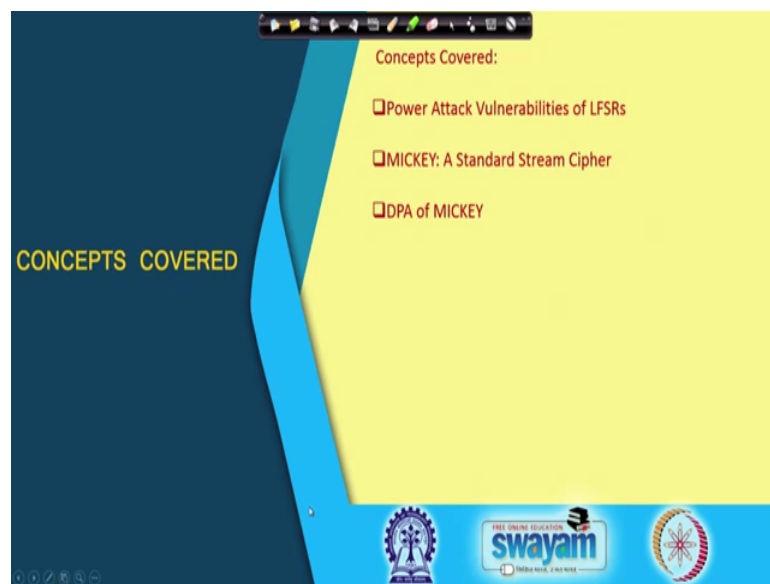


**Hardware Security**  
**Prof. Debdeep Mukhopadhyay**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 35**  
**Power Analysis – XI**

So, welcome back to this class on Hardware Security. So, we shall continue our discussions on the linear feedback, shift register and its resistance against power attacks.

(Refer Slide Time: 00:25)



So, we shall we be trying to discuss about power attacks of LFSRs and continue our discussion on the standard stream cipher which is called as MICKEY.

(Refer Slide Time: 00:31)

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

The slide features a yellow background with a dark blue curved border on the right. At the bottom, there are logos for Swamyam and other educational institutions, along with a small video feed of a presenter.

So, we basically stopped at this point where we were reflecting on the you know like the whether the Fibonacci LFSR is more vulnerable against power attacks compared to the Galois LFSRs.

(Refer Slide Time: 00:43)

Analysing Fibonacci LFSR

- $HD_t$  : Hamming Distance between two consecutive states of the LFSR at time  $t$ .
- $PD_t$  : Difference of consecutive HDs.
- $PD_t = HD_{t+1} - HD_t$
- It can shown that  $PD_t \in \{-1, 0, 1\}$

The slide features a yellow background with a dark blue curved border on the right. At the bottom, there are logos for Swamyam and other educational institutions, along with a small video feed of a presenter.

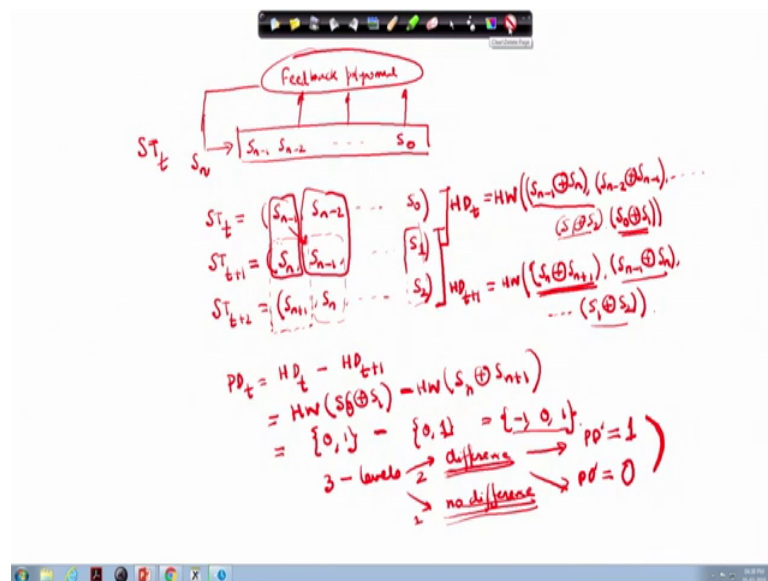
So, so, let us try to analyse the Fibonacci LFSR, ok. So, here we basically you see that the hamming distance or HD  $t$  that we have defined is the hamming distance between two consecutive states of the LFSR at time  $t$ . So, it turns out that we define a parameter called PD  $t$  and which is nothing, but the difference of consecutive HD  $t$ 's and this PD  $t$

if I denote it as  $HD_{t+1} - HD_t$ , ok. So, the idea is that this two this value can either be you know like minus 1, 0, or plus 1, ok.

So, if you remember right in the last class we basically defined PD dash, ok. So, PD dash was the fact that if the power consumption remains same or where as the power consumption defers. So, here you can see that the power consumption can remain same in one of the cases out of three cases. Whereas, the power consumption can differ in two cases out of the three cases; like when the  $PD_t - PD_{t+1}$  is minus 1, there is a difference in power consumption. Because you know like  $PD_{t+1} - PD_t$  minus 1 which means that  $HD_{t+1} - HD_t$  was one more than  $HD_t$ , and that is why this value got minus 1 and therefore, there is a change across clock cycles in the power consumption.

Likewise, when  $PD_t$  is 1, that also implies that, the power consumption is increased, ok. So, in both cases like in one case there is a decrease one case there is a increase, but; that means, that there is a change of power consumption. So, in order for the attack to work on real traces what we just need is to be able to distinguish on the from the you know like the fact that whether the power consumption has increased or the power, I mean whether the power consumption has remained the same or the power consumption has changed.

(Refer Slide Time: 02:35)



So, in this case right the  $PD_t$  value is minus 1, 0 and plus 1, and this you can easily understand. So, let us try to reflect on why this is so, ok. So, and that is pretty simple in

context to the Fibonacci LFSR, ok. So, in the Fibonacci LFSR remember that you have the state, right. So, the state let us write it as you know like  $S_{n-1} S_{n-2}$  so on till  $S_0$ . So, this is your LFSR state and you basically what you are basically trying to do is you are calculating  $S_n$  which is your feedback value, and this feedback value depends upon certain positions which were taking from here and you have got this feedback polynomial. So, this is your feedback polynomial which is nothing, but having some XORs and then you basically calculate the value of  $S_n$ .

So, that means, right I can write the value of say at of you know like let us denote this state as  $S^T$  and the corresponding time instance by the suffix  $t$ , then; that means, that  $S^T_t$  is this value right is basically  $S_{n-1}, S_{n-2}$ , so on till  $S_0$ , in the next time instance you have got  $S^T_{t+1}$ . So, in  $S^T_{t+1}$  what will happen is that this  $S_n$  will get into the register. So, it will be  $S_n$  comma and this will get shifted, so, you will have  $S_{n-1}$  and so on; this will continue till  $S_1$ .

Likewise in the next time instance you will have  $S^T_{t+2}$  and this is nothing, but  $S_{n+1}$  comma  $S_n$  and comma and this will continue till  $S_2$ . So therefore, right from here we can calculate the value of  $HD_t$ , right and what is the value of  $HD_t$  as we have discussed right it is nothing, but the hamming weight because of the hamming distance between those two parameters. It is the hamming weight of  $S_{n-1}$  XORed with  $S_n$ , ok. So, that means, these two things are XORed, ok; likewise I will XORed these two, ok. So, it is  $S_{n-2}$  XORed with  $S_{n-1}$  and likewise the finally, I got an  $S_0$  XORed which  $S_1$ .

