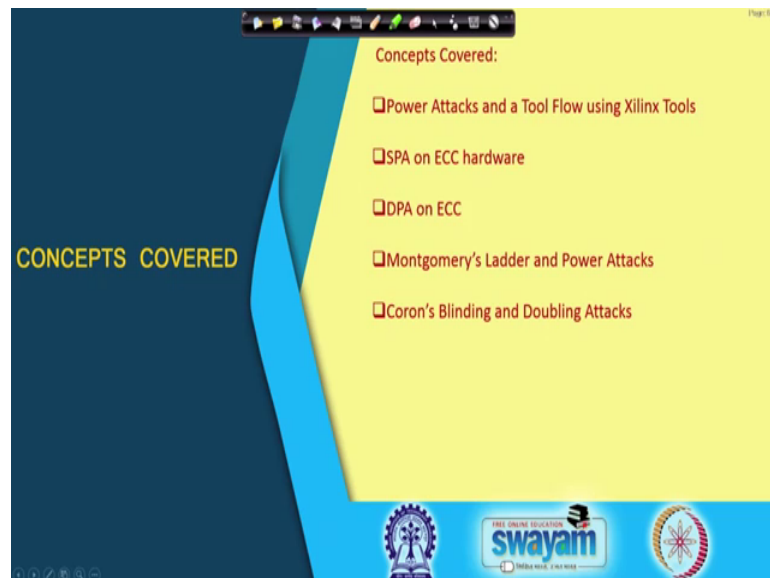


**Hardware Security**  
**Prof. Debdeep Mukhopadhyay**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 36**  
**Power Analysis – XII**

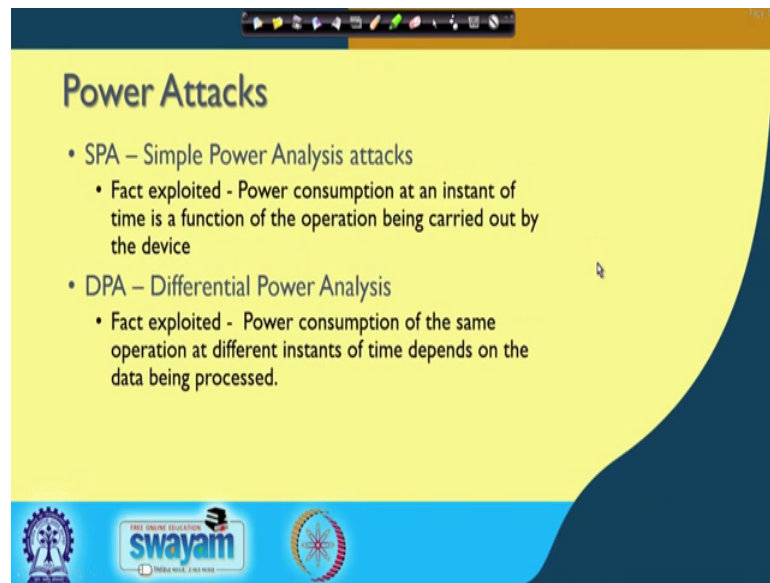
So, welcome to this class on Hardware Security. So, we shall be continuing our discussions on Power attacks. In fact, today we shall be looking the effect of power attacks on a new on essentially a different kind of primitive like we have seen block ciphers, we have seen stream ciphers.

(Refer Slide Time: 00:30)



In today's class, class we shall be looking an the effect of power attacks on elliptic curve cryptographic hardware. Like we have already discussed about the ECC hardware. We shall be trying to look at you know like the effect of SPA and DPA on ECC and also in particular we shall be talking about the structure which is called as Montgomery's ladder to implement the elliptic curve and we shall be discussing in the context of power attacks. Finally, we shall be discussing about Coron's blinding technique and doubling attacks.

(Refer Slide Time: 00:59)



The slide is titled "Power Attacks" and is presented on a yellow background with a dark blue footer. At the top, there is a navigation bar with various icons. The main content consists of two bullet points:

- SPA – Simple Power Analysis attacks
  - Fact exploited - Power consumption at an instant of time is a function of the operation being carried out by the device
- DPA – Differential Power Analysis
  - Fact exploited - Power consumption of the same operation at different instants of time depends on the data being processed.

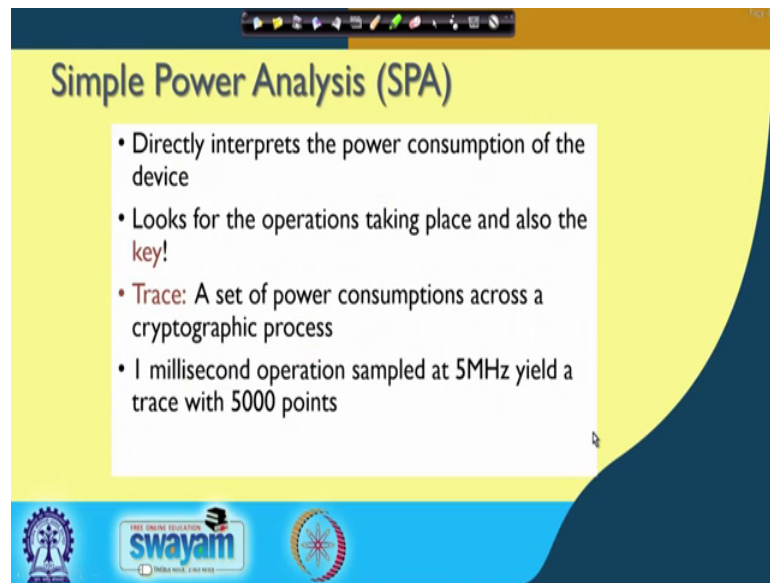
The footer contains three logos: the Indian Institute of Technology (IIT) logo on the left, the "swayam" logo in the center, and another circular logo on the right.

