

**Hardware Security**  
**Prof. Debdeep Mukhopadhyay**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 34**  
**Power Analysis – X**

So, welcome to this class on Hardware Security. So, today we shall be trying to see how power attacks can be applied on knew on a primitive which is called as stream ciphers.

(Refer Slide Time: 00:27)



So, in particular today shall be we shall be talking about stream ciphers and a very important component in stream ciphers which are called as LFSRS or Linear Feedback Shift Registers we shall discuss about like the types of LFSRS which are broadly 2. And then discuss about the power attack vulnerabilities of LFSRS. We shall be then talking about MICKEY which is a standard stream cipher and shall be discussing about the DP attack on MICKEY or power attack being performed on the stream cipher called MICKEY.

(Refer Slide Time: 01:01)

Introduction to Stream Ciphers

- In cryptography, a stream cipher is a symmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (keystream).
- In a stream cipher each plaintext digit is encrypted one at a time with the corresponding digit of the key stream, to give a digit of the ciphertext stream.
- A stream cipher always runs faster and have lower hardware complexity
- Wide range of applications. For example, GSM mobile phones use A5/1 encryption

<https://www.youtube.com/watch?v=BKkpEjbD8xM&index=19&list=PL71FE85723FD414D7>

swayam  
INDIA WIDE EDUCATION

So, to introduce stream ciphers like block ciphers stream ciphers are also symmetric key ciphers, where the plaintext digits or bits in general are combined with the pseudorandom stream generator or a pseudorandom key generator which are also called as the key stream generators.

So, in a stream cipher the idea is that each plaintext digit is encrypted one at a time, with the corresponding digit of the key stream to give rise to a digit of the ciphertext. So, typically these when we say that they are you know like processed in digits what we mean is they can be bitwise processed or they can be processed in words as well.

So, they can be also you know like maybe processed in bytes or it may be in words. So, often the reason is because we want to increase the amount of throughput that we get out of the stream cipher. So, a stream cipher typically is a very fast primitive and has it has got also a lower hardware complexity compared to say block ciphers and therefore, it finds wide range of applications. For example, the GSM, mobile phones use A 5 1 encryptions and therefore, its very very popular primitive.

(Refer Slide Time: 02:21)

**Linear Feedback Shift Register (LFSR)**

- An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit.
- A LFSR usually incorporate a **feedback mechanism** to perform exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input.

If you are interested right to know more about stream ciphers, then I would ask you to kind of look into this YouTube clip where you can find more details about the stream ciphers. So, then what are linear feedback shift registers which are you know like. So, the idea is that the LFSR or Linear Feedback Shift Registers are a very important component of stream ciphers.

So, just to see how an LFSR works; an LFSR right as a name suggests is basically a shift cipher which when clocked advances the signal through the registers from 1 bit to the next most significant bit; that means, as you can see in this diagram, when I am essentially clocking it then you will find that this essentially shifts like this bit sequence shifts to the right for example, in this diagram.

But then right we need to find out what is the bit that I will feedback that because when the bits are going to the right, then I am getting the output from the right inside this is my key stream which is being generated which will be probably XORed with the message to give you the cipher ok. So, now, the question is right what how do you calculate the input bit? For example, what I mean to say is that this is a feedback; this is the feedback this is the feedback that we would like to we would like to we would like to obtain. So, this is the feedback that you are essentially that we need to calculate and this is output that is generated by these pre stream generator.

Often in you will find that this output right is XORed exclusive or with a message to give rise to what is called as a cipher or the cipher text. So, this part is called you know like is a very important component of the stream cipher often of course, the composition is not so, simple as this and may be more involved as you will see in when we discuss about the MICKEY.

So, now, the LFSR essentially right one of the most important things is how do you calculate this feedback for the LFSR. And as you can see here this is my feedback generation circuit for example, there are some XORs that has been used and the reason like the feedback only users XORs this essentially is the reason why it is called as LFSR or a linear feedback shift register because it has got only XORs which are linear gates.

So, therefore, right I mean the idea is that we would like basically you know like tap out few points from the LFSR states. So, this is your linear feedback shift registers state. So, from this LFSR right we basically tap out few points for example, as shown by the blue or the shaded portions. And then you do an operation which is defined by this feedback circuit and you create this feedback bit which is basically being fed that into the LFSR. For example, we are doing an XORed between 1 0 and 1. So, 1 0 0 and 1 and therefore, you get 0. So, therefore, we are feeding back 0 into the LFSR whereas, the other bits are basically just shifted to the right. So, that is the broad philosophy behind how an LFSR works.