Likewise if I want to calculate  $HD_{t+1}$  that  $HD_{t+1}$  is nothing, but the hamming weight of I will be now considering the hamming weight means. This is these are the two things that will be XORing now. So, I will have here  $S_n$  XOR with  $S_{n+1}$ , and then again I will be XORing these two. So, it will be  $S_{n-1}$  XORed with  $S_n$ , and like this right I will be XORing these two. So, it is a  $S_1$  XORed with  $S_2$ .

So now, the definition of  $PD_t$  is nothing, but  $HD_t$  minus  $HD_{t+1}$ , and you can observe that here among all these terms you can see that this term matches with this term, and likewise if you had observed the previous time to this right it should have been  $S_1$  XOR of  $S_2$ . So, this also would have matched with this one ok. So therefore, the new term that is over here is one of them is this and the other one right is this, ok.

So, therefore, this difference would be nothing, but the hamming weight of  $S_0$  or you know like  $S_0 \oplus S_1$  minus the hamming weight of  $S_n \oplus S_{n+1}$  ok. So, note because these are the two things which are only different and therefore, write in the final result will depend on the difference between these two.

So, this hamming weight right since is there hamming XORed of only 1-bit values these can be either 0 or 1 and likewise this also can be either 0 or 1. And therefore, when I take the difference then the difference can be either minus 1, 0 or plus 1, and that is why right we have got three levels I would say here, but out of which right two are where there is a difference, and there is one case where there is no difference; that means, the power essentially remains constant, ok. And therefore, right in this case right we will have the PD dash value as 1 whereas in this case right your PD dash value will be 0 because there is no difference.

So, now, you see that the power of my attack depends on my ability of directly calculating this PD dash, which means of being correctly distinguishing this non-zero difference from this zero difference. So, that means, I mean from the non-zero difference from the I mean the non-zero difference from the zero difference, ok.

So, that implies that if right for example, if I have in this case right since there are only two non-zero differences compared to one zero-difference we will now compare this with the Galois LFSR, and we will see that you know like the PD t dash values or in the PD dash the PD t value there can take more levels, and that essentially will be my you know like the base of the argument or base of my comparison between these two configurations.

(Refer Slide Time: 08:29)

**Analysing Fibonacci LFSR**

- $HD_t$  : Hamming Distance between two consecutive states of the LFSR at time  $t$ .
- $PD_t$  : Difference of consecutive HDs.
- $PD_t = HD_{t+1} - HD_t$
- It can be shown that  $PD_t \in \{-1, 0, 1\}$

swayam

So, let us you know like see how this PD value looks like in the when we are in particular comparing and finding out the same stuff for the Galois LFSR.

(Refer Slide Time: 08:37)

**Analysing Galois LFSR**

- Number of taps :  $N$
- For a LFSR with primitive connection polynomial ,  $N$  is even.
  - $PD_t = HD_{t+1} - HD_t$
- It can be shown that  $PD_t \in \{-(N-1), -(N-3), \dots, -1, 0, 1, \dots, (N-3), (N-1)\}$

Handwritten notes:  $PD_t \neq 0$  because  $\neq 0$  because  $\dots N-1$

Abhishek Chakraborty, Bodhisatwa Mazumdar, Debdeep Mukhopadhyay: Fibonacci LFSR vs. Galois LFSR: Which is More Vulnerable to Power Attacks? SPACE 2014: 14-27

swayam

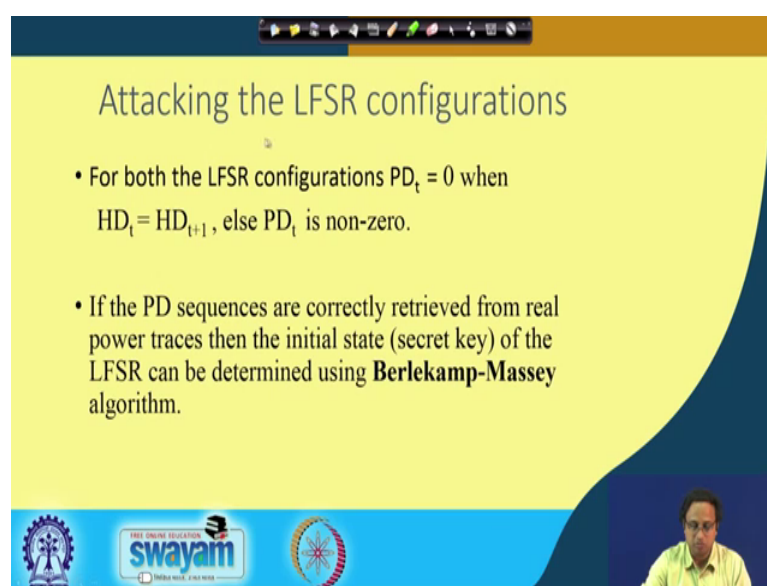
So, in Galois LFSR if your number of taps is  $N$  and for an LFSR with primitive connecting correction polynomial then  $N$  is even. And, in that case again I if I define PD dash in a similar way we can argue that the PD dash the PD or the PD  $t$  value can have more levels ok. So, you can for example, have levels of course, it will have some zero values, but there are more non zero values; like it can have minus 1 to minus  $N$ ,  $N$   $N$

minus 1 and likewise it can have from 1 to plus N minus 1, ok. So, that implies that in a similar way. So, in a similar way if you do if you observe right then. So, likewise right in this case if I take the PD t value and find out the zero difference case right in the zero difference there is one case because this is your zero difference case whereas, there are many non-zero difference levels.

So, and the non-zero different levels also observed that for example, it can go as high as up to n minus 1, ok. So, which means although you know so, if you are you know like doing this attack on real power traces then you can intuitively understand that it will be hard for you to distinguish these cases because they are almost close, ok. Whereas, if you have got some jumps which are you know like as high as this, then they are more discernible there more distinguishable from this zero level.

And, you can see more you know like more details on this derivation in this reference which has been shown here which is essentially a paper which is published in space in 2014 which talks about Fibonacci LFSR versus Galois LFSR which is more vulnerable to power attack. So, you can see a derivation of this particular result ok, but at this point right what is just important for us is to know this fact that in a Galois LFSR there are more jumps which are possible the PD t value can simply take more levels. So, what is the implication of this?

(Refer Slide Time: 10:39)



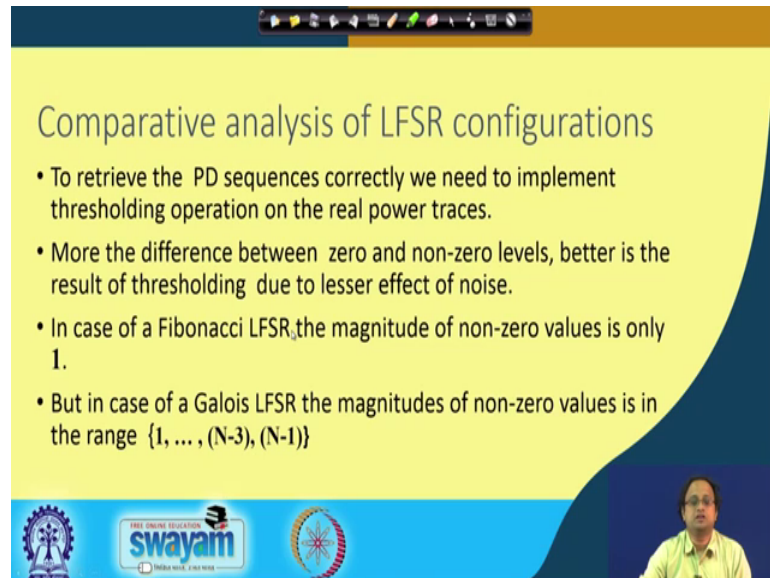
Attacking the LFSR configurations

- For both the LFSR configurations  $PD_t = 0$  when  $HD_t = HD_{t+1}$ , else  $PD_t$  is non-zero.
- If the PD sequences are correctly retrieved from real power traces then the initial state (secret key) of the LFSR can be determined using **Berlekamp-Massey** algorithm.

swamyam  
FREE ONLINE EDUCATION  
INDIA RISE, AS TOGETHER

So, therefore, right when you are trying to attack the LFSR configurations for both the LFSR conditions, right the PD t value equal to 0 when HD t equal to HD t plus 1 else the PD t is a non-zero value and if the PD sequences are correctly retrieved from the real power traces them the initial state the secret key of the LFSR can be completely determined using Berlekamp-Massey algorithm as we have already discussed.