So, before we are right, I mean let us quickly recapitulate about power attacks. We already discussed this in several contexts like there are two different broad categories of power attacks, one which is called as simple power analysis, other one which is called as differential power analysis. In SPA the fact that we were trying to kind of exploit is that the power or the power consumed depends upon the underlying operation which is being carried out.

For example, in the context of elliptic curves the power consumed for doubling operation is different from the power consumed for addition operation. So, if we can you know distinguish this in the power trace if we can observe this distinction in the power trace then that can leave to trivial leakage of the secret key.

On the other hand, right, DPA or differential power analysis is more sophisticated because it basically tries to exploit the fact that power consumed of this is essentially power consumed of the same operation is actually dependent upon the underlying data. And therefore, the dependence being more fundamental even if there is a protection against simple power attack that can break against differential power attacks.

(Refer Slide Time: 02:01)



The slide is titled "Simple Power Analysis (SPA)" and contains the following text:

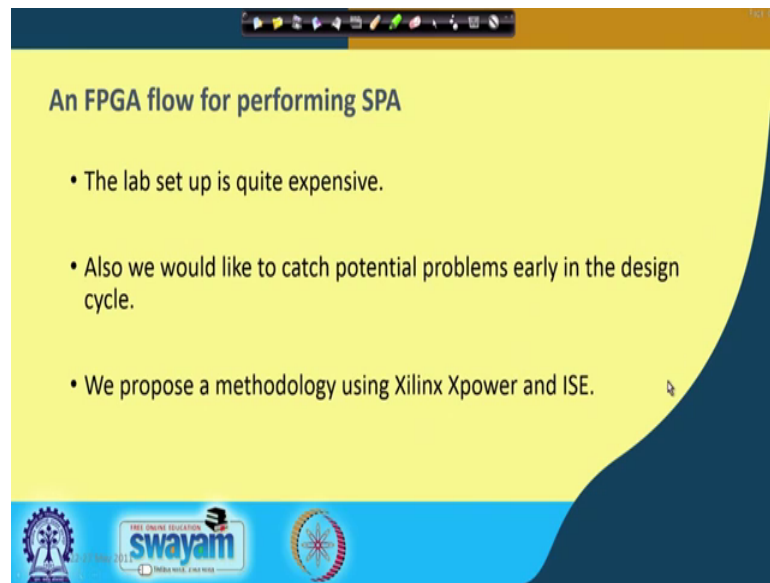
- Directly interprets the power consumption of the device
- Looks for the operations taking place and also the **key!**
- **Trace:** A set of power consumptions across a cryptographic process
- 1 millisecond operation sampled at 5MHz yield a trace with 5000 points

At the bottom of the slide, there are three logos: the Swamiji logo on the left, the "swayam" logo in the center (with "FREE ONLINE EDUCATION" above it and "INDIA 2020" below it), and the Ashoka Lion Capital logo on the right.

So, therefore, we have already studied that in SPA, right. We basically work with like maybe 1 trace or may be 2 trace or may few 100 traces. And the trace is nothing but a set of power consumption across cryptographic operations like when we take we will be execute the cryptographic process, right, we are basically trying to measure the power consumed of the total device.

Not only like of the specific operation, but essentially of the total device and that essentially is what we called as the power trace. So, we have already discussed about the power attacks setup and how do we you know like accumulate this power observations.

(Refer Slide Time: 02:34)



**An FPGA flow for performing SPA**

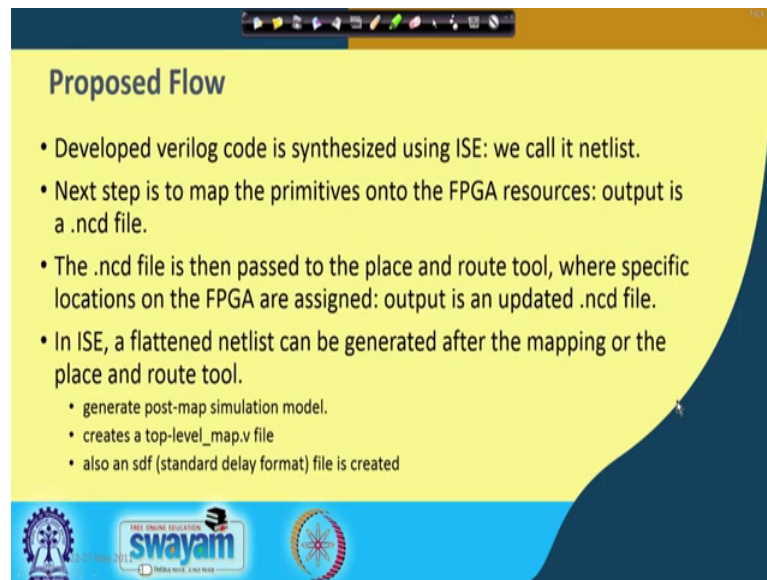
- The lab set up is quite expensive.
- Also we would like to catch potential problems early in the design cycle.
- We propose a methodology using Xilinx Xpower and ISE.

swayam  
INDIA WISE, LEAD WISE

And, but today, right as we have already kind of probably understood is that in general the lab setup is pretty expensive. So, here I shall be talk talking about in FPGA flow which we you can easily setup for performing you know like fairly reasonable power analysis ok. So, this tool is basically based upon you know the Xilinx environment, but pretty much you can also try to find out equivalent components in other cat flows ok.

So, without loss of generality let us try to understand I proposed methodology which we can build up using this a tool which is called as Xilinx Xpower. So, Xilinx Xpower and ISE are very you know popular tools of the Xilinx of essential essentially is popular FPGA tools and we try to develop a methodology using such kind of cat tools.