(Refer Slide Time: 05:17)

Stream Ciphers

- LFSR (Linear Feedback Shift Registers) are used as building blocks for stream ciphers
- LFSRs are susceptible to power based SCAs
- An  $n$ -bit LFSR can be completely determined by making  $O(n)$  power measurements.
  - neither the primitive polynomial nor the value of  $n$  be known to the attacker.

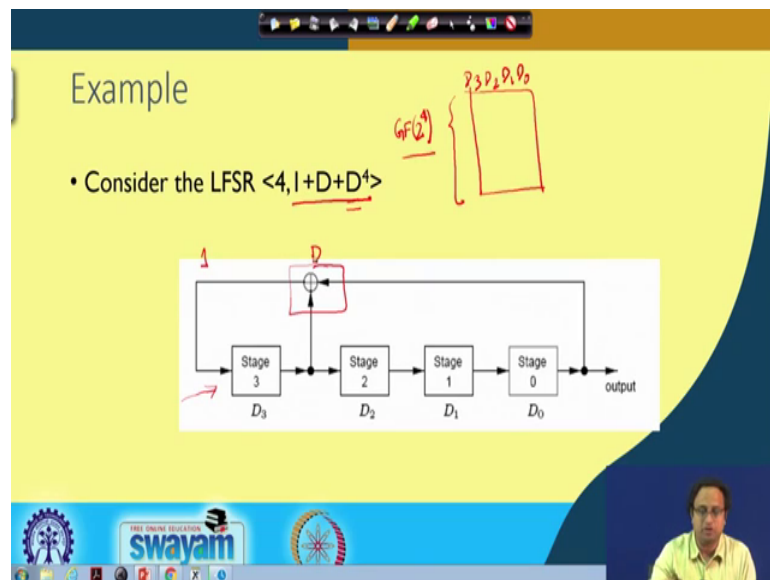
S. Burman, D. Mukhopadhyay, V. Kamakoti, "LFSR Based Stream Ciphers are Vulnerable to Power Attacks", INDOCRYPT 2007

swayam

So, now, idea is that LFSRs can be applied for stream ciphers and therefore, it is an important object to study when we are discussing or considering you know like how stream ciphers stand against power analysis or DP analysis. So, in this paper and this is an reference that you can see, which was published in INDOCRYPT 2007 you can see a paper which talks about LFSR based stream ciphers are vulnerable to power attacks.

So, you can look into this and see the there is a claim which we kind of discuss here which is that the which is essentially that the fact that LFSRS are susceptible to power based side channel attacks therefore, one can perform a power based attack on LFSRS. So, in particular the claim says that an  $n$  bit can be completely determined by making  $O(n)$  power measurement. So, if you make order  $n$  that is linear amount of power measurements, then you should be able to completely reconstruct the LFSR meaning the feedback polynomial which is often essentially a primitive polynomial can be completely calculated and in fact, you know like you can also calculate the number of bits which are there in the LFSR ok. So, you can calculate the number of stages which is essentially denoted by small  $n$  which essentially; that means, right you can totally deconstruct the LFSR.

(Refer Slide Time: 06:43)



So, let us consider an of an example to understand this ok. So, here is an LFSR which has got 4 stages as you can see like stage 0 stage 1 stage 2 and stage 3 and the feedback in this case is just determined by this exclusive or. So, which basically kind of steps in

stage 3 and XORed XORs with this bit to give back my feedback bit. So, you can see that this is the corresponding feedback bit that I obtain right the feedback bit is again here and that is basically obtained by this exclusive or so, this is my you know you know like feedback circuit in this case. And the polynomial is being has been described as 1 plus D plus D to the power of 4.

So, that basically essentially sets you the polynomial. So, the idea is you know like if for example, if there is a new polynomial that you want to read the typically write this is how we read it. So, the D to the power of 4 is basically the maximum bit or you know like since this is a 4 bit LFSR, you can imagine that the bit sequences will be which will be generate by this LFSR can be enumerated from by 4 bit. So, it can be enumerated by like 4 bits, depending upon the values of  $D^3$   $D^2$   $D^1$  and  $D^0$ . So, pretty much you can say that these elements are members of  $GF(2)$  to the power of 4.

So, now, as you know that for  $GF(2)$  to the power 4 right you can essentially set a corresponding polynomial say you know like for example, if I said 1 plus D plus D to the power of 4, then you can essentially obtain the elements of this field and in particular this feedback polynomial is you know written as 1 plus D to D plus  $D^2$  the power of 4 that is because of this position. So, therefore, the stands for 1 position, this position stands for D and gradually so and so.

So, basically right you basically can you know it depending upon the tap position the degree of the polynomial will get changed. So, in this case it is we basically get 1 plus D plus; 1 plus D plus D to the power of 4 which basically is the corresponding which basically tells me how the feedback circuit works. So, now, with this background let us see how the evolution of this LFSR works, when we are basically applying clocks to this linear feedback shift register.