(Refer Slide Time: 11:03)



The slide features a yellow background with a dark blue curved border on the right side. At the top, there is a navigation bar with various icons. The main content is a list of four bullet points. At the bottom, there are three logos: the Indian Institute of Space Science and Technology (IIST) logo, the 'swayam' logo, and the Indian Institute of Information Technology (IIIT) logo. A small video inset of a presenter is visible in the bottom right corner.

### Comparative analysis of LFSR configurations

- To retrieve the PD sequences correctly we need to implement thresholding operation on the real power traces.
- More the difference between zero and non-zero levels, better is the result of thresholding due to lesser effect of noise.
- In case of a Fibonacci LFSR, the magnitude of non-zero values is only 1.
- But in case of a Galois LFSR the magnitudes of non-zero values is in the range  $\{1, \dots, (N-3), (N-1)\}$

So, therefore, right this forms the basis of our comparison between these two configurations to retrieve the PD sequences correctly we need to implement a thresholding operation on the real power traces because in real power traces you to have a proper threshold to tell that this is you know like a zero difference and this is a non-zero difference because you will have real values on the in the power trace.

So, more the difference between a zero and non-zero level therefore, better will be your you know the result of thresholding and because the effect of noise will be less in such case. You will have a high you will have a better distinguishing power. So, as we have seen in case of Fibonacci LFSR the magnitude of non-zero values is only 1, which means the zero state the zero difference and the non-zero difference are very close to each other. On the other hand, in a Galois LFSR, the magnitudes of non-zero values is very large. It can go as high as  $n$  minus 1, ok. So, it is pretty large.



(Refer Slide Time: 12:01)

Comparative analysis (contd.)

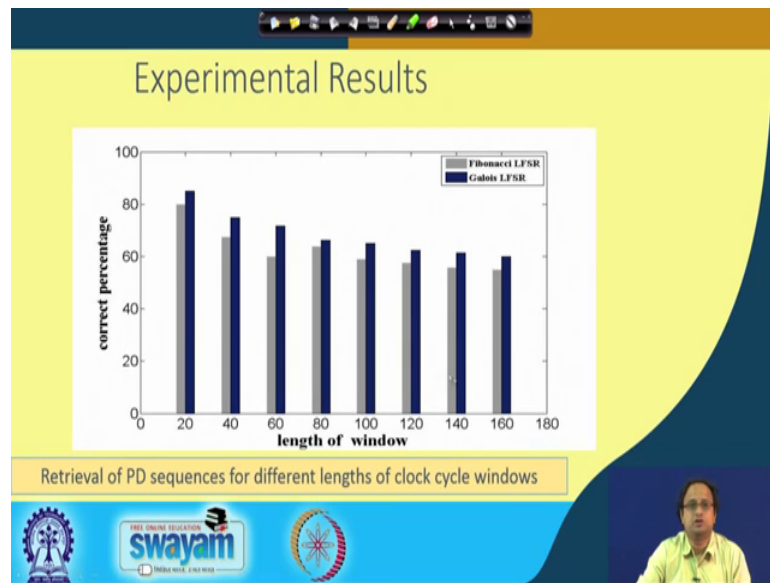
- It is clear that the difference between zero and non-zero levels of PD sequences is higher in Galois LFSR than it's Fibonacci counterpart except one case (for value 1).
- Higher distinction between the two levels imply that the effect of low SNR power sample points is less.
- Therefore we can conclude that the **Galois LFSR configuration is more vulnerable to power attacks** compared to its Fibonacci counterpart.

swamyam  
FREE ONLINE EDUCATION  
INDIA WISE, LEAD WISE

So therefore, right it is clear that the difference between zero and non-zero levels of PD sequences is higher in the Galois LFSR than it is Fibonacci counterpart, ok, as we have seen that the gap is much wider in the Galois LFSR. And therefore, right the there is a higher distinction possible in the case of Galois LFSR between the two levels and this implies the effect of low SNR power sample points is less ok; that means, the effect of noise will be less in the case of Galois LFSR.

Because, remember that if there is a noise on top of it right then there is a there is a quite high chance that the 0 and the 1 difference will get blurred. Whereas, right if you have got large differences possible as we can see that is possible in the case of the Galois LFSRs, then the effect of noise will be less, ok. So therefore, right we can conclude that the Galois LFSR configuration is most vulnerable to power attacks compared to the Fibonacci counterpart.

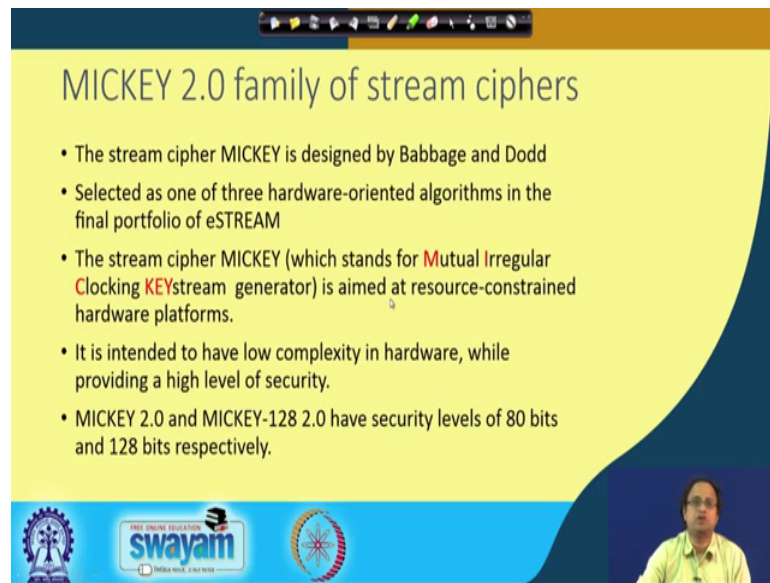
(Refer Slide Time: 13:01)



You can also experimentally observe this. So, this is an example of an attack which has been done on the Fibonacci LFSR shown by the grey colour compare with a blue colour which talks about the Galois LFSR. And, we have done several you know like instances of the in the attack is shown in this graph and you can see that in almost all the cases for the Galois LFSR, the correct percentage of retrieval of the PD sequences is higher, ok.

So, this bar shows you know like the correct percentage of the retrieve of retrieval of the PD sequences and as you can observe that the you know the retrieval rate is higher in case of the is higher in case of the Galois LFSRs and that is also expected from the theory that we just discussed. So, now with this background right it basically tells us that even stream ciphers like block ciphers are also vulnerable against power attacks or DPA attacks.

(Refer Slide Time: 13:59)



The slide features a yellow background with a dark blue curved border on the right side. At the top, there is a navigation bar with various icons. The title 'MICKEY 2.0 family of stream ciphers' is displayed in a large, dark font. Below the title, there is a list of five bullet points. At the bottom of the slide, there are three logos: the Swamyam logo, a circular logo with a gear and a person, and another circular logo with a gear and a person. A small video inset of a person is visible in the bottom right corner of the slide.