(Refer Slide Time: 03:21)



### Proposed Flow

- Developed verilog code is synthesized using ISE: we call it netlist.
- Next step is to map the primitives onto the FPGA resources: output is a .ncd file.
- The .ncd file is then passed to the place and route tool, where specific locations on the FPGA are assigned: output is an updated .ncd file.
- In ISE, a flattened netlist can be generated after the mapping or the place and route tool.
  - generate post-map simulation model.
  - creates a top-level\_map.v file
  - also an sdf (standard delay format) file is created

swamyam  
FREE ONLINE EDUCATION  
MEDIA WISE. LEARN WISE.

So, the idea is that first we develop a Verilog code as we have seen that we have writing the Verilog code forward design. We synthesize the Verilog code using ISE tool. And this particular after we have synthesized the Verilog essentially gets translated into a form which is called as the netlist ok. So, next what we do is we take the netlist and in the FPGA flow we basically map it into a FPGA resources the tool or the ISE tool does it for us and the output is something which is called as a dot ncd file ok.

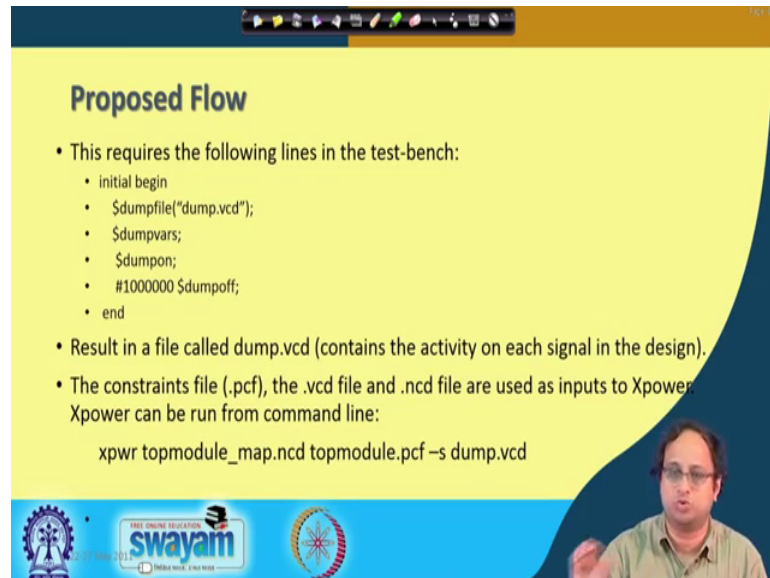
Now, this dot ncd file is then pass to the place and route tool again which is a part of the Xilinx tool chain where specific locations on the FPGA are assigned and the output is an updated dot ncd file. So, you get an improved version of the dot ncd file.

In ISE a flattened netlist can be generated after the map of the place and route tool. So, you just need to give this commands like generate post map simulation model, it creates a top level map underscore map dot v file. So, again this is the top level flattened netlist which means like all the higher keys like, if you have when you design several modules all the modules are kind of flattened to have the same level of abstraction, a single level of abstraction and then you basically create a dot sdf file. So, dot sdf for, sdf stands for the standard delay format. So, basically this particular file contains the delay or the parasitic delays of your circuit.

So, you basically have you know like the various gate delay information, you have the routing delay information. So, pretty much the tool whatever the tool is kind of

predicting about your design essentially is kind of noted down or kind of recorded in the dot sdf file.

(Refer Slide Time: 05:06)



**Proposed Flow**

- This requires the following lines in the test-bench:
  - initial begin
  - \$dumpfile("dump.vcd");
  - \$dumpvars;
  - \$dumpon;
  - #1000000 \$dumpoff;
  - end
- Result in a file called dump.vcd (contains the activity on each signal in the design).
- The constraints file (.pcf), the .vcd file and .ncd file are used as inputs to Xpower. Xpower can be run from command line:  

```
xpwr topmodule_map.ncd topmodule.pcf -s dump.vcd
```

Logos for Swamyam and other institutions are visible in the footer.

So, now what we do is we basically write the or you know like add these to our test bench we know that a test bench is a essentially a component that we have in Verilog through which you basically gives simulate toward design. You basically give inputs to the design and verify whether the output is as expected or not. So, what we add is, basically we add these lines like initial begin and then inside that we write dot basically we dump the dot dump you know dot vcd file.

So, vcd file basically is nothing but it contains the activity of each and every signal inside your design ok. So, it basically contains the switching of your design. So, you can kind of try to understand that the vcd file has got you know like kind of it measures the power consumption of the design, because you know like there are switchings which are happening, and those switchings are recorded in the dot vcd file.

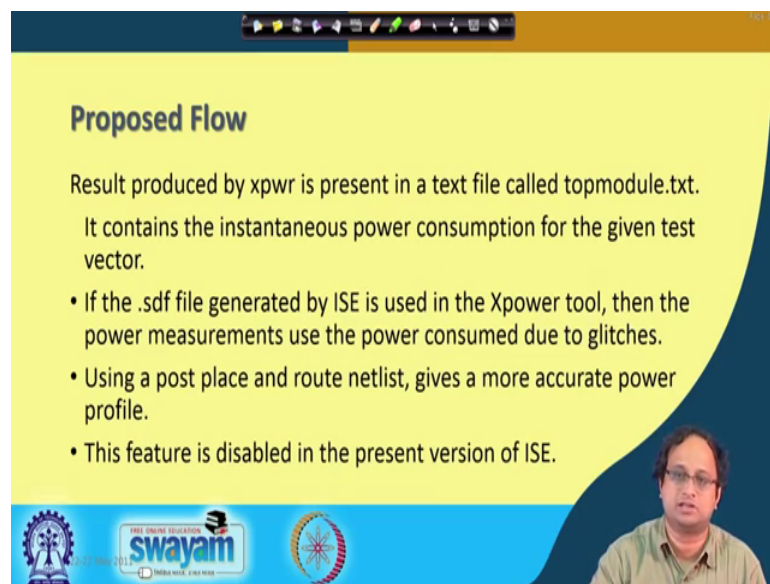
So, therefore, right these are some commands which may work to download the dot you know dot vcd file. And once you have this file, so you know like which basically contains the activity and therefore, you basically have a measure of the switching activity which is kind of a measure of the dynamic power of your circuit and then what you can do is you can you know like add a constant file which is called as dot pcf file and the dot

ved file and the dot ncd files are used inputs to the Xpower tool and the Xpower can be run by in the from the command line.