(Refer Slide Time: 09:09)

Sequence of the LFSR

t	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	HD <sub>t</sub>	PD <sub>t</sub>
0	0	1	1	0	2	0
1	0	0	1	1	2	1
2	1	0	0	1	3	1
3	0	1	0	0	2	0
4	0	0	1	0	2	0
5	0	0	0	1	2	1
6	1	0	0	0	1	0
7	1	1	0	0	1	0

Handwritten notes on the slide:

- Red arrows indicate the feedback calculation:  $HD_t = D_3 \oplus D_0$ .
- Red arrows show the state transition:  $ST_{t+1} = (D_3, D_2, D_1, D_0)$ .
- Equation:  $HD_t = HD(ST_t)$ .
- Equation:  $ST_{t+1} = (ST_t)$ .
- Equation:  $ST_{t+2}$ .
- Handwritten binary sequence:  $0110$  and  $0011$ .
- Handwritten binary sequence:  $0101$ .

So, let us take this table. So, therefore, here is a you know like we basically loaded of course, one thing is to be kept in mind that when you are considering an LFSR like this and if you load all 0 values, then the value remain stuck at 0. So, therefore, we will be considering non zero initial states for example, if I load it like 0 1 1 0. So, this is an you know like a non zero initialization which I do to this polynomial into to this LFSR.

So, this LFSR because of the choice of the feedback polynomial, which is 1 plus D plus D to the power of 4 stands out to be something which is called as a maximum length LFSR which means it basically generates a cycle length which is proportional to 15 ok. Remember that the 4 0 state is not a part of the cycle because we are not giving it because its a self-loop ok.

Whereas, right if you fill in like 1 plus D plus D to the power of 4 is a feedback polynomial, then all the 15 non zero steps will lie in a cycle and that is the maximal length cycle which is possible and that is by this LFSR is also called as a maximal length LFSR. So, now, if I for example, initialize its a at 0 1 1 and 0, then you can see that I can calculate the corresponding you know like the feedback because you can observe that D 3 D 2 D 1 and D 0 are the corresponding states and therefore, when I am XORing then I am XORing this position with this D 0 position. So, basically I am XORing D 3 and D 0 to get back my feedback value ok.

So, in this case for example, if you know like  $D_3$  and  $D_0 = 0$ . So, therefore, I XORed them and I get 0 as a feedback. So, therefore, next time you can see that 0 is being feedback for example. So, what I mean is that when I am feeding when I am XORing  $D_3$  and  $D_0$  I am getting back this feedback value. So, therefore, if I XORed 0 with 0 I get back 0 if I XORed 0 with 1 then I get 1 as a feedback ok. So, you can see that here 0 has been feedback here 1 has been feedbacks likewise right if I XOR  $D_1$  with this 1 I get this 0 which is being feedback whereas, for the other positions it just shift. So, you can see that these are nothing, but essentially shifting of these values ok.

So, therefore, right if you do this then you can see that if I start with 0 1 1 0 and then you like proceed like this, then I these are the corresponding states that will essentially you like immerge one after the other and now we basically defined 2 variable. So, one variable is denoted as  $HD_t$  and the other variable is denoted as  $PD_t$ . So, what are the definitions of  $HD_t$  and  $PD_t$ ? So, in  $HD_t$  we basically calculate the hamming distance between these 2 states. So, you can see that these are the successive states and I calculate the hamming distance between them.

So, for example, what I mean is that, if one of these states here for example. So, this state suppose I denoted as  $ST_t$  suppose I denote it as  $ST_t$  and it migrates to  $ST_{t+1}$  and in the next clock cycle it goes to say  $ST_{t+2}$  then  $HD_t$  stands for the hamming distance between them. So, therefore, I can write that  $HD_t$  is nothing, but you know like the; you know like the hamming distance between. So, let me write this as a hamming distance between the states  $ST_t$  and  $ST_{t+1}$  ok. So, this the hamming distance between these 2 states ok.

So, what I mean is that if I now have got 0 1 1 1 0 here and the next state say  $ST_{t+1}$  is 0 0 1 1 then you can see that the hamming distance is nothing, but the changes for example, here there is 1 change and here there is 1 change. So, you can actually calculate this ah. So, see that in 1 state it is in the first clock cycle it was at 0 1 1 1 0 and these goes to 0 0 1 and 1. So, the hamming distance can also be found out in this way. So, if I take an exclusive or between these bits then I have got 0 XORed with 1 is 1 1 XORed with 1 is 0 1 XORed with 0 is 1 and 0 XORed with 0 is 0. So, the hamming distance you can say is also the hamming weight of the exclusive or between successive states.