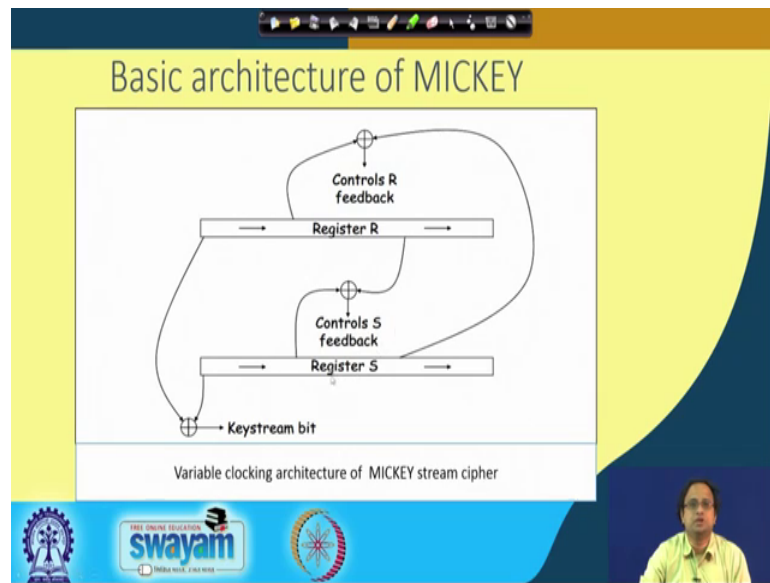
## MICKEY 2.0 family of stream ciphers

- The stream cipher MICKEY is designed by Babbage and Dodd
- Selected as one of three hardware-oriented algorithms in the final portfolio of eSTREAM
- The stream cipher MICKEY (which stands for **M**utual **I**rregular **C**locking **K**EYstream generator) is aimed at resource-constrained hardware platforms.
- It is intended to have low complexity in hardware, while providing a high level of security.
- MICKEY 2.0 and MICKEY-128 2.0 have security levels of 80 bits and 128 bits respectively.

Let us try to see you know you know like if I try to apply a DPA on an actual stream cipher. So, in an actual stream cipher right you will not have only like one single register, but may be you will have much more registers. So, MICKEY or what it stands for you know that the MICKEY was essentially a stream cipher which was designed by Babbage and Dodd. And it was selected as one the three hardware oriented algorithms in the final portfolio of eSTREAM which was a movement to you know or a you know like to basically standardise stream ciphers.

So, the stream cipher MICKEY which stands for Mutual Irregular Clocking KEYstream generator is aimed at resource-constrained hardware platforms and is intended to have a low complexity in hardware while providing a high level of security in. In fact, you know MICKEY 2 and MICKEY 128 2.0 have got security levels of 80 bits and even as high as 128 bits respectively.

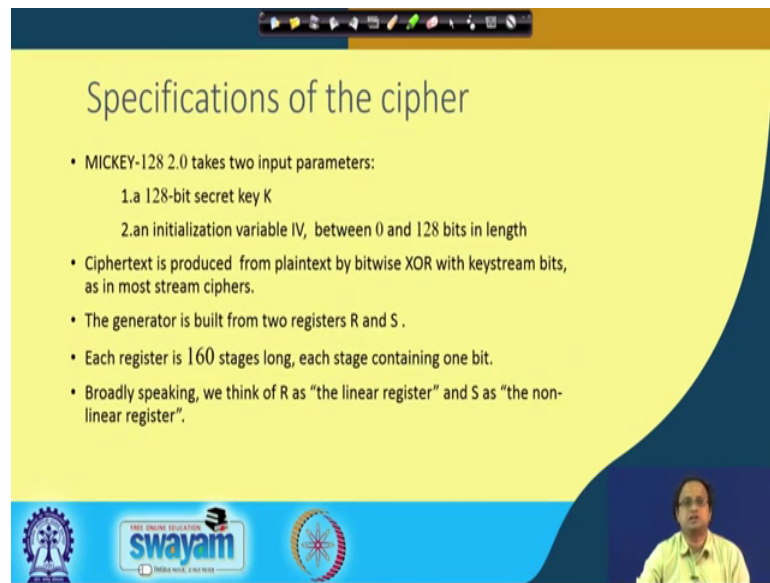
(Refer Slide Time: 14:55)



So, this is how the MICKEY looks like you can see that there are two registers; register R which is a linear feedback shift register and register S which is a non-linear feedback shift register. And there are there is a control for the independent control for the register R and for the register S, ok.

So, that means, the feedback essentially or the clocking of both the LFSRs is essentially takes place in a somewhat irregular way and that is why the its name. And there are finally, XOR to get your keystream way bits this is the corresponding output of your key stream generator.

(Refer Slide Time: 15:31)



The slide is titled "Specifications of the cipher" and contains the following bullet points:

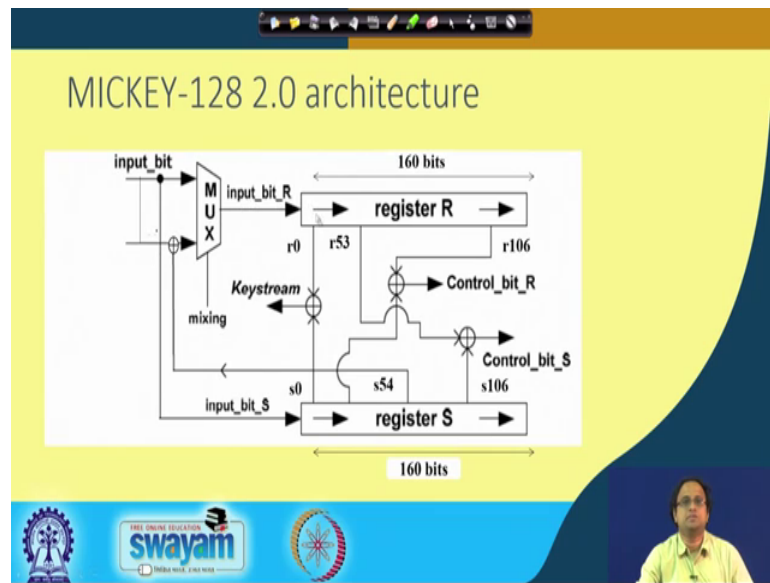
- MICKEY-128 2.0 takes two input parameters:
  1. a 128-bit secret key  $K$
  2. an initialization variable  $IV$ , between 0 and 128 bits in length
- Ciphertext is produced from plaintext by bitwise XOR with keystream bits, as in most stream ciphers.
- The generator is built from two registers  $R$  and  $S$ .
- Each register is 160 stages long, each stage containing one bit.
- Broadly speaking, we think of  $R$  as "the linear register" and  $S$  as "the non-linear register".

At the bottom of the slide, there are logos for "swayam" (Free Online Education) and "INDIA WIDE, 24x7x365". A small video inset in the bottom right corner shows a person speaking.

But, there is you know like. So, there are different stages of how of this particular LFSR works. So, as I said that MICKEY 128 2.0 takes two input parameters it takes a 128-bit secret key  $K$ . It also takes a initialisation vector which is a public information which is between 0 and 128-bit bits in length. The cipher text is produced from the plaintext by bitwise XORing between the key stream bits and as we have seen right, in most in stream ciphers that this is the output bit you know like when you take this key stream bit you XOR it with a message bits and you will get your corresponding cipher text bit, ok.

So, the generator is now as we had seen in the picture is built from two registers; register  $R$  and register  $S$ . Both of these registers are 160 bits long, and each state containing one bit and broadly speaking you can think of  $R$  as a linear registered whereas,  $S$  is essentially a non-linear register.

(Refer Slide Time: 16:29)



So, this is how the MICKEY architecture looks like and you can see that the register R and the register S are essentially updated and you know why the input bit is shown here as input a bit R and this the input bit S. And, there are you know two tap positions which I am basically taking from the register R and the register S and we are XORing it to get the key string which is XORed with your message.

