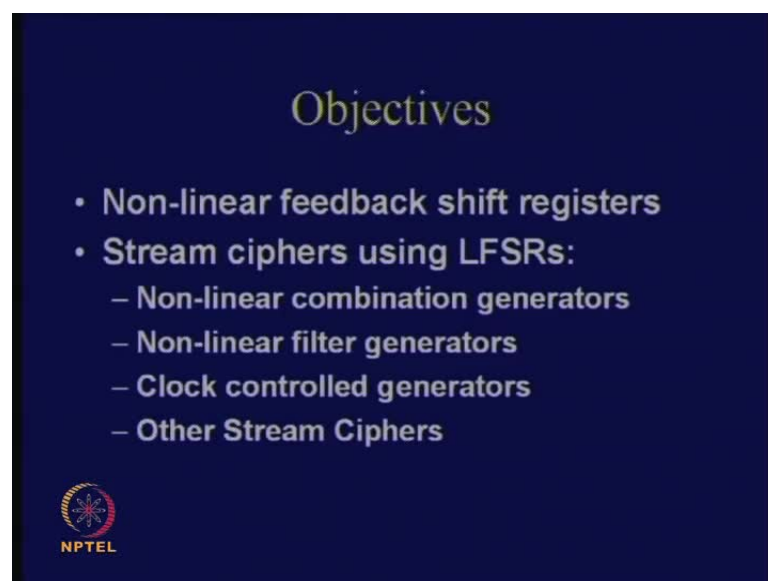
**Lecture No.#21**

**Stream Ciphers (Contd.)**

We will continue with Stream Ciphers. In the last day's class, we had discussed about berlekampmassey algorithm and the reason why we discussed, was to point out the fact that a single LFSR system or a single LFSR based stream cipher is not adequate to prevent attacks of the type. For example, known plaintext attack or chosen plaintext attack from the streams you are able to generate back the input seed of your LFSR based system and from there, the stream pattern is quite predictable in that fashion.


So although LFSR based stream ciphers were quite good, because they were amenable to easy efficient hardware implementations they had good proper statistical properties, the sequence had good statistical properties but, the patterns were quite predictable.

(Refer Slide Time: 01:09)



**Objectives**

- **Non-linear feedback shift registers**
- **Stream ciphers using LFSRs:**
  - **Non-linear combination generators**
  - **Non-linear filter generators**
  - **Clock controlled generators**
  - **Other Stream Ciphers**

**NPTEL**

We need need to do something different and in that pursuit in today's class, we will see: Non-linear feedback shift registers, and stream ciphers. Using more than one LFSRs that is using multi LFSRs system. Under that topic, we will discuss about non-linear combination, I mean non-linear combination generators, non-linear filter generator something which is called as clock controlled generator and we will also point out that there are some other stream ciphers which are actually not LFSR based therefore, there are modern proposals which are not based on LFSRs and they are also being quite followed, studied.

(Refer Slide Time: 01:51)

### Non-linear feedback shift registers

- A Feedback Shift Register (FSR) is non-singular iff for all possible initial states every output sequence of the FSR is periodic.

NPTEL

First of all, let us start with non-linear feedback shift registers. As the diagram says that, it is exactly the same as that of feedback shift register which we have already seen but, only the difference is, in this case your feedback is non-linear.

In an LFSR, our feedback was a linear algebraic equation. If you remember, this particular thing that is suppose you have got else pages in your LFSR or in your shift register then, the contents of each of the delay elements are being taken as an input to your feedback polynomial. But, in case of a non-linear feedback shift register this particular polynomial or this particular equation is non-linear. We know what the definition of a linear and non-linear is. It follows exactly if I write this or describe this in the algebraic normal form or ANF form then, that means that this particular expression would encompass terms which are end terms.

They are not only based on ex-ors but, there are certain terms which are also end based terms. Feedback shift register (FSR) is called a non-singular feedback shift register if and only if, for whatever be the initial states of your FSR the sequence is, which this generates are periodic in nature which means, that there is a definite periodicity in the sequences which are being generated. In that case, (Refer Slide Time: 01:51) we say this particular polynomial is a non-singular feedback shift register. As we have studied, we need periodicity but, the thing is that we also need to maintain or ensure that your periodicity is quite high because small LFS, small periodicity is not like because of possible attacks.

(Refer Slide Time: 01:51) Therefore, we also need to ensure that although we are replacing this, we on one hand in order to prevent the berlekampmassey algorithm to work we require our feedback polynomial to a non-linear equation but, at the same time because of this non-linearity the easy analysis of the LFSRs are stopped. We cannot predict that, it is periodic and things like that therefore, even if we allow non-linear terms into my feedback polynomial, I must also give guarantees of this periodicity that is, periodicity should be at least for example, greater than so and so.

So we have to produce guarantees of that nature therefore, we need to be little bit more careful because, the analysis becomes harder compared to LFSRs which are called easy to analyze.

(Refer Slide Time: 04:37)

*de Bruijn Sequence*


An FSR with feedback function  $f(s_{j-1}, s_{j-2}, \dots, s_{j-L})$  is non-singular iff  $f$  is of the form:

$$f = s_{j-L} \oplus g(s_{j-1}, s_{j-2}, \dots, s_{j-L+1})$$

for some Boolean function  $g$ .

The period of a non-singular FSR with length  $L$  is at most  $2^L$ .

If the period of the output sequence for any initial state of a non-singular FSR of length  $L$  is  $2^L$ , then the FSR is called a *de Bruijn FSR*, and the output sequence is called a *de Bruijn sequence*.

 NPTEL

There is something which is called as de bruijn sequence. It is defined as follows, an FSR whose feedback function that is  $f_j$  minus, which operates as,  $s_j$  minus 1 to  $j$  minus 1. It is non-singular so, what is meant by non-singularity? It means the sequences which are being generated are periodic, for whatever be the input seeds of your FSR.

It is periodic. It is non-singular if and only if,  $f$  is of this particular form so, what is the form  $f$  is equal to  $s_j$  minus 1 **xored** with whatever, be some for any boolean function  $g$  which operates on the remaining values. You see that  $s$  runs from,  $j$  runs from, I mean the index of the inputs of  $g$  runs from  $s_j$  minus 1 to  $s_j$  minus 1 plus  $L$ . This is a non-singular, we can assume this particular fact and the period of a non-singular FSR with length  $L$  is at most  $2^L$ .

The period of a non-singular FSR with length  $L$  is at most  $2^L$  therefore, the idea is for example, if you have a 3 bit FSR which is of this nature then the period can be at most  $2^3$  because, there are 8 possible steps but, for those particular generators for which the period is always equal to  $2^L$  is the initial state of your FSR are known as de bruijn generators. That means a de bruijn generator, generates all the states or all the possible state values in its cycle so, if you just compute the cyclic states then, all the possible  $2^L$  states lie in the cycle therefore, (Refer Slide Time: 04:37) this is the definition of a de bruijn FSR and the sequence is which it generates are known as de bruijn sequences.

I think, we saw one example of de bruijn sequences. if you have really pointed upon the point which I have told you to ask is that, if you remember that maximal length LFSR had generated for example, a 4 bit maximal length LFSR had generated all  $2^4 - 1$  non 0 values in 1 cycle, only thing which was not in that cycle was the 4 0 state that is all 0 state.