So, therefore, write the hamming distance in this case is 2 ok. So, that is essentially written as 2 here likewise hamming distance between the clock cycle 1 state and the clock cycles 2 state is also 2, but you can see that between 2 and 3 there are if I you know XORed them I have got 1 here I have got 1 here and I have got 1 here. So, therefore, the hamming distance in this case is 3 ok.

So, now, I basically observe the hamming distances in this way. So, one of the motivations of looking into HD t is because as we discussed right among when we discussing about powered models 1 of the most common powered models in CMOS circuits are essentially what are based hamming distances. So, therefore, you can pretty much assume that this HD t is a good approximation of the power consumption of your device.

So, now what we try to see is that, we basically introduced another variable say call it as PD t and try to observe that whether the HD t changes across clock cycle. So, you can imagine that you like if you have basically the HD t is an indication of the power consumption. So, then it tells me that whether the power consumption increases across the clock cycle or kind of remains the same or diminishes across the clock cycle, because in some cases the power can also diminish as we can see that here the HD t is 1.

So, therefore, it means that what we are trying to see over here when we are seeing PD t is basically trying to see whether the power consumption kind of remain same or whether it becomes or whether it increases or whether it decreases.

So, the reason. So, what we plot PD t is just the fact you know like. So, we say the PD t is 0 if the power consumption remains the same and PD t is 1 if the power consumption changes; that means, right basically compared 2 successive HD t values, if the 2 successive HD t values are remaining same then we indicate that by PD t 0 and if they differ then we indicate it by PD t 1.

So, therefore, right you can see that since 2 and 2 are same I indicate as 0, 2 and 3 are different I indicate as 1, 3 and 2 are different I indicate as 1 2 and 2 are same I indicate as 0 2 and 2 are again you know like same. So, I indicate as 0 2 and 1 are different I indicate as 1 1 and 1 are so, I indicate it by 0. So, like this I continue because there will be more states in this evolution until you get back to this 0th state ok.

So, therefore, now we observe a very interesting observation. The interesting observation is this that if you observe  $PD_t$ , then this  $PD_t$  right gives you a binary sequence I shown here. This turns out to be nothing, but a delayed sequence of one of these states for example; in this case it kind of pretty much exactly matches with the  $D_0$  state ok. So, you can see that here I get 0 1 1 1 0 0, but I here I get for example, for  $D_0$  it is 0 1 1 0 0 and so on which exactly matches with  $PD_t$  ok.

So; that means, what is the implication of this? The implication is that if I have got a good power measurement, then I can you know like get an idea about  $HD_t$  based upon the power model and therefore, right if I get to know about  $HD_t$  and then try to find out a  $PD_t$  which basically a 0 1 decision to know that whether the power is increasing I mean whether the power is changing or whether the power is remaining constant across clock cycles, then I get 1 of the states of the LFSR.

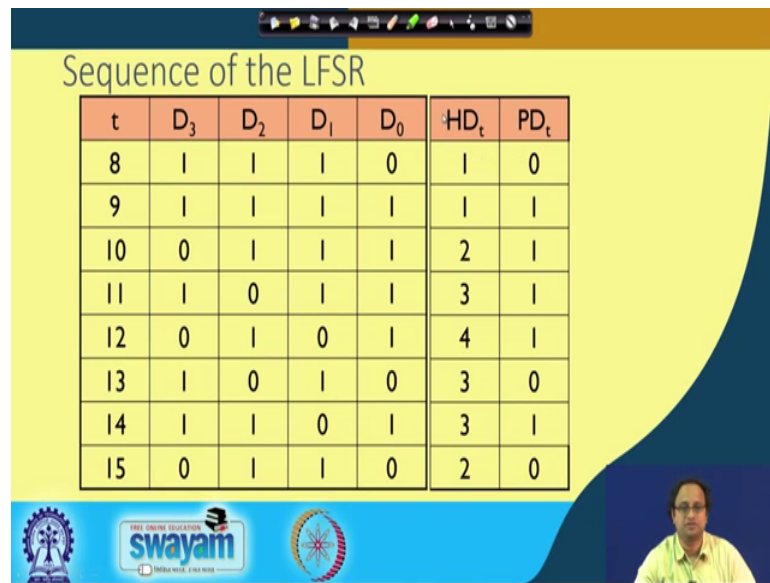
So, now, if you get the states of the LFSR ok. So, then you basically have you know like we will see very soon that you can actually reconstruct the entire LFSR. So, now, you can probably argue with me that you know like I already have access to  $D_3$  right because  $D_3$  is the output of or maybe  $D_0$  is the output of my key stream generator. So, what I am trying to say is this that is, you know like if I basically you know see these diagram then  $D_0$  is output which I am providing.