So, therefore, if you run it right then you will get a kind of profiling of power with time. So, basically you will get a dynamic simulation of the power consumption. So, you do not get a monolithic value of power at the end, but you kind of get you know like power simulation which you can use as a representative of your power trace ok. So, this can be a very you know I would say the e house arrangement of doing power analysis where do not have access to sophisticated tools.

Even otherwise, right, even if you have a access to sophisticated tools from here you can try to understand what are the points of interest of your circuit and you can try to kind of focus on your real power trace at those points ok.

(Refer Slide Time: 07:05)



**Proposed Flow**

Result produced by xpwr is present in a text file called topmodule.txt. It contains the instantaneous power consumption for the given test vector.

- If the .sdf file generated by ISE is used in the Xpower tool, then the power measurements use the power consumed due to glitches.
- Using a post place and route netlist, gives a more accurate power profile.
- This feature is disabled in the present version of ISE.

The slide footer includes logos for Swayam (Free Online Education) and the Ministry of Education, Government of India, along with a small video inset of a man in the bottom right corner.

So, therefore, right this proposed this is you know like basically the what it does is the this result which is produced by Xpower is present in a text file which is say called top module dot txt. Basically, contains the instantaneous power consumption for the given test vector. If the dot sdf file is also used in this simulation then the power measurements use also the power consumed due to the glitches.

So, it is pretty accurate in that sense, and pretty useful. So, using a post place and route netlist gives a more accurate power profile and this feature is unfortunately disabled in

the present version of ISE. So, if you can go back and you know like get some of older versions or maybe there are variants of this tool which you can try to still work with.

(Refer Slide Time: 07:46)

The slide, titled "SPA of our ECC Processor", features a yellow background with a blue header and footer. The header contains a navigation toolbar. The main content area is divided into two sections. On the left, a list of bullet points describes the state machine's operation. On the right, a state machine diagram shows 12 states arranged in a circle, with 4 blue states (D1-D4) and 8 red states (A1-A8). A key bit input is shown at the bottom left, with arrows indicating transitions between states. The footer includes logos for Swamyam and other educational institutions.

- The state machine for the scalar multiplication in the ECCP has 12 states.
- 4 states (D1 ··· D4) for doubling and 8 states (A1 ··· A8) for addition.
- Each iteration in the scalar multiplication handles a bit in the key starting from the most significant one to the least significant bit.
  - If the key bit is zero a doubling is done and no addition is done.
  - If the key bit is one the doubling is followed by an addition.
- The dissimilarity in the way a 1 and a 0 in the key is handled makes the ECCP vulnerable to side channel attacks.

So, what we will do here is that we would be again kind of look back at our elliptic curve processor which we already design in one of our previous classes and this is a state machine that we left at which we left our elliptic curves hardware. So, remember that we have some initial 3 clock cycles, but most of most importantly, right, when then scalar starts to get processed then you basically try to see whether your key bit is 0 or whether your key bit is 1.

So, if your key bit is 0 then you just do the doubling operation which is shown by the blue circles; that means, you pass it through like 4 clock cycles and then you basically get out of this point and then you again come back to this state. So, basically you go like D 1, D 2, D 3, D 4 and then again get back to D 1 ok. On the other end right, if your key bit is one then you basically go through the doubling states then you go through 8 distinct states in addition in our state machine and then you again go back to D 1 ok.

So, therefore, in this particular cycle as you can see there is a dependence, right, in this in how you are basically migrating through the final state machine depending upon your scalar bit ok. And that essentially is one of the reasons why we have a dissimilarity and therefore, we will see that this particular design is basically vulnerable against simple power attack ok.



So, the dissimilarity in this way right, is basically in the way that a 1 and a 0 in the keys handled makes you know like the ECCP or the elliptic curve processor vulnerable to side channel attacks or basically simple side channel attacks to be more specific ok.

(Refer Slide Time: 09:16)

**Leakage of the Scalar**

- The **duration** of an iteration depends on the key bit.
  - A key bit of 0 leads to a short cycle compared to a key bit of 1.
  - Thus measuring the duration of an iteration will give an attacker knowledge about the key bit.
- Each state in the FSM has a unique **power consumption** trace.
  - Monitoring the power consumption trace would reveal if an addition is done thus revealing the key bit.

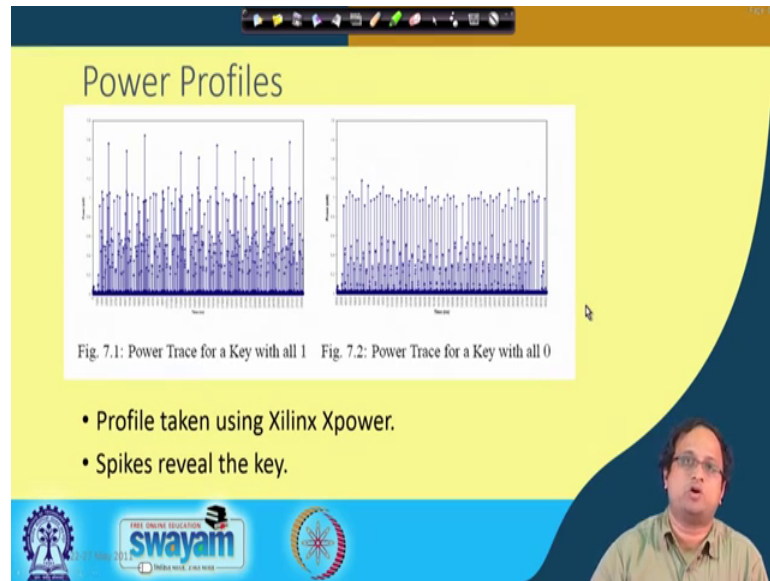
So, let us see. So, therefore, right I mean there are several leakage which are possible because of this observation that is you know like for example, even from the perspective of timing ok. If you measure the duration of an iteration then you will see that the duration of the iteration actually depends upon the key bit, right, because if the key bit is 0 then you have a shorter cycle compared to you know like a key bit of 1, where are doing both doubling and addition. And when the key bit is 0 you are only doing doubling operation where you are spending 4 clock cycles whereas, right if you are doing addition; that means, if the key bit is 1, your 4 plus 8 that means, 12 clock cycles in this particular operation.