If you remember that this particular sequence which is generated, we have seen that the feedback polynomial in case of an LFSR, was a maximal polynomial was a primitive polynomial. The sequences, which it generated were actually something which was defined as  $m$  sequences so, if you remember that I had given you one question to think upon how can you modify that particular LFSR structure, so that the all 0 state is also included in the cycle. If you do that, it becomes a de bruijn generator.

You can actually take an maximal length LFSR and if you can solve that exercise then you can modify that and produce something which is called a de bruijn generator.


(Refer Slide Time: 08:20)

**Example**

$$f(x_1, x_2, x_3) = 1 \oplus x_2 \oplus x_3 \oplus x_1 x_2$$

t	x <sub>1</sub>	x <sub>2</sub>	x <sub>3</sub>
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1

t	x <sub>1</sub>	x <sub>2</sub>	x <sub>3</sub>
4	0	1	1
5	1	0	1
6	0	1	0
3	0	0	1



This is how you can do it but, before that let us consider one example of a de bruijn generator. It says that; consider this particular polynomial or equation which is called  $f(x_1, x_2, x_3)$ . It operates upon  $x_1, x_2$  and  $x_3$ , it is equal to  $1 \oplus x_2 \oplus x_3 \oplus x_1 x_2$  so, these are very simple feedback polynomial which is been considered. If you start from 3 zeros for example, you can work out this exercise and you can see that this 0 will get shifted here, this 0 will get shifted here and this is 1, you can easily understand why it is 1. You can just feed these values, all these are 0s and only 1 remains therefore, this becomes 1.

What about the next stage; this 1 gets shifted here, this 0 gets shifted here, and what is the feedback that you get so, you see that the feedback here can be obtained by computing this particular value and in this case, I am trying to compute  $x_2 \oplus x_3 \oplus x_1 x_2$  is equal to 0,  $x_3$  is equal to 0,  $x_1 x_2$  is equal to 0, you get back 1.

Similarly, you can see that it will be 1 xored with  $x_2$  which is 1 xored with 1 is 0 so xored with 0 is again 0, and xored with  $x_1 x_2$ , you can work out this example, can you just find out, what is the feedback value? this will be 1 xored with  $x_2$  is 1 xored with 1 is 0 xored with 0 and so  $x_1, x_2$  is 1, you get back 1 so similarly, you can work out the

remaining things also and you will find that all the eight states are actually lying in one cycle.

So therefore, this is an example of a de bruijn generator so, the idea that next we will considered is that given a say a three bit maximal length LFSR, how to produce a three bit maximal length LFSR because in that case I just need my feedback polynomial to be primitive.

(Refer Slide Time: 11:01)

**Converting a maximal length LFSR  
to a de-Bruijn FSR**

Let  $R_1$  be a maximum length LFSR of length  $L$  with linear feedback function:

$$f(s_{j-1}, s_{j-2}, \dots, s_{j-L}).$$

Then the FSR  $R_2$  with feedback function:

$$g(s_{j-1}, s_{j-2}, \dots, s_{j-L}) = f \oplus \bar{s}_{j-1}, \bar{s}_{j-2}, \dots, \bar{s}_{j-L+1}$$

is a de Bruijn FSR.

NPTEL

In that case, I know how to produce a three bit maximal length LFSR? How can I modify to obtain a such a de bruijn generators, this is how you can do it and its quite simple to follow. In order to convert, say that let R 1 be a maximal length LFSR of length L with linear feedback function  $f(s_{j-1}, s_{j-2}, \dots, s_{j-L})$  therefore, this is an maximal length LFSR.

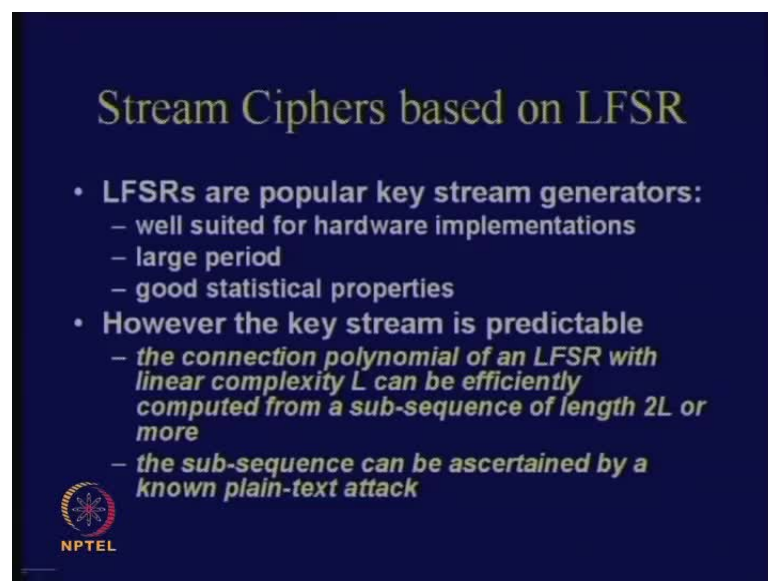
What is the meaning of that, the polynomial  $f$  should be a primitive polynomial and what field it has to be primitive then, the FSR that is  $R_2$  with feedback function of  $g$  which operates upon the same input variables and if you define  $g$  to be in this fashion. It is nothing but,  $f$  xored with the naught of  $s_{j-1}$  anded with this is not comma this is and anded with  $s_{j-2}$  anded with so on,  $s_{j-L+1}$ . If you just leave the last one out and you just take this and and you xore then, what do you obtain is a de bruijn FSR.

Can you understand why? You note just one fact, I leave it to you that to think and rest of things just work out a small example and check that if all the values. When does this xore come into play, except for all the 0 value that is all these values are 0 this is nothing but, the same feedback. Therefore, you get back f only when this all all of them becomes equal to 0 then you are essentially forcing back 1, if you remember, in case of a maximal length LFSR our problem was that all the 0 state essentially, was feeding back 0 but, in this case you can observe, that it will feedback a 1 and it will get back into the other thing.

This is nothing but simple, you can state state or pose this problem as a simple switching theory problem. Therefore, it is just how to how to get out of the wrong state so we have done simple examples like that previously also, but, this particular problem if you solve, what you obtain is a de bruijn FSR, because all the 2 power L values are now lying in one cycle and what is the difference from a maximal length LFSR this particular polynomial or this particular feedback equation is no more linear. It is actually a non-linear thing.


So, you have solved the non-linearity problem at the same time we have ensured their periodicity is also quite high. You can just take one small example and work it out.