But note that the attacker may not always have access to  $D_0$ , because this maybe emerged in a big circuit it maybe emerged into stream cipher or this you know this point is not is not accessible to the adversary ok. But here what we observe is that from the power measurements and from this estimate of the  $PD_t$  vector, you basically know the values of  $D_0$  ok. And if you know the values of  $D_0$  then you can essentially reconstruct the entire LFSR ok.

(Refer Slide Time: 18:01)

Sequence of the LFSR

t	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	HD <sub>t</sub>	PD <sub>t</sub>
8	1	1	1	0	1	0
9	1	1	1	1	1	1
10	0	1	1	1	2	1
11	1	0	1	1	3	1
12	0	1	0	1	4	1
13	1	0	1	0	3	0
14	1	1	0	1	3	1
15	0	1	1	0	2	0



So, now, what, so, this is a continuation of this evolution as I said that you basically got in 15 and you can see that 15 is nothing, but 0 1 1 1 0 which means after this the cycle repeats ok. Again you can see that this 0 0 4 ones exactly matches with D 0. So, you can essentially prove as well, you can also go to the reference that I mentioned and see the complete proof where we essentially can argue that the spirit is indeed the delayed sequence of this LFSR states. (Refer Slide Time: 18:31)

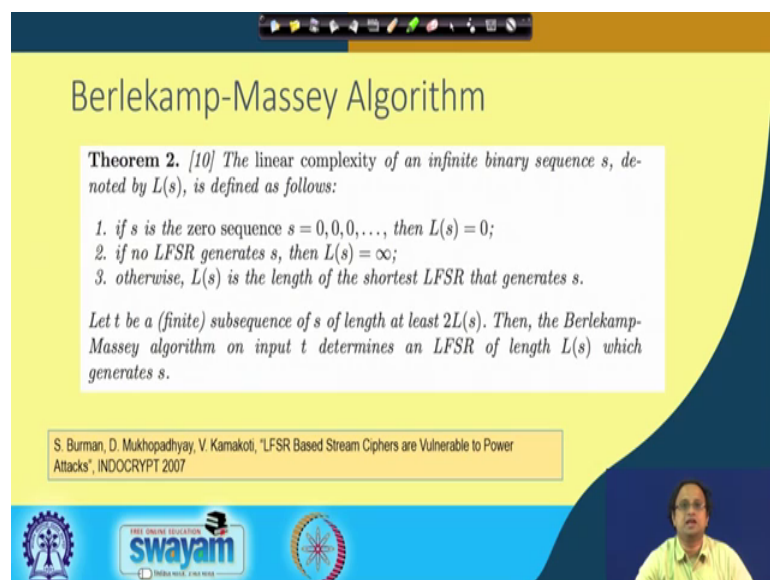
Berlekamp-Massey Algorithm

**Theorem 2.** [10] The linear complexity of an infinite binary sequence  $s$ , denoted by  $L(s)$ , is defined as follows:

1. if  $s$  is the zero sequence  $s = 0, 0, 0, \dots$ , then  $L(s) = 0$ ;
2. if no LFSR generates  $s$ , then  $L(s) = \infty$ ;
3. otherwise,  $L(s)$  is the length of the shortest LFSR that generates  $s$ .

Let  $t$  be a (finite) subsequence of  $s$  of length at least  $2L(s)$ . Then, the Berlekamp-Massey algorithm on input  $t$  determines an LFSR of length  $L(s)$  which generates  $s$ .

S. Burman, D. Mukhopadhyay, V. Kamakoti, "LFSR Based Stream Ciphers are Vulnerable to Power Attacks", INDOCRYPT 2007



So, now what we do is, we basically apply a result which is essentially called as the Berlekamp Massey Algorithm. So, this description of the Berlekamp Massey Algorithm kind of summarizes the power of these algorithm. We will not go into details of the

algorithm, but you essentially can you like at this point just concentrate on the implication of this algorithm.

So, the algorithm says that it basically defined something which is called as a linear complexity of an infinite binary sequence. So, which essentially is defined by  $L_s$  or denoted by  $L_s$  and is defined as follows. So, the idea is that if  $s$  is a 0 sequence something like  $s$  equal to 0 0 0 and so on then you can immediately then we define  $L_s$  to be 0 ok.

On the other hand if no LFS are generate  $s$  then I say  $L_s$  to be infinity which means that no linear no finite linear feedback shift register can generate the sequence ok. On the other hand right if it is otherwise; that means, if we have got if it is possible to generate the sequence then  $L_s$  stands for the length of the shortest or the minimal length LFSR that can generate this  $s$  ok. For example, in our case the sequence that we saw essentially was generated by a 4 bit LFSR ok; that means, like you somebody gives you this sequence the 0 0 1 1 0 0 0 0 1 0 0 followed by 0 four 1s and 0 1 0 1 0.