Now, there is an important point in this architecture like you can see this is the input bit, ok. So, this input bit is basically you know like mark stream depending up on this select line which is mixing. So, this mixing as we will see right is initially kind of disabled, ok. So, I mean rather you know like you basically initially you would like to you know like initially you have to load in the there are two parameters that you have to load the initialisation vector and the key which are done in two specific stages of the of the stream cipher.

(Refer Slide Time: 17:25)

The slide is titled "Variable Clocking of Registers" and is part of a presentation. It contains the following text:

**I. CLOCKING OF REGISTER R**

**CLOCK\_R (R, INPUT\_BIT\_R, CONTROL\_BIT\_R)**

- Let  $r_0 \dots r_{159}$  be the state of the register  $R$  before clocking, and let  $r'_0 \dots r'_{159}$  be the state of the register  $R$  after clocking.
- $FEEDBACK\_BIT = r_{159} \oplus INPUT\_BIT\_R$
- For  $1 \leq i \leq 159$ ,  $r'_i = r_{i-1}$ ;  $r'_0 = 0$
- For  $0 \leq i \leq 159$ , if  $i \in RTAPS$ ,  $r'_i = r'_i \oplus FEEDBACK\_BIT$
- If  $CONTROL\_BIT\_R = 1$ :
  - For  $0 \leq i \leq 159$ ,  $r'_i = r'_i \oplus r_i$

The slide also features a Swamyam logo at the bottom left and a small video feed of a presenter at the bottom right.

So, let us see how essentially we do that. So, first we clock, so; let us see about the clocking of the registers R and S. So, the first thing is that we see is that how we clock register R. So, in register R right there are as we can see the parameters are R INPUT BIT R and the CONTROL BIT R the idea is that. So,  $r_0$  to  $r_{159}$  are the states of the register R before clocking and after we have done the clocking right they are denoted as  $r'_0$  to  $r'_{159}$ ; note that they are 160 bits in length.

The FEEDBACK BIT as we have seen write is calculated by XORing of  $r_{159}$  with this input bit R signal, ok. So, that means, right you are either toggling  $r_{159}$  or you are basically passing  $r_{159}$  as it is and then this is the evolution of the of the shift register. As you can see that from 1 to 59 I am just shifting, ok. And, and then what we do is we basically calculate the value of  $r_i$  again you know like you basically take in the FEEDBACK BIT and you XOR it with the corresponding  $r_i$  values and if the CONTROL BIT R is 1, then you basically XOR  $r_i$  with  $r_i$  you know  $r_i$  with  $r_i$  and you get the value of  $r_i$ .

(Refer Slide Time: 18:43)

Variable Clocking continued...

**2. CLOCKING OF REGISTER S**

- Let  $s_0 \dots s_{159}$  be the state of the register  $S$  before clocking, and let  $s'_0 \dots s'_{159}$  be the state of the register after clocking. We will also use  $\hat{s}_0 \dots \hat{s}_{159}$  as intermediate variables to simplify the specification.
- $FEEDBACK\_BIT = s_{159} \oplus INPUT\_BIT\_S$
- For  $1 \leq i \leq 158$ ,  $\hat{s}_i = s_{i-1} \oplus ((s_i \oplus COMPO) \wedge (s_{i+1} \oplus COMPI))$ ;  $\hat{s}_0 = 0$ ;  $\hat{s}_{159} = s_{158}$ .
- If  $CONTROL\_BIT\_S = 0$ :
  - For  $0 \leq i \leq 159$ ,  $s'_i = \hat{s}_i \oplus (FBO \wedge FEEDBACK\_BIT)$
- If instead  $CONTROL\_BIT\_S = 1$ :
  - For  $0 \leq i \leq 159$ ,  $s'_i = \hat{s}_i \oplus (FB1 \wedge FEEDBACK\_BIT)$

swayam

So, likewise right you can also you know like you can go into the specifics, but essentially there is another way in which you are clocking register S. So, you are basically independent clockings of register R and clockings of register S the point one point which you can probably see what here is that there is an in the in the feedback over here there is an AND gate which has been used. This AND like the if you see like you know like now this AND is essentially is one of the reasons why we call this register S as a non-linear feedback shift register.

(Refer Slide Time: 19:19)

Clocking of overall generator

**3. CLOCKING THE OVERALL GENERATOR**

$CLOCK\_KG(R, S, MIXING, INPUT\_BIT)$

- $CONTROL\_BIT\_R = s_{94} \oplus r_{106}$
- $CONTROL\_BIT\_S = s_{106} \oplus r_{93}$
- If  $MIXING = TRUE$ ,
  - $CLOCK\_R(R, INPUT\_BIT\_R = INPUT\_BIT \oplus s_{90}, CONTROL\_BIT\_R = CONTROL\_BIT)$
  - $CLOCK\_S(S, INPUT\_BIT\_S = INPUT\_BIT, CONTROL\_BIT\_S = CONTROL\_BIT)$
- If instead  $MIXING = FALSE$ ,
  - $CLOCK\_R(R, INPUT\_BIT\_R = INPUT\_BIT, CONTROL\_BIT\_R = CONTROL\_BIT)$
  - $CLOCK\_S(S, INPUT\_BIT\_S = INPUT\_BIT, CONTROL\_BIT\_S = CONTROL\_BIT)$

swayam



So, basically you update the register R and the register S through these algorithms and then you can you clock the overall generated in this way. So, you basically if you set your mixing as TRUE then you basically CLOCK your R and you CLOCK your S by you know like by this equations. So, you can see that here I take my INPUT BIT R and I just XOR the input and I obtain it by XORing INPUT BIT with S 80 which is one of the states which I am deriving from S.

So, you can see that it is very interesting way of updating the clocks because now register R is being mixed with the state of register S, and that is happening because mixing is said to be true. On the other hand, I you can see that the clock S essentially where your updating state is you are you know like mixing in the you know you are basically in this case by the mix the clock S is independently processed. So, you basically just take the INPUT BIT S and you initialise it to input or you basically assign it INPUT underscore BIT to INPUT underscore BIT underscore S.

On the other hand, right if you are setting MIXING to FALSE then register R works independently. So, therefore, this particular XOR is not present when you are updating say CLOCK underscore R.

(Refer Slide Time: 20:35)

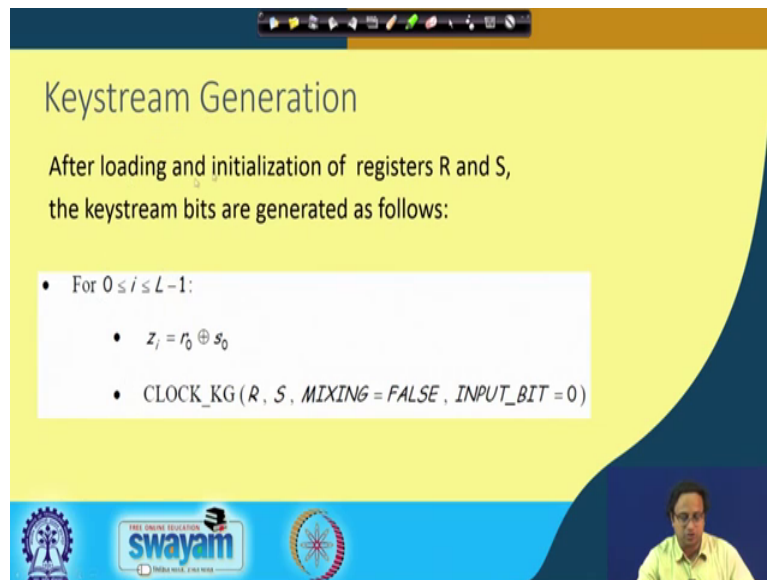
The slide is titled "Key loading and Initialization" and contains the following bullet points:

- Initialise the registers  $R$  and  $S$  with all zeros.
- (Load in  $IV$ .) For  $0 \leq i \leq IVLENGTH - 1$ :
  - $CLOCK\_KG(R, S, MIXING = TRUE, INPUT\_BIT = iv_i)$
- (Load in  $K$ .) For  $0 \leq i \leq 127$ :
  - $CLOCK\_KG(R, S, MIXING = TRUE, INPUT\_BIT = k_i)$
- (Preclock.) For  $0 \leq i \leq 159$ :
  - $CLOCK\_KG(R, S, MIXING = TRUE, INPUT\_BIT = 0)$

The slide also features logos for Swamyam and other educational institutions at the bottom.