(Refer Slide Time: 13:52)



**Stream Ciphers based on LFSR**

- **LFSRs are popular key stream generators:**
  - well suited for hardware implementations
  - large period
  - good statistical properties
- **However the key stream is predictable**
  - *the connection polynomial of an LFSR with linear complexity  $L$  can be efficiently computed from a sub-sequence of length  $2L$  or more*
  - *the sub-sequence can be ascertained by a known plain-text attack*

 NPTEL

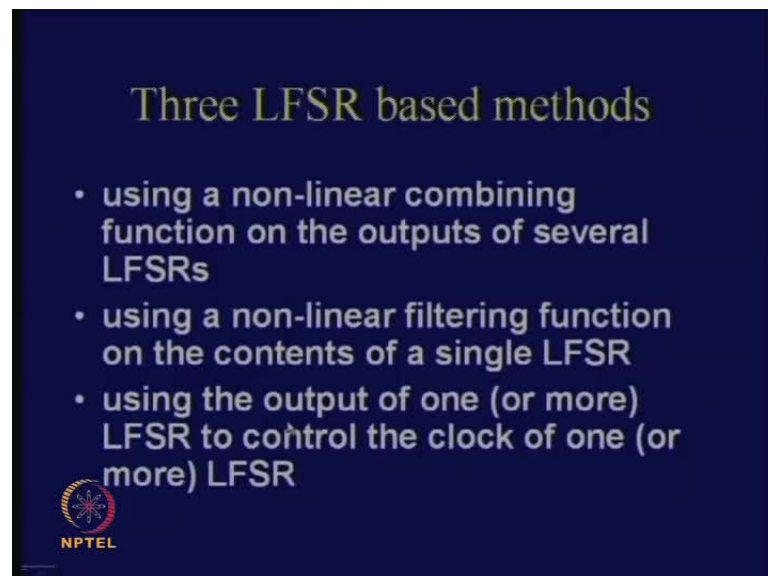
Now, we will consider stream ciphers which are based on LFSRs. The last one, the last example that we considered was not actually based on LFSR. It was a feedback shift

register which was not linear but, now we will consider stream ciphers which are based on LFSRs. Why do we like LFSRs, because they are well suited for hardware implementations? They have got large period, they have got good statistical properties and however the key stream is quite predictable. We have seen the connection polynomial of an LFSR with linear complexity  $L$  can efficiently be computed from a sub-sequence of length  $2L$  or more.

If you give a sub-sequence of length  $2L$  then, you can actually compute the corresponding connection polynomial. We can also compute the linear complexity  $L$  by using the algorithm berlekampmassey algorithm the complexity of the algorithm is quadratic.

(Refer Slide Time: 13:52) The subsequence, we thought of this point that the sub-sequence therefore, in order to recover we require the sub-sequence and how can you obtain the sub-sequence you can ascertain that by a known plain-text kind of attack. If you know the plain-text, if you know the corresponding cipher stream then you can just xore them and you can obtain the keys stream and if you obtain I mean say at least  $2L$  number of key stream values from there you can regenerate or reconstruct the LFSR.

(Refer Slide Time: 15:31)



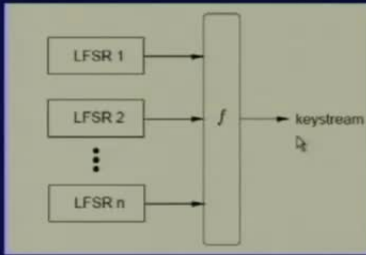
We require more than one LFSR systems stream ciphers and this is how you will just consider three types of such cases, we will consider using a non-linear combining function on the outputs of several LFSRs suppose, there is not one LFSR but, there are



more than one LFSRs and all of them are producing a key stream so, what we will try to do is that, we will try to combine them using a non-linear combining function. This is one example, one possible way, the other thing what we can do is, we can use a non-linear filtering function on the contents of a single LFSR so I will explain more details of this and also what we can use is that we use the output of one or more LFSR to control the clock of one or more LFSR. I will explain these possible constructions.

(Refer Slide Time: 16:19)

**Non-linear combination generators**



The diagram shows a block labeled 'f' with multiple input arrows from boxes labeled 'LFSR 1', 'LFSR 2', and 'LFSR n'. A single output arrow from block 'f' is labeled 'keystream'.

- $f$  is a non-linear combining function
- Suppose that  $n$  maximum length LFSRs, whose lengths  $L_1, L_2, \dots, L_n$  are pairwise distinct and greater than 2, are combined by a non-linear function  $f(x_1, x_2, \dots, x_n)$ , which is the ANF form. Then the linear complexity of the key stream is  $f(L_1, L_2, \dots, L_n)$ , where the XORs are replaced by integer additions.

NPTEL

Let us consider this; suppose there are  $n$  LFSRs and all these LFSRs are producing a key stream. What we do next is that we take a function  $f$  and we combine all of them and produce a key stream note, that here  $f$  is a non-linear combining function, suppose there is a result which states like this, that  $n$  maximal length LFSRs all of them are maximal length LFSRs and then lengths of all of them are say  $L_1, L_2$ , and so on till  $L_n$  where,  $L_1$  to  $L_n$  are all mutually co-prime.

What does mutually co-prime, mean and also they are pair wise distinct. You can follow from there, and if they are greater than two that means if if you consider fairly large LFSRs for example, then you can actually predict if the corresponding non-linear function  $f$  has got ANF form, ANF form means an algebraic normal form. Suppose the form of  $f$  is  $f = x_1 \oplus x_2 \oplus \dots \oplus x_n$  and so on till  $x_n$  so this is the ANF form. Then you can predict the corresponding linear complexity of this particular key stream. What you do is that you take this  $L_1$  to  $L_n$  and you feed in this function  $f$  and what you do is that you replace all

the xores by simple integer additions. If you do that you can actually compute the corresponding linear complexity of the key stream.

(Refer Slide Time: 18:02)

### Example: the Geffe generator

- 3 maximum length LFSRs whose lengths  $L_1$ ,  $L_2$  and  $L_3$  are pairwise relatively prime.
- $\text{Period} = (2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$
- $\text{Linear Complexity} = L_1 L_2 + L_2 L_3 + L_3$

NPTEL

Let us, consider one simple example this is something which is called the geffe generator. There are three LFSRs here and this is producing one output, this is producing one output, this is producing one output and this is the key stream therefore, you can see that this is particular if, I box this one then this is my  $f$  and what is the form of  $f$ ? You take  $x_1$  you take  $x_2$  add them so it is  $x_1 x_2$  xored with  $x_3 x_2$  bar.

(Refer Slide Time: 18:43)

$$\begin{aligned}
 f(x_1, x_2, x_3) &= x_1 x_2 \oplus \bar{x}_2 x_3 \\
 &= x_1 x_2 \oplus (1 \oplus x_2) x_3 \\
 &= x_1 x_2 \oplus x_2 x_3 \oplus x_3.
 \end{aligned}$$

ANF.

LFSR 1  $\rightarrow L_1$   
 2  $\rightarrow L_2$   
 3  $\rightarrow L_3$

$$L = L_1 L_2 + L_2 L_3 + L_3.$$

NPTEL

The form would have been like this, you take  $x_1$ ,  $x_2$  and you XOR that with  $\bar{x}_2$ ,  $x_3$ . This would have been my form. Therefore, this is my  $f(x_1, x_2)$  and  $x_3$  so I can write  $x_1$ ,  $x_2$  and I can write XOR and I can write  $1 \oplus x_2$ ,  $x_3$  and this is equal to  $x_1 \oplus x_2 \oplus (1 \oplus x_2) \oplus x_3$  and this is expressed in which form in ANF form.

Now, if the linear complexity of your LFSR 1 is  $L_1$  that of 2 is  $L_2$  and that of 3 is  $L_3$  and if all of them are not equal to each other. They are greater than two, then the corresponding linear complexity of the corresponding Geffe generator stream would have been  $L$  equal to  $L_1, L_2$  plus  $L_2, L_3$  plus  $L_3$  here, you note that these plus are integer additions and you are doing this operations over integers and the periodicity of this particular Geffe generator also would have been equal to  $2$  to the power of  $L_1 - 1$  multiplied by  $2$  to the power of  $L_2 - 1$  into  $2$  to the power of  $L_3 - 1$ . So these are guarantees that we can actually produce.