Then this sequence can be generated by an LFSR whose length is 4 and therefore our  $L_s$  is 4 because is the minimal length LFSR of length 4 which can actually generate the sequence. So, therefore, right I mean the idea is that if  $t$  is a finite sequence or subsequence of  $s$  of length at least  $2 L_s$  ok. So; that means, if I give you a sequence of length at least  $2 L_s$ , then the Berlekamp Massey algorithm on input of this  $t$  that is on input subsequence can completely generate this LFSR of length  $L_s$ , which essentially is can be used to generate  $s$  ok. So that means, you will as an attacker I you just needs an exposure to  $2 L_s$  amount of data to completely reconstruct the LFSR.

So, that means, in our case if is 4 then you just need a subsequence of length 8 and that should be enough for you to solve the entire equation the equation sets and kind of reconstruct the LFSR which means you.

(Refer Slide Time: 21:01)

The Attack

- Step 1: Measure  $Pow(0)$  (dynamic power) at  $t=0$
- Step 2: For  $t=k, k \geq 1$ :
  - Measure  $Pow(k)$  (dynamic power)
  - $PD'_{k-1} = 1$ , if  $Pow(k-1) \neq Pow(k)$ , else 0
  - Input  $PD'_{k-1}$  to Berlekamp-Massey (BM). If BM terminates then exit, else repeat Step 2.

[https://www.youtube.com/watch?v=xWzFuDtfX\\_M&list=PL71FE85723FD414D7&index=20](https://www.youtube.com/watch?v=xWzFuDtfX_M&list=PL71FE85723FD414D7&index=20)

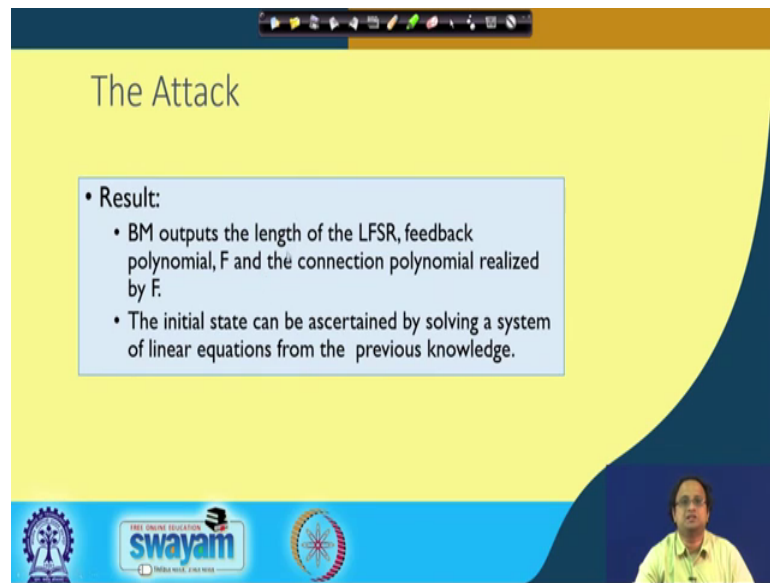
swamyam

So, these are very strong algorithm and if you are interested to know more about Berlekamp Massey algorithm explain you can go through this YouTube clip, which you can which can basically give you an explains the working principle of Berlekamp Massey algorithm. (Refer Time: 21:28) this course we will be using this Berlekamp Massey algorithm as a black box and the this how you can engage it. So, what I do in this attack is now I basically measure the power of the dynamic power at  $t$  equal to 0 and then for all the subsequent time instances I measure the dynamic power.

The idea is that if the power of 1 time instance is not equal to the previous time instance, then I define this  $PD$   $t$  dash variable and make it 1 ok. On the other hand right if this is 0 this that implicates that the power consumption does not change across the clock cycles. So, now, I basically take this  $PD$   $t$  dash and I as I said that based on the; based on the theory that the  $PD$   $t$  dash is nothing, but a delayed sequence of one of the states of the LFSR, then I just we will try to fill it to Berlekamp Massey algorithm ok.

Now the Massey at some point right the Berlekamp Massey algorithm will terminate and then I exit which means like I have been able to you know like give enough amount of data to the Berlekamp Massey algorithm. So, it can reconstruct the l the LFSR ok.