So, thus measuring the duration of an iteration will give an attacker knowledge about the key bit. So, if the attacker is able to understand this duration then it gives an easy way of getting the key because it there is a clear dependence on that. There is a clear leakage, ok. So, each state in the finite state machine also has a unique power consumption trace; that means, as I said that as we discussed right, each of the each state in the final state machine is doing a specific setup operation, right. And as we discussed, right one of the fundamental premise of simple power attack is that power essentially depends upon the

underlying operation. So, therefore, every state has got a unique you know like footprint in the power trace ok.

So, monitoring this power consumption trace therefore would be interesting to look at and we will see how we can use it to understand the secret key bits ok.

(Refer Slide Time: 10:39)

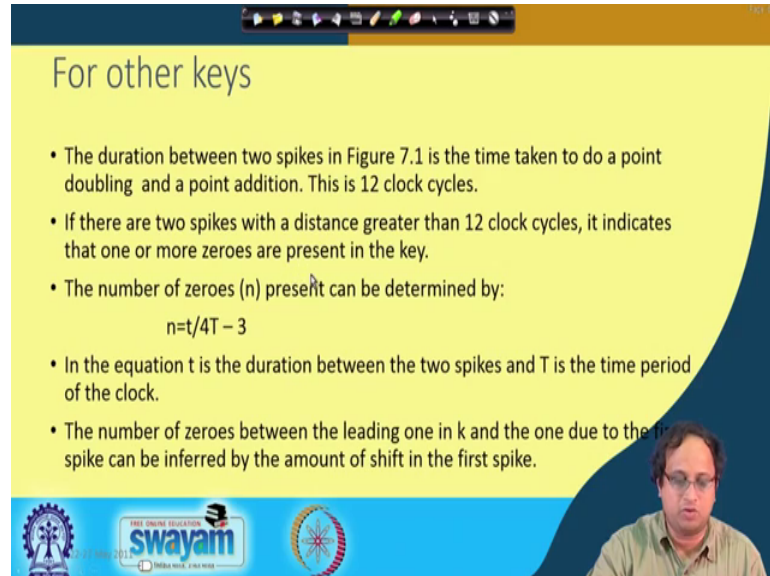


So, we will take this example and we will consider some power profile. So, these are pleased power profiles and got by our tool chain ok. So, you can try to do the same thing in actual traces as well ok. So, you will not get exactly similar profiles, but you will get your like the basic idea will still be the same ok. So, here is a power trace which has been acquired with an all 1 key; that means, all the key bits are 1, whereas, in this case, right it is all 0 which means like there is a MSB which is 1 followed by all 0s ok. So, that means, here I am only doing doubling operations whereas, here I am doing doubling addition, doubling addition, doubling addition and so on.

So, now you can see that there is a clearly there is a distinction between these two power profiles. You get spikes here and these spikes are additional power consumptions which are due to the you know like the successive ones that you have in your secret key bit ok. So, clearly let us by seeing up our power profile you can distinguish between these two keys ok. And that essentially is a leakage, because ideally, right the power trace should not speak anything about the key, and it should be silent about the key. So, immediately

you can understand that this is a vulnerable design. If somebody is able to observe this power profiles then there is a sneak that he gets he or she gets into the secret key ok.

(Refer Slide Time: 11:49)



The slide is titled "For other keys" and contains the following text:

- The duration between two spikes in Figure 7.1 is the time taken to do a point doubling and a point addition. This is 12 clock cycles.
- If there are two spikes with a distance greater than 12 clock cycles, it indicates that one or more zeroes are present in the key.
- The number of zeroes (n) present can be determined by:  
$$n = t/4T - 3$$
- In the equation t is the duration between the two spikes and T is the time period of the clock.
- The number of zeroes between the leading one in k and the one due to the first spike can be inferred by the amount of shift in the first spike.

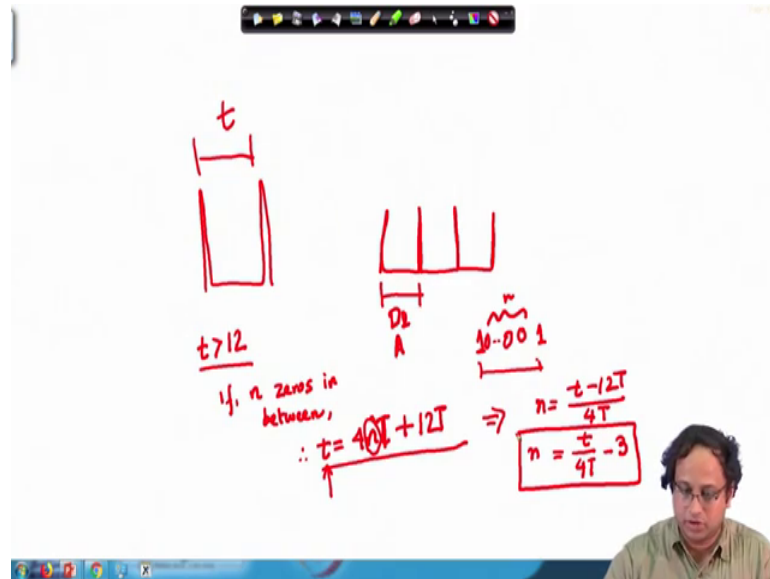
The slide also features a video inset of a man speaking in the bottom right corner and logos for "swayam" and "INDIA RISE, EDUCATION RISE" at the bottom.