(Refer Slide Time: 20:30)


**Correlation Attacks**

- The Geffe generator is cryptographically weak, because information about the states of LFSR 1 and LFSR 3 leaks into the output sequence.

$$\Pr(z(t)=x_1(t)) = \Pr(x_2(t)=1) + \Pr(x_2(t)=0)\Pr(x_3(t)=x_1(t))$$

$$= \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$$

Similarly,  $\Pr(z(t)=x_3(t)) = \frac{3}{4}$



This Geffe generator is actually weak, why it is weak? Well, you see that all though it has got a good linear complexity and at the same time it has also got a reasonably good periodicity. I mean high value of periodicity but still it is weak, because of something which is called correlation attacks.

Why correlation attacks? Let us try to understand that, if you remember in your equation, your equation was  $x_1 \oplus x_2 \oplus \bar{x}_2 \oplus x_3$  and XOR with  $x_3$  so, you see that the Geffe generator is actually cryptographically weak, because there is an information

leakage of the output stream which is being generated and the value of LFSR 1 for example, LFSR 1 generates the stream which had note by  $x_1$  and the output stream is suppose  $z$ ,  $z_t$  means at  $t$  th instance suppose  $z$  is being generated. Now let us consider, the probability or compute the probability that  $z_t$  is actually equal to  $x_1$ .

(Refer Slide Time: 21:52)

$$\begin{aligned}
 f(x_1, x_2, x_3) &= x_1x_2 \oplus \bar{x}_2x_3 \\
 &= x_1x_2 \oplus (1 \oplus x_2)x_3 \\
 &= x_1x_2 \oplus x_2x_3 \oplus x_3.
 \end{aligned}$$

ANF.

LFSR 1  $\rightarrow$   $L_1$   
 2  $\rightarrow$   $L_2$   
 3  $\rightarrow$   $L_3$

$L = L_1L_2$

This you can follow from the equation you see that, if  $x_2$  is equal to 1 this is always true. In your equation if  $x_2$  is equal to 1 then, it is always true because, you get  $x_1$  xored with  $x_1$  xored with  $x_3$  xored with  $x_3$  so you see that  $x_3$  and  $x_3$  gets cancelled and you have  $x_1$  but, what if your  $x_1$  is  $x_2$  is equal to 0, if  $x_2$  is equal to 0 then, this vanishes, this vanishes and only  $x_3$  remains.

(Refer Slide Time: 20:30)

Therefore, you get the internal result if  $x_3$  is equal to  $x_1$ , you can write this equation that is probability  $z_t$  is equal to  $x_1$  is equal to probability that  $x_2$  is equal to 1, and if  $x_2$  is equal to 0 then  $x_3$  is equal to  $x_1$  and what is the probability of this? if you assume that, your inputs are independently and randomly chosen then, your  $x_2$  is equal to 1 has a probability of half,  $x_2$  equal to 0 has a probability of half, and  $x_3$  is equal to  $x_1$  is also half therefore, this probability works out to 3 by 4.

Similarly, you can show that  $z_t$  is equal to  $x_{3t}$  is also equal to  $3 \cdot 4$ . If you really want to generate a very strong cipher an ideal value of this should have been equal to half but, here it shows that there is a deviation from half that is, it is half plus  $1/4$ .


(Refer Slide Time: 20:30) This deviation can be exploited by certain type of attacks which are called correlation attacks so, how does correlation attacks work? what you do is that, you assume for example, if you just considered this single 3 LFSR system; what you do is that, you assume the input value of; for example, LFSR 1 and then you consider the corresponding, you know the key stream which is being produced by the geffe generator and then we assume these particular seed and compute the output of your LFSR 1, you find the number of co-incidences between this value and the corresponding key stream then check whether, it matches with the probability of  $3/4$  then, instead of taking the corresponding output of the LFSR 1, you consider a shift of that, may be a shift of one step. Now check the co-incidences.

How many possible shifts are possible, that depends upon its periodicity. If its periodicity is  $2^{L-1}$ , you can half at most  $2^{L-1}$  such attempts, out of at least one you will get, where this particular thing holds. You can repeat these effort for the other LFSRs also. So, if you engage an effort of  $2^{L-1} + 2^{L-2} + 2^{L-3} + \dots + 2^0$  because, all of them are actually independent. Then, you should be able to recover the corresponding seed but, what was your ideal size of the seed for this geffe generator. It was actually equal to  $2^{L-1} \cdot 2^{L-2} \cdot 2^{L-3} \cdot \dots \cdot 2^0$ , so you see that all though you have got a very large key size expected, you can actually reduce that to a much smaller value and therefore, this geffe generator is actually not a very strong stream cipher.

(Refer Slide Time: 25:44)

## Correlation attacks

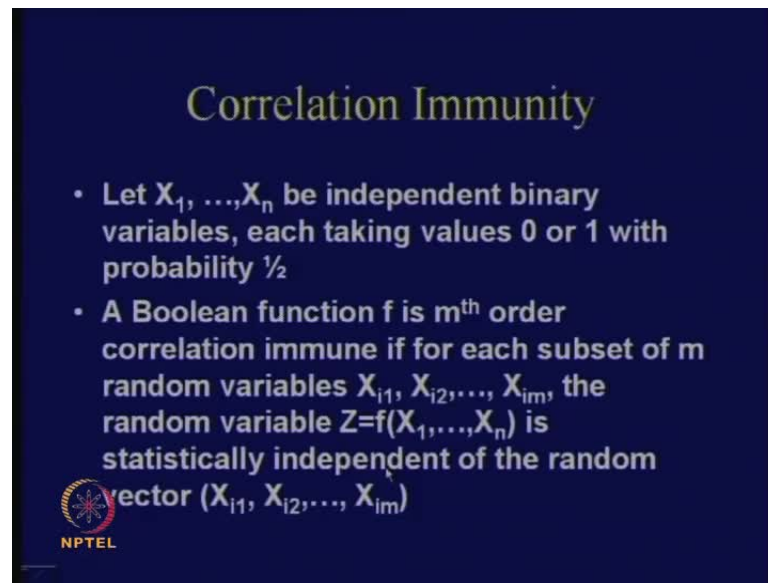
- Consider  $n$  maximum length LFSRs  $R_1, R_2, \dots, R_n$  with lengths  $L_1, L_2, \dots, L_n$
- Number of keys =  $(2^{L_1}-1)(2^{L_2}-1)\dots(2^{L_n}-1)$
- Suppose that there is a correlation between the keystream and the output of  $R_1$  with probability  $p > 1/2$ 
  - guess the initial state of  $R_1$
  - Compute the number of coincidences between the keystream and all possible shifts of the output sequence of  $R_1$ , until the probability is more than  $p$ .
  - Number of trials =  $(2^{L_1}-1)$
  - Since the initial states of the LFSRs can be known independently, total number of trials =  $\sum (2^{L_i}-1)$



If you consider this example, this is generalization so if you consider that  $n$  maximal LFSRs are there  $R_1$  to  $R_n$  and all of them have got lengths say  $L_1$  to  $L_n$  then, a number of keys should be actually  $2^{L_1-1}$  multiplied by  $2^{L_2-1}$  so onto  $2^{L_n-1}$  but, suppose you have got a correlation between the keystream and suppose the output of  $R_1$  with the probability of  $p$  which is significantly greater than half.


What you do is that, you guess the initial state of  $R_1$ , compute the number of coincidences between the keystream and all possible shifts of the output sequence of  $R_1$  until, you find that your probability is more than  $p$  therefore, number of trials required would be at most  $2^{L_1-1}$ , and since the initial states of the LFSRs can be known independently, total number of such trials will be  $\sum 2^{L_i-1}$ , where  $i$  runs for all the LFSRs. This particular value is significantly smaller than this product term and you have an attack therefore, what we would like is to make the boolean function something, which is called correlation immune and that gives you the definition of correlation immunity.