(Refer Slide Time: 22:29)



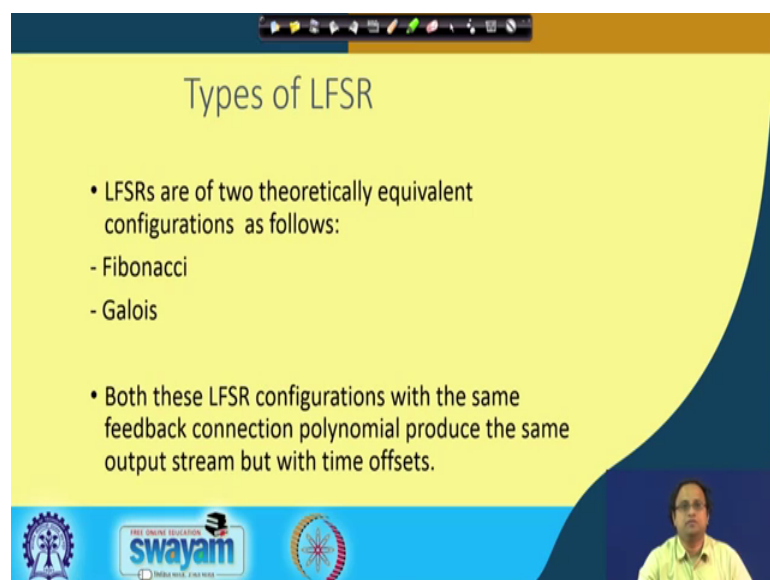
The Attack

- Result:
  - BM outputs the length of the LFSR, feedback polynomial,  $F$  and the connection polynomial realized by  $F$ .
  - The initial state can be ascertained by solving a system of linear equations from the previous knowledge.

swayam

So, therefore, the result says there are Berlekamp Massey algorithm outputs the length of the LFSR and also the feedback polynomial  $F$  and also the connection polynomial which is realized by  $F$ . And the initial state can be ascertained by solving a system of linear equations from the previous knowledge. Once you know the LFSR structure then guessing the input state is just solving a few solving a system of linear equations.

(Refer Slide Time: 22:57)



Types of LFSR

- LFSRs are of two theoretically equivalent configurations as follows:
  - Fibonacci
  - Galois
- Both these LFSR configurations with the same feedback connection polynomial produce the same output stream but with time offsets.

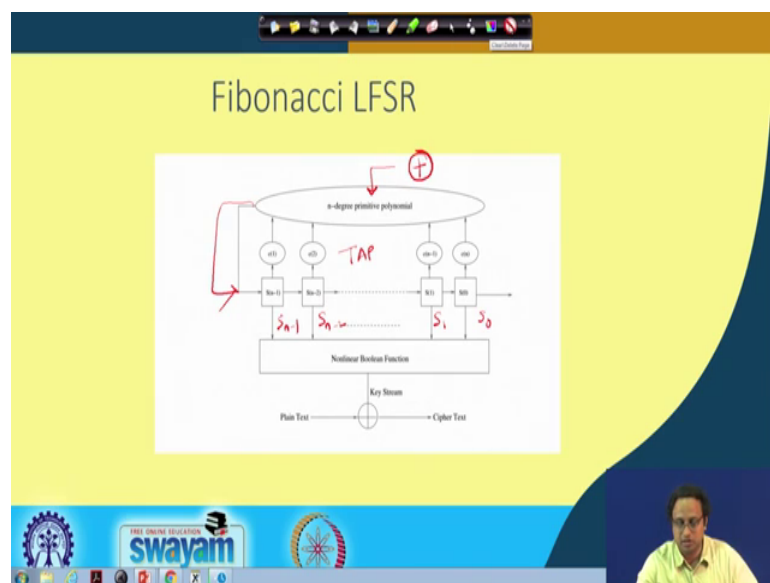
swayam

So, therefore, we can you know like try to kind of reflect on this points a little bit further, based on the fact that there are 2 types of linear feedback shift registers. So, one is one is

what we have just seen is what is called as a Fibonacci LFSR and there is another class of LFSR it is called as the Galois LFSR. So, theoretically both of these are isomorphic which means both these LFSR configurations with the same feedback connection polynomials produce the same output stream, but with time offsets ok.

So, now, you know like although from the cryptographic point of view both of these LFSRs are equivalent and you can use one instead of the other, the question is whether they are also equivalent with respect to power attacks ok.

(Refer Slide Time: 23:43)