So, now, what we will see whether we can develop a systematic way of getting the keys, ok. And this is basically again a very simple technique or trick that we can apply particularly for the design that we discussed ok. So, we can of course, develop more sophisticated techniques where we do you know template matching, power pattern matching and so on basically. But we will see that even a simple trick can work ok.

So, basically what we observe is that the duration between two spikes in figure 7.1; that means, figure 7.1 is essentially this figure ok, where essentially you have got some spikes over. So, the reason why you have you know like these spikes is because of addition ok. So, basically, right be the duration between these two between these two spikes is the time which is taken to do a point doubling and do a point addition because you are doing both doubling as well as addition and that is 12 clock cycles, 4 plus 8, that is 12 clock cycles ok.

So, if there are two spikes with a distance which is greater than 12 clock cycles therefore, it means that there are (Refer Time: 12:51) 0s in between, right. So, basically what I am trying to say is that suppose you have got you know like spikes like ok. So, let me clear this up one second.

(Refer Slide Time: 13:10)



Suppose you have got spikes like say these are your spikes, which were getting in your power trace and these basically are comprising of you know like the instance between doubling and addition that is you are doing both double, you are doing both double and you are also doing addition ok. So, therefore, right for double you have got 4 clock cycles whereas, for addition you have got 8 clock cycles ok.

Now, imagine that you basically observe this, you basically get two spikes, right and you basically record that the time is  $t$  in between these two spikes ok. So, if  $t$  is equal to like 12 clock cycles you mean it is know that is like two once one after the other; that means, there are no 0s in between, right. So, on the other hand, right if  $t$  is more than 12; that means, if you know like the  $t$  is greater than 12 then you can understand that if there are you know like if there are  $n$  zeros in between like  $n$  zeros in between ok, so every 0 will correspond to only a doubling operation, right.

So, therefore, you can write that  $t$  will be in that case equal to  $4nT$ , because you know like I mean will be equal to  $4nT$ . So,  $T$  means the time period or the clock time period, ok so  $4nT$  plus 12 ok. So, what I mean to say I am talking about a key sequence where there is a two 1s and there are  $n$  0s in between ok. So, there is like 0 0 0 0 and there are  $n$  0s ok. So, that means, if this is the time instance or time difference between them then all the 0s will be doing a doubling operation, right. And therefore, every doubling will take 4 into  $T$  because  $T$  is the time period of my clock, so it will take 4 into  $T$  and there are  $n$

0s. So, it will take 4 into n into T ok. So, therefore, I will get a simple equation like this ok.

And if I know the clock time period then if I know the value of t then I can calculate the value of n which is the number of 0s in between, right. So, basically, I can pretty much just write this as you know like n would be equal to t minus 12 divided by 4 T and that is nothing but t by 4 T minus. So, this will be plus 12 T because there are 2 you know like; so you will have 12 T and therefore, t by 4 T minus 3 ok. So, therefore, right this is my this is the simple equation that we will be trying to adopt in order to get the number of 0s in between ok.

So, now let us try to get back into a our presentation and see that this is the formula that we use, t equal to n equal to t by 4 T minus 3 ok. So, therefore, this gives me the number of 0s which are present and between two 1s ok. And the number of 0s between the leading one in k and the one due to the first spike can be inferred by the amount of shift in the first spike ok. The first spike you can get by you know the number of 0s between the leading one in k and the one because remember that we are always assume in our algorithm that the first there is the leading one, right and we are basically trying to process from the next from the from the bit after that ok.

(Refer Slide Time: 16:32)

Example

Fig. 7.3: Power Trace when  $k = (B9B9)_{16}$

Can you read the key from the trace?

swayam

THE OPEN EDUCATION

So, let us just try to apply this formula and see what how we get. So, for example, this is an unknown key trace ok. So, we have got an unknown observation and the question is

like can we read the key from the trace, because that is what SPA tries to do. So, therefore, what we will do now is that we will see these spikes and we will time these spikes basically, this will we will try to observe like what is the time difference between these spikes ok. Suppose the attacker gets these spikes and looks at the x axis and finds out the time distance between the spikes ok.

(Refer Slide Time: 17:04)

**The other key bits**

- There are 9 spikes indicating 9 ones in the key (excluding the leading one).
- The clock has a period  $T = 200\text{ns}$ .
- The first spike  $t_1$  is obtained at 3506th ns.
- If there were no zeros before  $t_1$  the spike should have been present at 2706th ns.
- The shift is 800 ns equal to four clock cycles. Therefore a 0 is present before the  $t_1$  spike.

Fig. 7.3: Power Trace when  $k = (D91D9)_{16}$

Table 7.1: SPA for the key  $(D91D9)_{16}$

$i$	$t_i - t_{i-1}$	$n$	Key inferred
1	-	-	01
2	2400 ns	0	1
3	2400 ns	0	1
4	4000 ns	2	001
5	2400 ns	0	1
6	3200 ns	1	01
7	2400 ns	0	1
8	2400 ns	0	1
9	4000 ns	2	001

And you can see very interestingly there are 9 spikes, here, we can read out the there are like 1, 2, 3, 4, 5, 6, 7, 8 and 9. So, there are 9 spikes here. So, 1 is below actually ok. So, there are 9 spikes which you can observe here. Of course, you can understand that there will be noise and sometimes you know like you will probably miss one or two spikes, but here you can at least see few of them are pretty distinct ok.

Like at least these ones are quite distinct. So, what I do is therefore, what I observe first is that the distance between the spikes are not same. So, you can see like the distance between these two spikes and the these two spikes and maybe these two spikes are quiet different like this is like quiet wide apart, right. So, therefore, you are expecting that there are more 0s in between ok.