(Refer Slide Time: 27:08)



**Correlation Immunity**

- Let  $X_1, \dots, X_n$  be independent binary variables, each taking values 0 or 1 with probability  $\frac{1}{2}$
- A Boolean function  $f$  is  $m^{\text{th}}$  order correlation immune if for each subset of  $m$  random variables  $X_{i_1}, X_{i_2}, \dots, X_{i_m}$ , the random variable  $Z=f(X_1, \dots, X_n)$  is statistically independent of the random vector  $(X_{i_1}, X_{i_2}, \dots, X_{i_m})$

 NPTEL

What is correlation immunity? Correlation immunity is again a criteria like, we have seen non-linearity, we have seen some other things like balanceness degree. Now, we have another parameter which we also need to keep in mind and it is called correlation immunity. Correlation immunity says; if suppose, you have got  $X_1$  to  $X_n$  as your independent binary variables. Consider for example, about three LFSRs system, how many inputs are there for your  $f$  function there are three inputs. There are three LFSRs all of them are producing some output.

If you understand that why the attack worked? The attack worked because, the output of the LFSR 1 was actually leaking some information into the output stream so, if I want a correlation immunity. What I would like to do is, the mutual information between the output of the LFSR 1 and the output stream should be actually 0 that is the objective and how we can do that? If you understand, (Refer Slide Time: 27:08) first of all let us study the definition, It says that let  $X_1$  to  $X_n$  be independent binary variables each taking values say 0 or 1 with probability of half then, a boolean function  $f$  is actually called  $m$  th order correlation immune. If for each subset of  $m$  random variables that is  $X_{i_1}, X_{i_2}$  to  $X_{i_m}$ , the random variable which you obtain  $Z$  is actually statistically independent of the random vector  $X_{i_1}$  to  $X_{i_m}$ .

This is what I intend to do therefore, I can state this in fact, that the mutual information of this vector and the corresponding  $Z$  random variable should be equal to 0. If you

remember, the definition of mutual information which we have done in our Shannon's theory class, then you can understand how to do that.

(Refer Slide Time: 27:08) One way of doing this is or one way of ensuring this is, if you take any boolean function  $f$  and out of them  $X_1$  to  $X_m$  are some random variables so what I would like to do is, I have to give a proof that if I want an  $m$ th order correlation immunity for  $f$  then, I have to give a proof that if I fix  $X_1$  to  $X_m$  to some constant value then, the remaining function which are obtained as  $f$  should still be balanced. If it is balanced then, it is not leaking any information and if it is not balanced then, it is leaking some information.

(Refer Slide Time: 30:28)

$$f(x_1, x_2, x_3) = x_1x_2 \oplus \bar{x}_2x_3$$

$$= x_1x_2 \oplus (1 \oplus x_2)x_3$$

$$= x_1x_2 \oplus x_2x_3 \oplus x_3$$

ANF.

LFSR 1  $\rightarrow L_1$   
 2  $\rightarrow L_2$   
 3  $\rightarrow L_3$

$$L = L_1L_2 + L_2L_3 + L_3$$

$$f(x_1, x_2, x_3) = x_2 \mid_{x_1=1}$$

$$f(x_1, x_2, x_3) = x_3 \mid_{x_1=0}$$

$$f \mid_{x_1=1} = x_2 \oplus x_2x_3 \oplus x_3$$

$$f \mid_{x_1=0} = x_2x_3 \oplus x_3$$

$$= x_3x_2$$

If you remember your boolean equation,  $x_1 \oplus x_2 \oplus x_2 \oplus x_3 \oplus x_3$  which we had. In that case, if you follow that is; if in the equation say  $x_1 \oplus x_2 \oplus x_2 \oplus x_3 \oplus x_3$  with  $x_2 \oplus x_3 \oplus x_3$ . In this equation we had held the value of say  $x_1$  to a constant value then, what is the remaining thing I get. If for example, I will make  $x_1$  equal to 1 or if I make  $x_2$  is equal to 1. I get  $f(x_1, x_2, x_3)$  and  $x_3$  is equal to  $x_1$ .

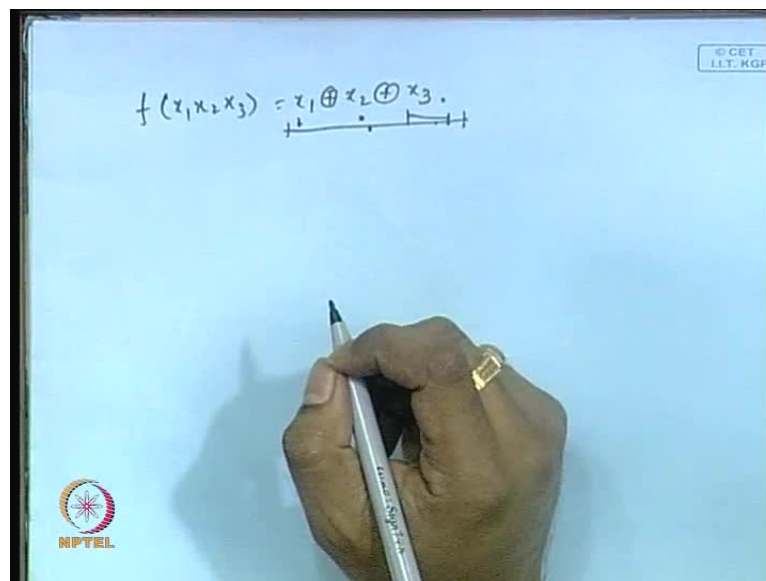
You see that  $x_1$  is a balanced equation. Why balanced? Because, if I obtain a truth table then, there are half number of equal number of zeros and ones. Now you see that actually you have to do this process for all possible things therefore, all possible subsets we have considered only  $x_2$  being equal to 1, what if  $x_2$  is equal to 0, you see that  $x_2$  is equal to 0. We have considered  $x_2$  is equal to 0 this was for  $x_2$  is equal to 0. What if  $x_2$  is equal



to 1? You obtain  $f$  of  $x_1 x_2$  and  $x_3$  is equal to, if  $x_2$  is 1, you obtain  $x_3$ . It is still balanced.

Now, you consider that if for example,  $x_1$  is equal to 1 so, if your  $f$  is; if I consider that  $x_1$  is equal to 1 then, what do you obtain? You obtain  $x_2$  xored with  $x_2$ ,  $x_3$  xored with  $x_3$ . Consider other cases also  $x_1$  equal to 0. What you get if  $x_1$  is equal to 0? You get  $x_1$  equal to 0,  $x_2$ ,  $x_3$  xored with  $x_3$ . What about this, you can write this as  $x_3$ ,  $x_2$  bar. This is no more a balanced function. This was the balanced function or was it? I do not think, this one was also not balanced. So, having only one and term does not mean that it will be actually unbalanced. (Refer Slide Time: 30:28) For example, if you consider the function of this type that is  $x_1$  xored with  $x_2$ ,  $x_3$  then, this is actually a balanced function. In this particular definition, it is actually an unbalanced function. Since, this is an unbalanced function you see that this particular function  $f$  is not correlation immune of one.