So, now you can observe that so, I say that initially there is a phase when you are uploading the you are loading the key and your loading the initialisation vector. So, first you load in the IV and then you load in key ok, K and then you do a pre clock session.

(Refer Slide Time: 20:49)



### Keystream Generation

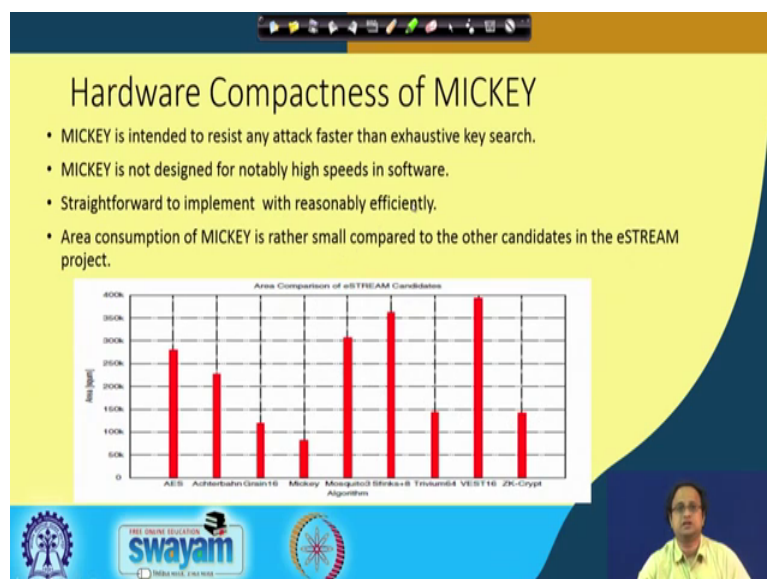
After loading and initialization of registers R and S, the keystream bits are generated as follows:

- For  $0 \leq i \leq L-1$ :
  - $z_i = r_0 \oplus s_0$
  - `CLOCK_KG(R, S, MIXING = FALSE, INPUT_BIT = 0)`

The slide also features logos for Swamyam and other educational institutions at the bottom.

And, then after that you start your key stream generation, ok. So, you can see that in all the states right mixing has been set to true which means since mixing is true the clock register R is updated by this equation. And, finally, right when you are you know like generating the key stream at that point MIXING is set to FALSE. And therefore, you are just using this to generate the you know you are just using it to generate your key stream which you are XORing with the message to get the cipher text bits.

(Refer Slide Time: 21:21)



### Hardware Compactness of MICKEY

- MICKEY is intended to resist any attack faster than exhaustive key search.
- MICKEY is not designed for notably high speeds in software.
- Straightforward to implement with reasonably efficiently.
- Area consumption of MICKEY is rather small compared to the other candidates in the eSTREAM project.

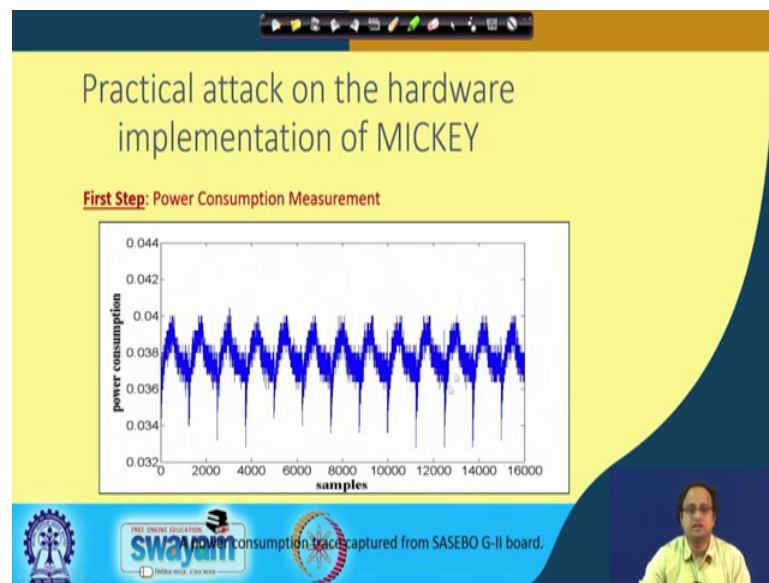
Area Comparison of eSTREAM Candidates

Algorithm	Area (µm²)
AES	~28000
Acorn/Grain	~22000
Grain128	~12000
Mickey	~8000
Micropuff	~30000
Serpent	~38000
Trivium	~18000
VECTO	~40000
ZUC-Crypt	~15000

The slide also features logos for Swamyam and other educational institutions at the bottom.

So, therefore, here is so, so that is the more or less you know like working of the MICKEY stream cipher. And, the MICKEY was essentially very popular in particular when we discuss about hardware design. So, you can see here this the result or the footprint of hardware implementation on MICKEY compared to some other potential candidates like Trivium and so on and you can see that MICKEY essentially stands out by taking quite small amount of area in terms of resources.

(Refer Slide Time: 21:49)



But, as we can see right that as in most algorithms or most cryptographic algorithms it also use vulnerable against power attacks, ok. So, in order to see that we basically observe as in the first state we basically put in our setup for doing power trace acquisition and here you can see how the power trace of MICKEY looks like. So, they are the various samples that has been observed and this is a power consumption you can see there is a distinct nature of the power traces and that is expected from the you know like the evolution of the stream cipher. So, this has been acquired on a SASEBO G-II board, ok.

(Refer Slide Time: 22:27)

**Second step:** Building the hypothetical power consumption matrix by power model

- **Basic idea:**
  1. to look for intermediate values which can reflect the power consumption effectively
  2. then choose a proper power model

In our target cipher MICKEY -128 2.0 the registers R and S are chosen as intermediate values.

**Attack point:** Duration of key loading (by varying IV, keeping key fixed)

**Third step:** Attacks based on different statistical analysis (in our case Correlation Power Analysis attack).

And, in the second step we basically try to develop the attacks. So, as very important, right; I mean in DPA as we have seen in even in the context of block ciphers you need to make a target position, ok. So, the basic idea is that you basically built the your hypothetical power consumption matrix by a power models. So, this could be hamming weight or hamming distance power model and the basic idea is to look for intermediate values which can reflect the power consumptions effectively, and then you choose a proper power model.