So, therefore, we find that in our case the clock time period is  $T$  equal to 200 nanoseconds. So, this is essentially the time that that the clock time of my processor, of my elliptic curve processor and the first spike  $t_1$  is obtained that say some 3506th nanosecond ok. So, if there were no zeros before  $t_1$  then the spike would have been

present as 2706th nanosecond ok. So, the idea is that there is a shift of 800 nanosecond and the reason why you have got this shift is because there are 4 clock cycles, the shift is 800 nanosecond which is equal to 4 clock cycles therefore, there is a 0 which is present before the  $t_1$  spike ok.

See, right because if there is one 0 present then for that one 0 you are doing a doubling operation ok; that means, even before this spike has come up, right there are doubling operations ok. So, therefore, there are 4 doubling operations ok. So, basically what I am trying to say is that it is quite easy to obtain the time period between two edges, like successive addition operations, but how do you get the number of 0s before you get the first addition done ok. So, therefore, in this case we observe that what you should start at 2706 and it has starts from 3506, so there is a delay, there is a shift and that shift is because of a doubling operation ok.

So, now, if you observe the clock cycles like 1, 2, 3, 4, 5, 6 and so on you will see that the first time the difference here is 2400 nanosecond and 2400 nanosecond if you plug in to that formula that we discussed, right. This is the time interval between two spikes so that means, if I plug in like you know like. So, this is the time, 2400 if you plug in here like  $t$  by  $4T$  minus 3 then this is 2400 divided by 4 into 200 and that essentially will be 2400 by 4, right is 600 divided by 200 is 3. So, 3 minus 3 is 0 ok, so, therefore,  $n$  stands to be 0.

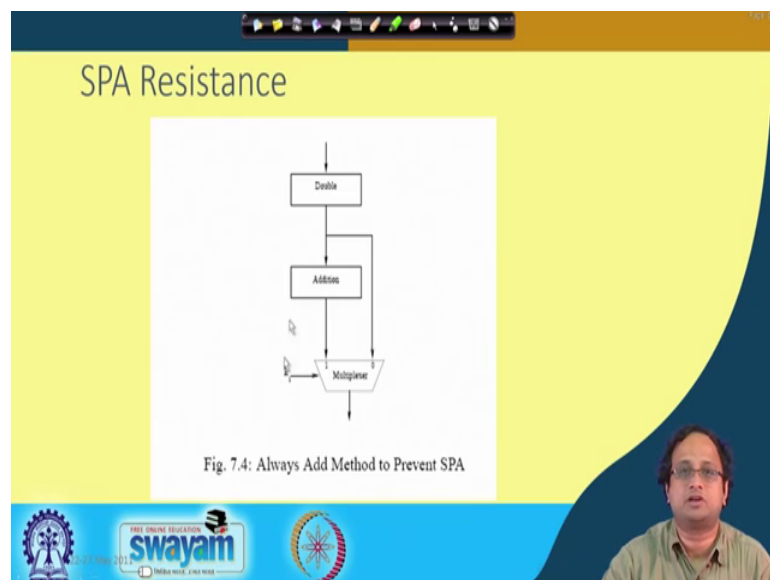
So, therefore, we do it here and we find out that  $n$  is 0. So,  $n$  is 0 means there is no inter between you know there is no in between 0 and therefore, you can infer that my initial key was 0 1 that is because of the first shift that I told about like we have got one doubling followed by addition.

But then the next one, right there is no 0, that means, this key is followed by this 1 is followed by 1 ok. Likewise for the second time interval you get 2400 which again stands for 0 which means this 1 is followed by 1 ok. Then you get the next time difference is 400 nanosecond. So, 400 nanoseconds stands for 2, so you can again plug in the same form into same formula and you will get  $n$  as 2. So,  $n$  as 2 means there is one is followed by two 0s and then there is a 1 ok. So, that means, right now you can you know like pretty much read the entire secret key in this way and in find out that the key is 01, 1, 1,

001, 1, 01, 1, 1, 001 and this stands for nothing but B9 B9 and which is the actual key which has been used in this elliptic curve operation ok.

So, therefore, right, I mean you can see that you know like SPA works in this case, because SPA essentially is a pretty powerful tool because if because many of the our designs, right if I do it as a beginner then I will probably not take care of this kind of vulnerabilities and therefore, our design although it will be functionally correct will be prone to this kind of attacks ok. So, therefore, what we need to of course look into is how we can protect against this kind of vulnerabilities.

(Refer Slide Time: 21:30)

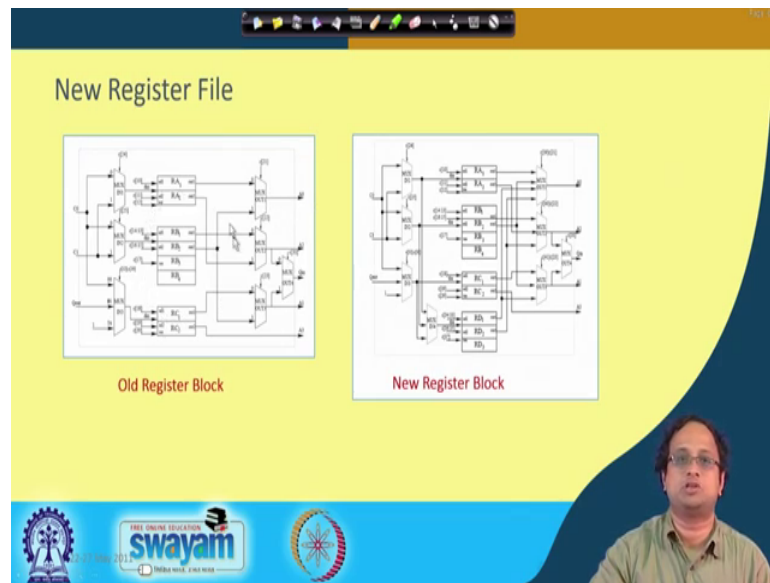


And therefore, the first resistance that probably we can think of is that let us try to develop an architecture where we do always add; that means, addition is always done, rather than in the previous architecture where we were doing a condition of addition only when the key bit was one I was doing an addition, but now I will take change my strategy and I will say that I will always do a dummy addition at least ok. So, therefore, what I do is it can be represented by the simple circuit.