(Refer Slide Time: 33:40)

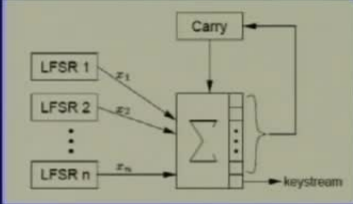


If you consider a simple example or simpler example: that is  $f x_1 x_2 x_3$  is equal to  $x_1$  xored  $x_2$  xored with  $x_3$  then, this is actually correlation immune of order two why because, if you hold any two values here, you will find that you will obtain a balanced equation. See for example, you fix  $x_1$  you fix  $x_2$ , you have got  $x_3$ . You fix  $x_3$  and fix  $x_1$  you obtain  $x_2$ .


If you have a linear equation of  $n$  variables then, it is actually correlation immune of  $n$  minus 1, order  $n$  minus 1 but, for non-linear equations; it is not so simple. You have to do that thing therefore, you see that if you really want that your function  $f$  should be able to protect against correlation immunity I mean, correlation attacks then, you have to guarantee that function  $f$  is actually correlation immune for order  $m$ , this guarantee you have to produce.

(Refer Slide Time: 34:42)

### Summation Generator



- The lengths  $L_1, L_2, \dots, L_n$  of the  $n$  LFSRs are pairwise prime.
- Period of the key-stream =  $\prod (2^{L_i} - 1)$ , while its linear complexity is close to this number.

 NPTEL

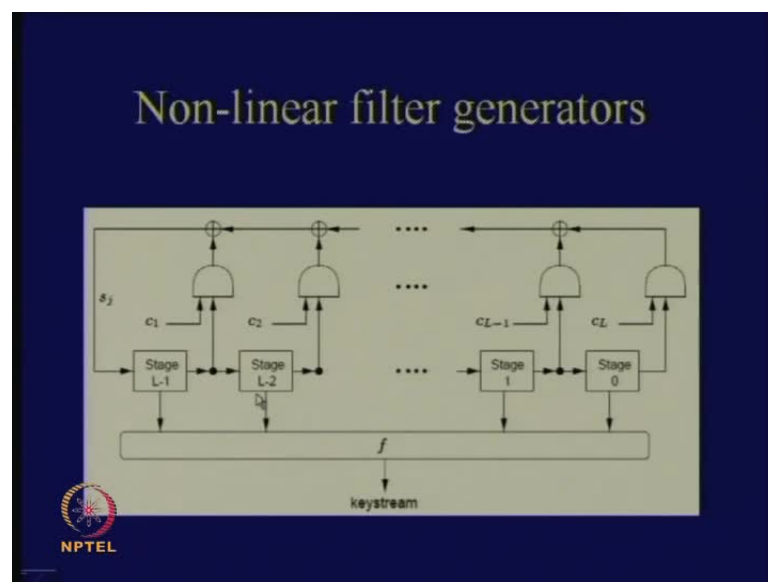
This is an example that, we will follow of something which is called as Summation Generator. Though, this is an example I mean real life example, you see that there are  $n$  LFSRs and what you do is that these streams that you obtain, you obtain a simple integer sum therefore, the integer sum will essentially produce a vector value of all these. If you add all these values, all of them are ones and zeros so, you just do an integer addition and the lower value you produce as a keystream. So, this is the LSB of your sum and the remaining things you essentially feedback to your carry, and this carry serves as a memory to your generator. In your next stage you also add this carry along with input you start with 0 and after that you keep on feeding back values into the carry.

There are rules which says that, if the lengths  $L_1$  to  $L_n$  of the  $n$  LFSRs are actually pairwise prime then, your period of the keystream will be product of  $2$  power  $L_i$  minus 1 and its linear complexity will also be close to this number.

This is an example and therefore, there are more intricacies. We can go more deep into this but, I am not going to this. I just produce, I gave you as an example, and there are some weakness also but, I will not go into them. This essentially derives or is based upon the fact that your carry, when you are doing an integer addition your carry is actually a highly non-linear term.

(Refer Slide Time: 34:42) You can take this as an exercise that, for your  $I$  th stage of your integer addition, your carry is actually a boolean function of  $2L$  variables, and these  $2L$  variables you can just try to see that, what is the non-linearity? and what is the correlation immunity of the function?

(Refer Slide Time: 36:43)




Now, we go into second class of generators based on LFSRs we have got  $n$  LFSRs and you see that this is nothing but, a simple LFSR that we have seen. This output of LFSRs is the  $L$  stage LFSR and this outputs are being combined by a boolean function  $f$  and it produces a given keystream. In this case this  $f$  is non-linear but, your LFSR, that corresponding shift register is still based on a linear feedback. You are taking all these intermediate values and you are combining them by a function  $f$ , this is one way of solving this problem.

(Refer Slide Time: 37:27)

## Clock controlled generators

**Alternating Step Generator:**

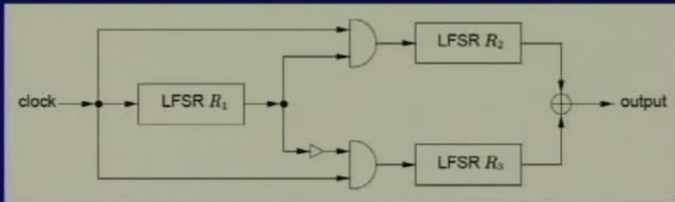
- A control LFSR  $R_1$  is used to selectively step two other LFSRs,  $R_2$  and  $R_3$ .
- Output sequence is the XOR of  $R_2$  and  $R_3$ .
- Algorithm:
  - Register  $R_1$  is clocked.
  - If output of  $R_1$  is 1, then  $R_2$  is clocked,  $R_3$  is not clocked but the previous output is repeated.
  - If output of  $R_1$  is 0, then  $R_3$  is clocked,  $R_2$  is not clocked but the previous output is repeated.




We will rather a little bit concentrate upon something, which is quite interesting, called as Clock controlled generators. There are two types of clock controlled generators, one: which is called an Alternating Step Generator. What is the construction principle? There are essentially three LFSRs here, say control LFSR  $R_1$  is used to selectively step two other LFSRs,  $R_2$  and  $R_3$ .

(Refer Slide Time: 38:03)

## Example



- If  $R_1$  produces a de Bruijn sequence, the alternating step generator has high period, high linear complexity, and have good statistical properties.



Output sequence is the xore of  $R_2$  and  $R_3$ . What you do is as follows, you take your register  $R_1$  therefore, this a diagram, just consider around this diagram you take the


register R 1 and you clock this. If this register R 1 produces a 1 then, you take the output of register R 2 therefore, you see that this register R 2 is clocked. If this produces 0 then, you are not clocking this register. What it produces here, is the previous output value and taking an xore of these two and producing in the producing output. If this value is 0 then, this one is not clocked but, this particular LFSR is clocked and when this is not clocked this gives you the previous value and you are taking the xore and you are producing the output.

This is the principle of something, which is called an Alternating clock generator and there are results which says that, if this particular R 1 produces a de bruijn sequence. You know, what is the de bruijn sequence therefore, if this R 1 produce a de bruijn sequence then, the alternating step generator satisfies good properties, that means it has got high period, it has got goodlinearity complexity and it is supposedly a very good stream cipher generator, key generator.