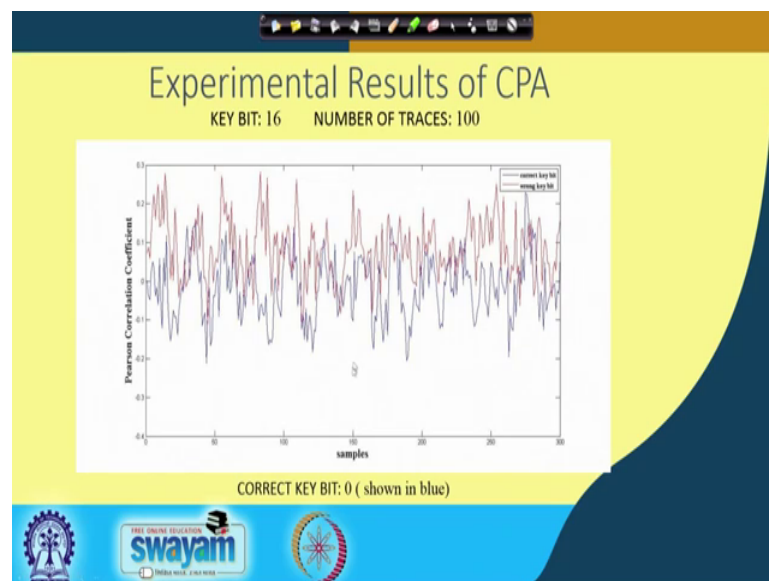
So, what we do in our case is basically we target the register R and the register S, and we try to find out the kind of the summation of the register R content and the register S content and we find out the hamming weight of that essentially as your power model. So, then we attack. So, when do we attack? So, as we have seen that in the stream cipher there are different fields of the stream cipher. In particular we basically target the phase where we are uploading the key, as you can see that the key bits are uploaded one by one and we basically do that.

So, we basically keep the key fixed and we kind of vary the initialisation vector and that is and we specifically target the content of the register R and the register S to derive a hypothetical power and then we kind of match this hypothetical power or correlate this hypothetical power with the real power which has been observed by our observations, ok.

So, that is from your side channel acquisition. So, and that that is done that is done in the third step where we basically you know like try to co-relate and we basically apply like co-relation power analysis and we recover the key bit by bit ok; that means, you basically you know like load the key bit the key bit can be one you basically guess the key bit the. The key bit can be 1, can be 0 and then you do a CP attack and try to kind of confirm whether your guess was correct or whether your guess was wrong, ok.

The idea is that if your guess was correct if your key bit was correctly guessed then right, the hypothetical power; that means, the content of the register R and the content of the register S will co-relate very highly with the actual observed power. Whereas if your guess was wrong; that means, if your key bit was wrongly guessed then you are correlation will be small. And therefore, you can distinguish between these two cases to discard the wrong guesses and correctly know the key in a bit by bit fashion.

(Refer Slide Time: 24:43)

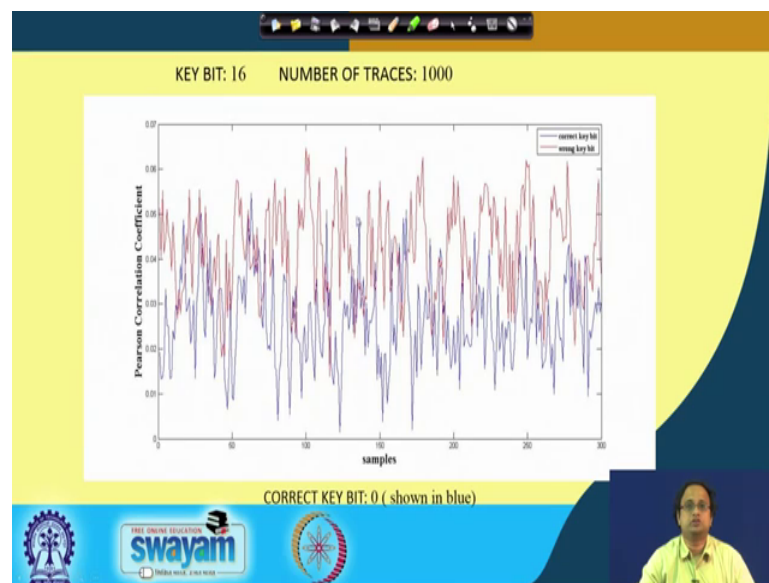


So, here are some experimental results that you can observe like this for example, when the key bit 16 is being tried to be recovered understand that when you are trying to recover key bit 16 that I am assuming that from key bit 0 to key bit 15 has already been recovered, and I am just guessing the key bit 16 I am making a guess either 0 or 1. In this case, right the correct bit is 0 and it is shown in blue and you can see that blue is below, ok.

So, one of the reasons right I take the one which is below is because you know I will get a negative dip in my power consumption, and therefore, I basically observe the negative correlation, ok. But, something which co-relates more in the negative direction in this particular graph 0 is at this point. So, you can find out 0 essentially this is this line, ok. So, 0 is this line; that means, right I observed that since 0 is this line the blue line that is the that is a one which corresponds to the correct key bit essentially is correlating more compared to the red lines which are essentially are more at the top, and therefore, right we basically can correct to determine that the correct bit is 0.

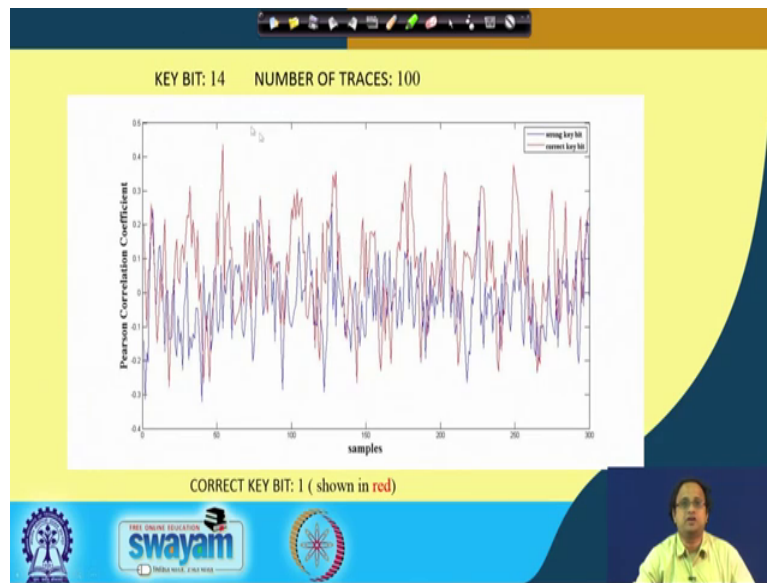
You can do similar analysis at different positions. For example, this is again when we have increased the number of power traces. So, in the previous case the number of power traces was 100, here the number of power traces is made to be 500 and here you can see peaks in the negative direction are even more and they are therefore, more distinguishable from the wrong guess.

(Refer Slide Time: 26:11)



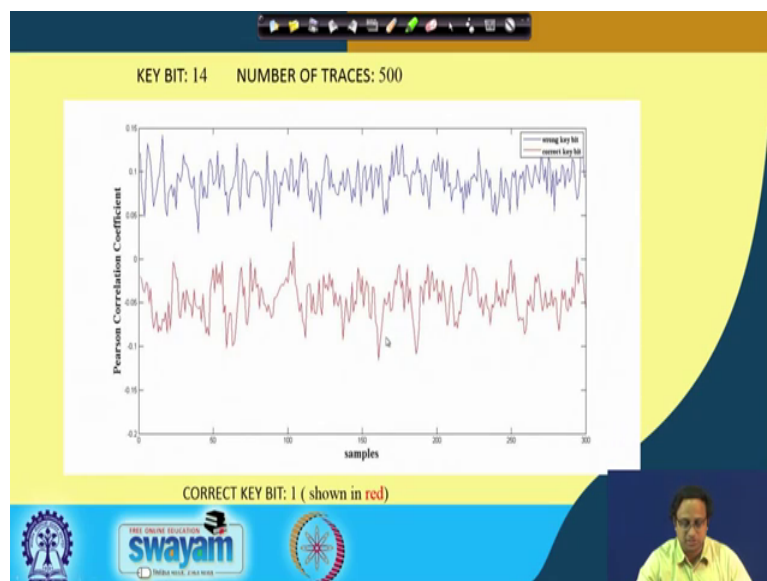
You can so, this is an example again when we are increasing further the number of phases you can see that the negative direction correlations are increasing further and therefore, they are more distinguishable, ok.