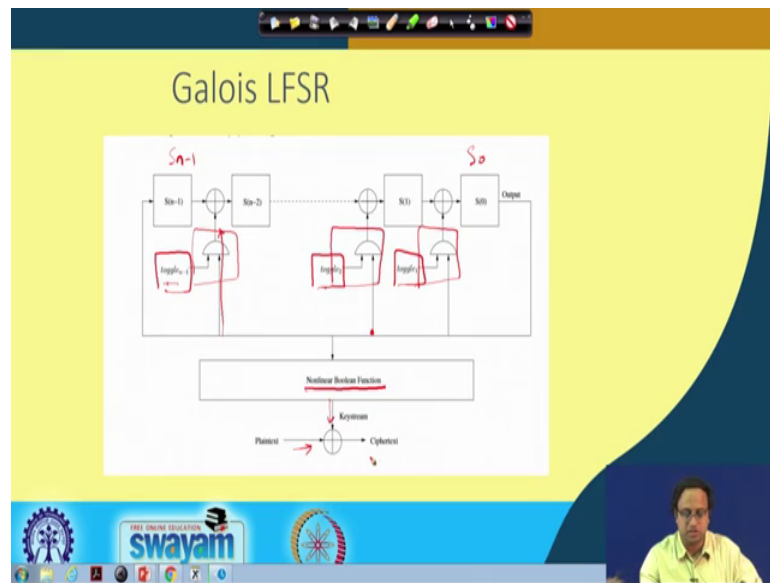


So, just to kind of revise these how a Fibonacci LFSR looks like and these are not general description of as we have seen like here you have got the  $n$  states of the LFSR. So, these denoted from you know like if I just write down right this is basically the  $S_0$  state this is the  $S_1$  state and so on the  $S_{n-2}$  states this is the  $S_{n-1}$  state.

And then what we do is basically, we try to kind of observe the feedback polynomial. So, the feedback polynomial is nothing, but as we have seen right is an XORed is best based on XORed. So, therefore, the only gates which are here are XORed, but then you basically take you basically tap out few positions and these are called as taps and then you calculate this the output of the circuit and you feed it back. So, this is your feedback.

So, this particular configuration is what is called as the Fibonacci style of LFSRS and this is oppose to what is called as the Galois LFSR ok.

(Refer Slide Time: 24:43)



In the Galois LFSR you do this. So, you have again this  $n$  states as you can see like from  $S_0$  to  $S_{n-1}$ , but now the feedback is done in a slightly different way ok. So, you can see that what we do here is that the feedback is now kind of put across every you know like at every stage for example, this is 1 stage you like this is another stage and so, on ok. So, these are all the different stages and you basically you know like try to find out you basically tap out you know like these are the your tap positions that depending upon the feedback polynomial, you end it; that means, you enable that switch or you disable that switch.

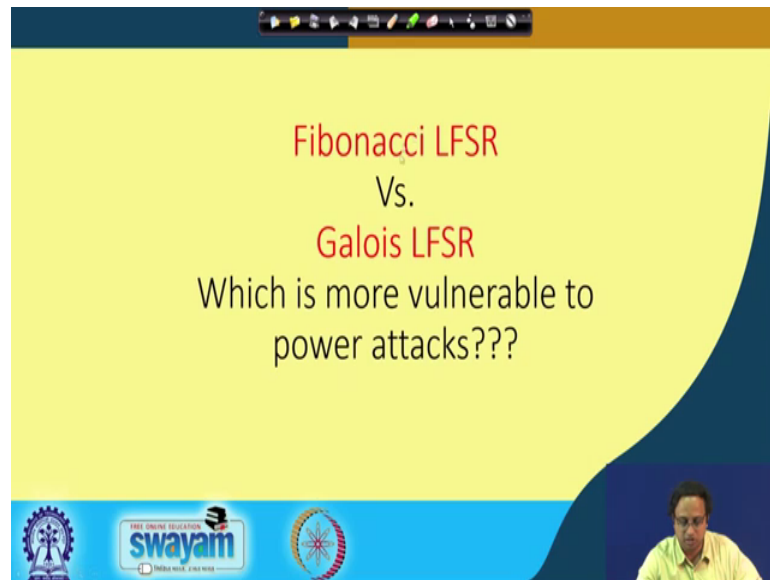
So, basically either you allow this feedback to pass or you do not allow this feedback to pass by using this switch. So, we call this say as the toggle switch, which means that if the toggle is 0 then this path is disabled, so; that means, you are not taping out this position again you know, you can essentially draw an isomorphic relationship which means that for every Fibonacci LFSR you can write a Galois LFSR and also for every Galois LFSR you can write a feedback you know like Fibonacci LFSR.

Also note that in this diagram and also in the previous diagram, we basically define what is something is called as a non-linear Boolean function. So, typically what is done is you know like based upon this gates in this bits that essentially are over here, we basically combined them by a non-linear feedback non-linear Boolean function and then produce this key stream which is XORed with the plaintext to give you the Ciphertext.



So, this is the kind of model that we compare with and the question is right that the question that we try to address in this discussion is whether these 2 things are equivalent or not.

(Refer Slide Time: 26:27)



Fibonacci LFSR  
Vs.  
Galois LFSR  
Which is more vulnerable to  
power attacks???

swayam

So; that means, we basically try to basically consider that whether Fibonacci LFSR and the Galois LFSR are equivalent with respect to power attacks. So, this we will see in the next class.

Thank you.