(Refer Slide Time: 39:21)

**Example (contd.)**

- $R_1 = \langle 3, 1 + D^2 + D^3 \rangle$ ,  $R_2 = \langle 4, 1 + D^3 + D^4 \rangle$ ,  
 $R_3 = \langle 5, 1 + D + D^3 + D^4 + D^5 \rangle$
- Suppose initial states of  $R_1 = [001]$ ,  
 $R_2 = [1011]$ ,  $R_3 = [01001]$
- Output sequences:
  - $R_1$ : 1001011
  - $R_2$ : 1101 0111 1000 100
  - $R_3$ : 1001 0101 1000 0111 0011 0111 1101 000
  - z: 1011 1010 1010 0001 0111 1011 0001 110...

 NPTEL

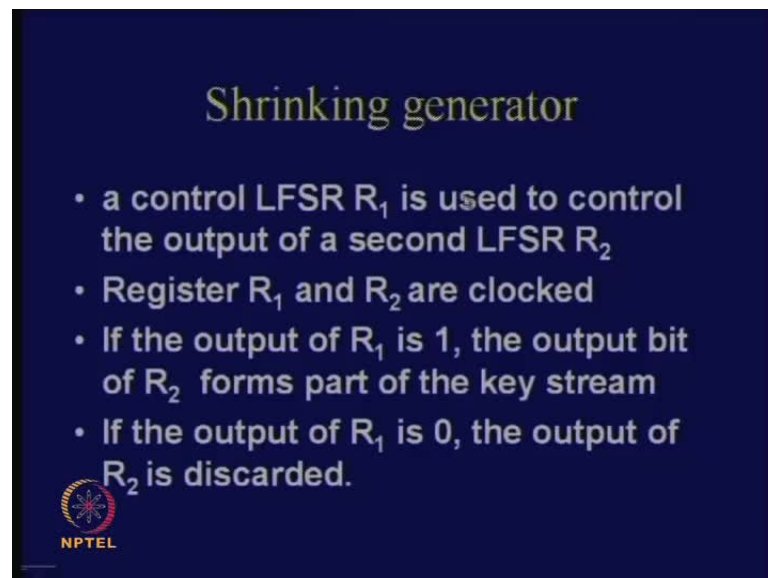
Let us consider, a concrete example of R 1 being equal to 3, 1 plus therefore, you know this nomenclature; this way of denoting this, is the length and this is the feedback polynomial. It is 1 plus D square plus D cube and similarly, R 2 and R 3 are as follows.

You should note that, R 2 and R 5, R 3. We have kept the feedback polynomial to be primitive so, in that case this produces R 2 is essentially a four bit LFSR. A maximal four bit LFSR means, there should be 15 periodicity therefore, you see that R 2 produces

a periodicity of 15,  $R_1$  produces a periodicity of 7. So,  $R_1$  a de bruijn generator? No, it is simply a maximal length LFSR and how can I produce, how can I obtain the output keys? This you can just concentrate or just think or reflect on this that, if this is the keystream which is being produced by a three bit maximal LFSR, how can I obtain the output of a three bit de bruijn generator?


I am not going into that therefore, you just think on this follows from the previous definition of de bruijn generator that we have discussed. If you take  $R_1$  and if you take  $R_2$  and you know that you can also take  $R_3$  then, this is how you obtain  $Z$  therefore, you see that if  $R_1$  produces a 1 then, what you are supposed to do is that, you are supposed to take  $R_2$  and  $R_3$  is initialized to 0. So, 1 xored with 0 will give you a 1, next you see that there is a 0, if you get a 0 then you clock this one and you keep this 1 same. This was initially 1 and you have got a 1 here. 1 xored with 1 is 0, so you obtain a 0 here similarly, you can obtain the keystream and you see that you have got a fairly large periodicity of the value. So, till now you do not see a periodicity here. This is a sequence being generated with a significantly large period

(Refer Slide Time: 41:42)



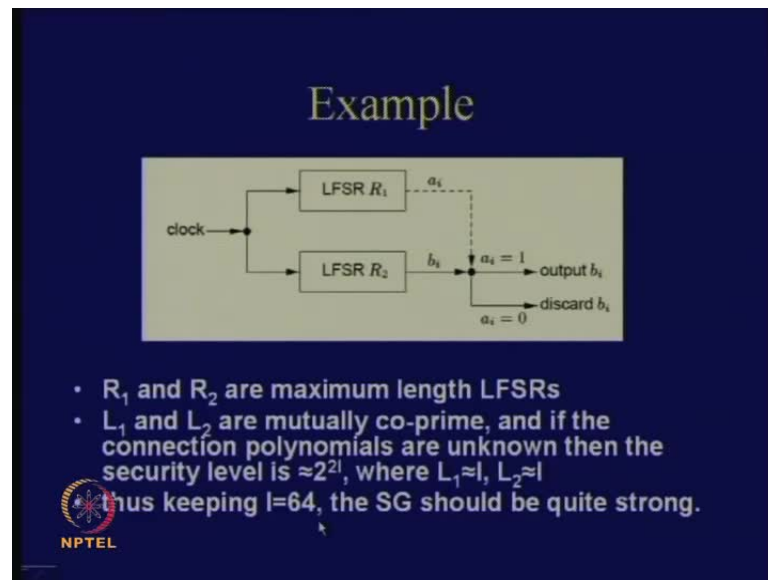
**Shrinking generator**

- a control LFSR  $R_1$  is used to control the output of a second LFSR  $R_2$
- Register  $R_1$  and  $R_2$  are clocked
- If the output of  $R_1$  is 1, the output bit of  $R_2$  forms part of the key stream
- If the output of  $R_1$  is 0, the output of  $R_2$  is discarded.

 NPTEL

Another class of generator of this class is something, which is called a Shrinking generator therefore, how it works is as follows you take a control LFSR  $R_1$ .

(Refer Slide Time: 41:54)




Again you consider this diagram, you take  $R_1$  and you take  $R_2$  and suppose that  $R_1$  produces a 1, if  $R_1$  produces a 1 then, you take the output of  $R_2$  into the sequence if  $R_1$  produces a 0, you drop this sequence. do you follow; if  $R_1$  produces a 1 you take this sequence, the output of we take  $b_i$  into your output, if  $R_1$  produces a 0, then you drop the value of  $b_i$  that means, you do not take it therefore, you have to consider the next value of  $R_1$  so you see that immediately the output sequence of  $b_i$  is being shrunked because, there are some values which you are dropping.

You can state this, in a different way that is; if you consider LFSR 2 and ensure that suppose  $R_2$  is the maximal length LFSR, you arbitrarily drop some values from  $R_2$  depending upon the output of another LFSR therefore, sometimes you drop and sometimes you do not drop and therefore, this is how the construction of a shrinking generator works. If there are results which says that if  $L_1$  and  $L_2$  are mutually co-prime and if the connection polynomials are unknown then, the security level is actually roughly equal to  $2^{L_1 + L_2}$ . If you ensure that  $L$  is equal 64 then, this shrinking generator should be quite strong because, you know that it should give you a guarantee of 128 bits and  $2^{128}$  is still quite high. We are not giving guarantees of unconditional security remember that we are giving guarantees of computational security only.

(Refer Slide Time: 43:39)

**Example (contd.)**

- $R_1 = \langle 3, 1 + D + D^3 \rangle$ ,  $R_2 = \langle 5, 1 + D^3 + D^5 \rangle$
- Suppose initial states of  $R_1 = [100]$ ,  
 $R_2 = [00101]$
- Output sequences:
  - $R_1$ : 0011101
  - $R_2$ : 1010 0001 0010 1100 1111 1000 1101 110
  - $x$ : 1000 0101 1111 1011 10...