(Refer Slide Time: 26:25)



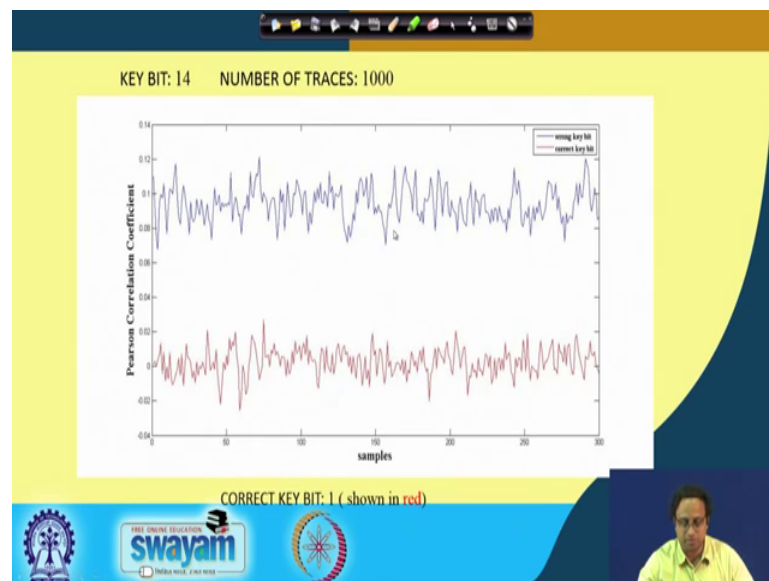
So, you can similarly target at other key bit positions. So, this is a scenario when key bit 14 is being targeted. So, in this case 0 till 13 are already been recovered and I am just guessing key bit 14 and you can observe that again there is a similar you know like distinction which is being done and you can observe that there is an amount of separation which you get with 100 power traces, ok. But, here it is not so, distinguishable as you can observe right because the red and the blue are kind of quite inter bind, ok.

(Refer Slide Time: 26:53)



So, now what happens is when I am increasing number of power traces in this case I see the separation is quite nice, ok. So, you can see that these are the; so, in this case the correct key bit is shown by the red colour and you can see that there is a more negative correlation right which is what we are expecting here and therefore, you can we can correctly say that the correct bit in this case is one ok. So, in so, this is shown in in red colour here.

(Refer Slide Time: 27:19)



If you are taking 1000 traces the separation is even more, ok.

(Refer Slide Time: 27:21)

### Remarks regarding the attack

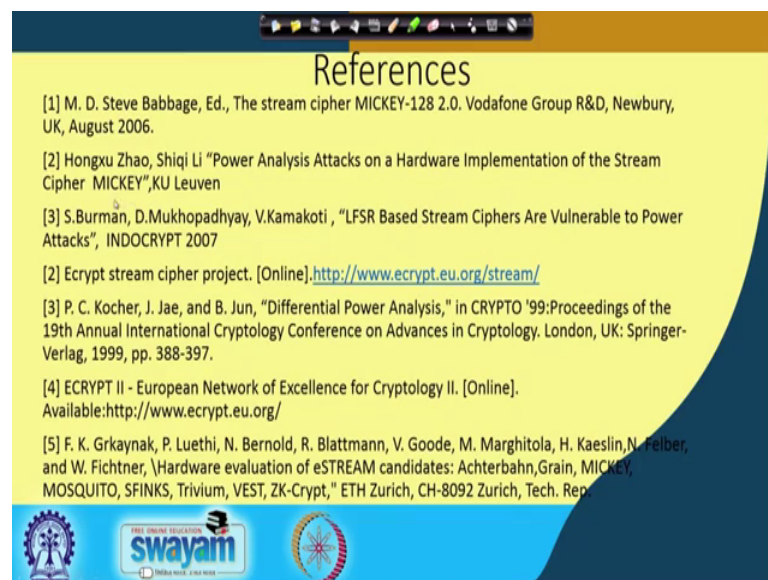
- In our experimental setup higher power consumption corresponds to a dip in the negative direction.
- Therefore in CPA attack we look for negative peak(s) to predict the correct key bit value.
- Nature of correlation plots varies from bit to bit !
- In some cases, only 500 traces (by varying IV keeping key fixed) are enough to distinguish correct key bit from the wrong one.
- Around 1000 traces are found to be sufficient in most of the cases.



So, we can make some remarks about the attack is that in our experimental setup higher power consumption corresponds to a dip in the negative direction, and that can happen in your set up. So, depending upon that, you have to look for co-relations either in the positive direction or the negative direction. So, therefore, in our case we look for negative peaks to predict the correct key bit value. The nature of the correlation plot varies from bit to bit, ok.

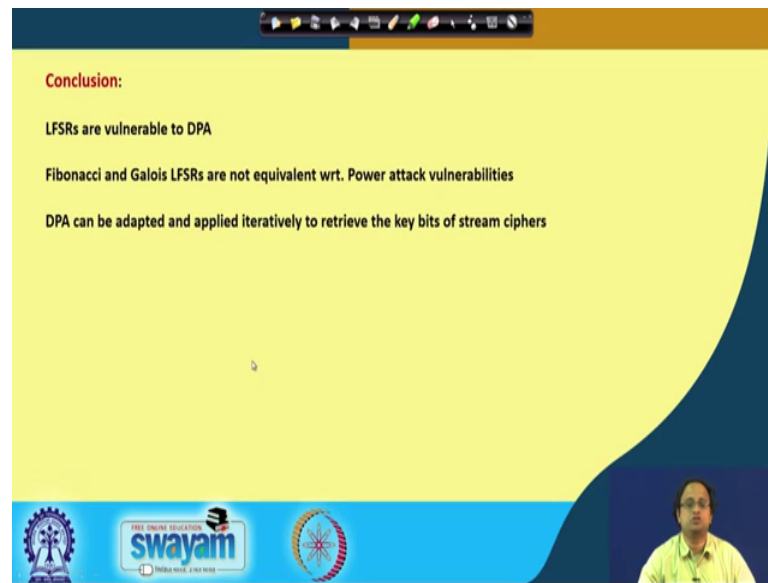
In some cases only 500 traces are quite good enough. In the some cases right we find out that we need more. And, in what we found out is that around 1000 traces are found to be sufficient to get the complete retrieval of I mean to get the retrieval done or to get the retrieval be successful in all the possible bit positions.

(Refer Slide Time: 28:05)



So, here are some references which we are followed from the book, ok; I mean which we have followed for this discussion. In particular right this reference 3, is the paper that I referred when I was talking about the linear feedback shift register, that is the Fibonacci LFSR in particular. Whereas, there are some other references which essentially talks about power attacks on stream ciphers. For example, this second reference is on a power and it is attack in hardware implementations of the stream cipher MICKEY which we kind of read it and exhibited here, ok.

(Refer Slide Time: 28:37)



**Conclusion:**

LFSRs are vulnerable to DPA

Fibonacci and Galois LFSRs are not equivalent wrt. Power attack vulnerabilities

DPA can be adapted and applied iteratively to retrieve the key bits of stream ciphers

So, therefore, I like to conclude what we discussed is that LFSRs are vulnerable to DPA. In particular Fibonacci and Galois LFSRs are not equivalent with respect to power attacks vulnerabilities although theoretically they are isomorphic to each other, but they are not so when it when we consider power attacks and we found out that the Galois LFSRs are more vulnerable to DPA. Because, there has more separations right there are more levels of jumps which are possible in the Galois LFSR and therefore, they are more vulnerable to a power attack.

In particular we also discussed that DPA can be adopted and then applied iteratively to retrieve the key bits of stream ciphers we showed in by an example of MICKEY, but pretty much this idea could be extended to others stream ciphers like grain, Trivium and so on ok.

So, with that we would like to come to an end to this class and thank you for your attention.