This is another example, it says that  $R_1$  is equal to  $3, 1 + D + D^3$ ,  $R_2$  is this initial states are so on so, you can obtain this. Let us workout this example, you see that if  $R_1$  produces a 0 here, then you know that, what if  $R_1$  produces a 0 then, you are dropping this file. Next, you again obtain a 0 you are not taking anything, you get a 1 then, you are taking this therefore, this 1 comes in the output next, what you obtain is again a 1. You are taking this 0, this 0 is coming to the output. Next, you again obtain a one, you see that 1 comes this is a 0 so again 0 comes hereafter, that you again obtain a 0, which means you are dropping this value, next value and again you obtain a 1 so you are taking a next value into the output which is still 0 so, this is the way how you can obtain the output of a shrinking generator.



(Refer Slide Time: 44:39)



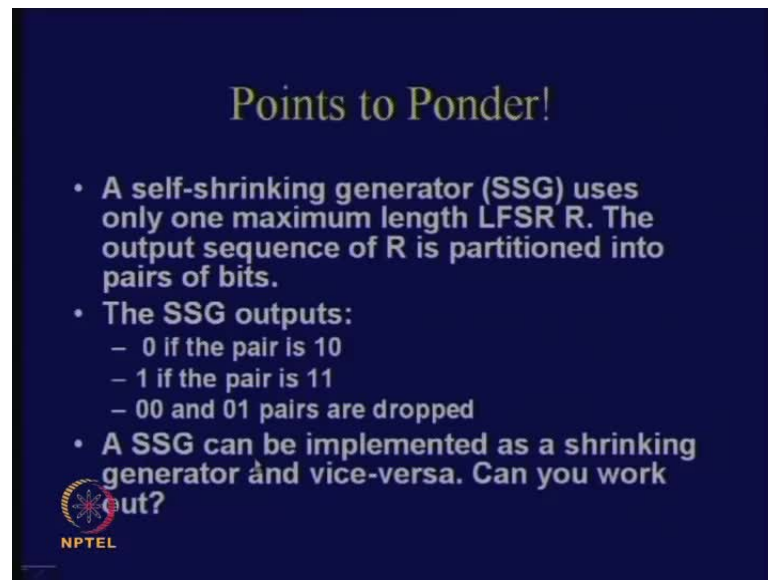
**Modern Stream Ciphers**

- **Several proposals in the Estream Website**
- **There are hardware and software candidates**
- **Search for standard Stream Ciphers**
- **New attack techniques have been developed, like algebraic attacks, cube attacks.**
- **Stream cipher design have become all the more challenging.**

  
NPTEL


I will just conclude with this slide, it says; that modern stream ciphers, for modern stream ciphers can actually based on LFSR systems, non LFSR systems also and there are several proposals that is actually a contest running on now a days which is supposedly trying to standardize some stream ciphers like, we have AES for block ciphers. There is a contest going on and there is something which is called an Estream Website and there are hardware and software candidates which are mentioned in the estream website. The search for standard stream ciphers continues while new attacks are also being developed, there are some attacks which are of called algebraic attacks, there are some attacks called cube attacks, which are quite powerful to analyze stream ciphers. There are some new tests also which are been evolved something which is called de monomial test and so on they are quite hard, enquire intricate to protect against and stream cipher designs have become all the more challenging.

(Refer Slide Time: 45:41)



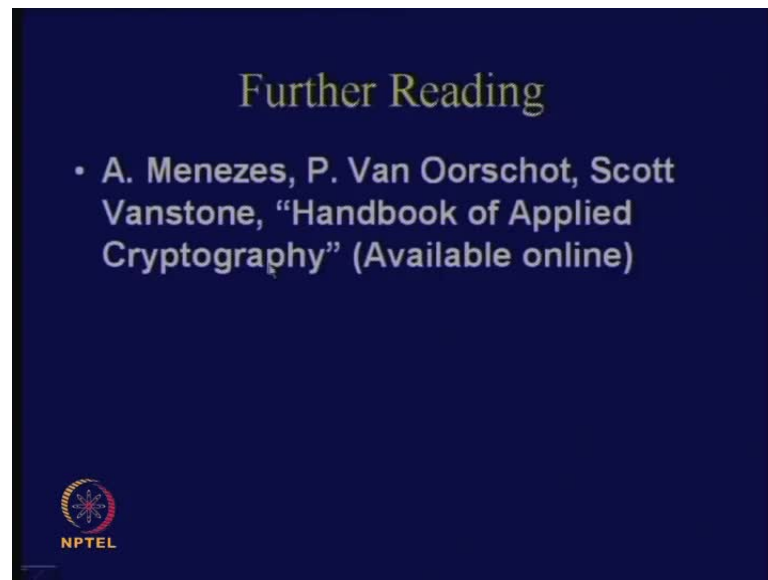
**Points to Ponder!**

- A self-shrinking generator (SSG) uses only one maximum length LFSR R. The output sequence of R is partitioned into pairs of bits.
- The SSG outputs:
  - 0 if the pair is 10
  - 1 if the pair is 11
  - 00 and 01 pairs are dropped
- A SSG can be implemented as a shrinking generator and vice-versa. Can you work out?

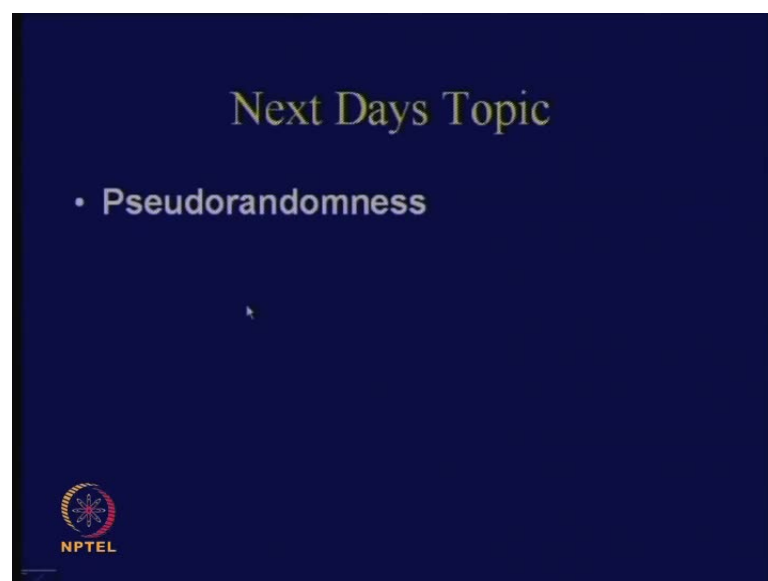
 NPTEL

So people are still searching for standard stream ciphers you can just ponder upon this point. There is something which is called as self-shrinking generator or SSG. What it does is as follows, it uses only one maximal length LFSR R, the output sequence of R is now partitioned into pairs of bits therefore, consider 1 single LFSR R and the output keystream you are partitioning into pairs depending upon whether, your pair is 1 0 or pair is 1 1, you are producing 0 or 1. If the pair is 0 0 and 0 1 then, you are dropping it. This SSG can actually be implemented as a shrinking generator and vice-versa that is, a self-shrinking generator you can implement using two LFSRs in the shrinking generator topology and also vice-versa. You can just try to work it out whether you can do it.

(Refer Slide Time: 46:33)



(Refer Slide Time: 46:47)



I have followed the menezes book menezes and oorschot and vanstone book on handbook of applied cryptography. It is available online, you can just go through this chapter and also chapter six so next day's topic will be pseudorandomness.