So, I do a doubling, and then I do an addition, but if the key bit is 1, right then I will pass this addition I will multiplex this addition; but the key bit is 0 then also I am doing an addition, but I am not using this addition rather I am passing this input to the output ok. So, this is essentially nothing, but just simple a multiplexer idea it, but essentially this should prevent our SPA ok.



(Refer Slide Time: 22:23)

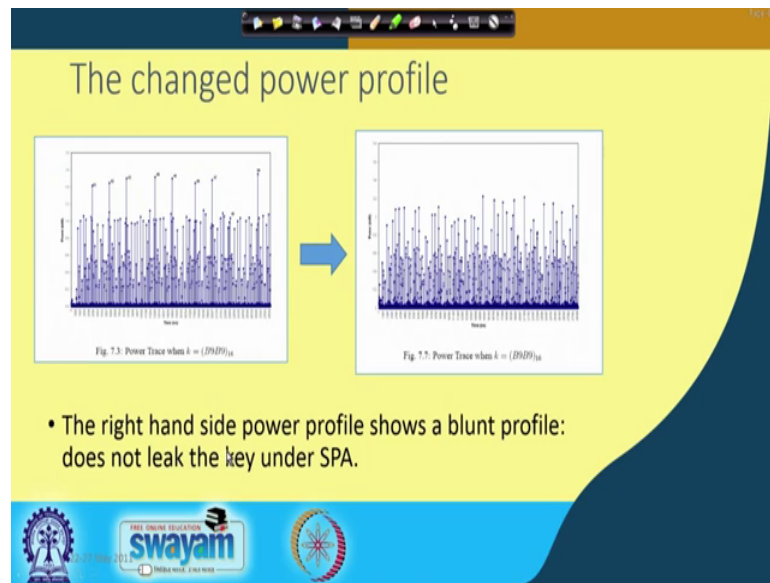


So, therefore, now again let us look back at our architecture. So, this was our original old register block that we kind of design. In order to make this modification we have to change in it in this data path ok. So, what we do here is now remember that we are free register banks, now along with this free register banks we add an additional register bank. You can see that now there are 4 register banks. So, this register bank is basically storing the result of after the doubling operation. So, whatever is coming after the doubling operation I am storing that in a temporary register file.

The idea is that if the secret key bit is 0, right then essentially I will take only the output of from these register file ok. So, therefore, the addition which is happening even after even if the key bit is 0 is something which I need to ignore because that is a dummy addition operation that is if I take that then I will get a wrong result ok. So, therefore, now you will see that because of this you know like register that we have introduced here dummy register you can say which is storing the value of your or rather let me say it is a temporary register which is storing the value of your doubled operation or doubled output.

Essentially needs also you know multiplexer at the input and also needs few additional lines to get the result out in the output port ok. So, therefore, these multiplexers are also little bit widened up because now you need also this these register outputs to be you know like to be channeled outside.

(Refer Slide Time: 23:48)



So, therefore, right there is a cost that you need to pay. I mean essentially but rather same time, right if you observe the power profile now with this change, you will see that those clearly distinct spikes are now not present ok. So, pretty much you will see that now every you get additional or every time instance and therefore, this is this you can see that even if I change the key the pedal will look roughly the same.

So, at least to the visible eye you will not be able to kind of easily demark it and try to see that there is a dependence ok. So, therefore, the right-hand side power profile shows a blunt profile and therefore is more registered against simple power attacks.

(Refer Slide Time: 24:22)

Security has a price!

The FSM has been modified to perform always both doubling and addition.

The total number of clock cycles is always 12.

Fig. 7.5: FSM for SR-ECC

Processor	Device	Slices	Frequency	Clock Cycles
ECCP	Xilinx Virtex 4 (XC4VFX140)	21852	64.46MHz	1883
SR-ECCP	Xilinx Virtex 4 (XC4VFX140)	23511	56.46MHz	2811

What is the price that you pay? Of course, you have to pay a price because there are some additional hardware material here we have used. So, in this case this is a transformed FSM. So, now, you see the those conditional lines are gone, rather you know like whenever you get into this doubling chain you ensure that you would do the addition and then give get out only then you are not using that result ok, you are using the result of the output of the doubled stage ok.

So, the total clock cycle is therefore, always 12. So, in a way this is also timing attack resistance or timing resistance because it is always taking the same number of clock cycles to give you the output. So, the time essentially is not leaking about the key. So, if you see the footprint on hardware you will see that there is a block that means, there is a diminish in the frequency, there is an increase in the slice, there is a you know like the increase in the number of clock cycles, but that is the price that you pay for this security, for security.

(Refer Slide Time: 25:13)

**Montgomery Ladder**

Algorithm 2.4: The Montgomery ladder for elliptic curve scalar multiplication.

**Input:** An integer  $d \geq 1$  and a point  $P$  on elliptic curve.  $d = \sum_{i=0}^{k-1} 2^i d_i$ .

**Output:**  $dP$ .

```
Q1 ← P and Q2 ← 2P.
for i from k-2 downto 0 do
  if d_i = 1 then
    Q1 ← Q1 + Q2 and Q2 ← 2Q2.
  end
else
  Q2 ← Q1 + Q2 and Q1 ← 2Q1.
end
end
return Q1.
```

$k=9=(1001)$   
 $Q1=P, Q2=2P$   
 $k2=0 \Rightarrow Q2=3P,$   
 $Q1=2P$   
 $k1=0 \Rightarrow Q2=5P,$   
 $Q1=4P$   
 $k0=1 \Rightarrow Q1=9P,$   
 $Q2=8P.$

Result is in Q1.

swamyam  
 FREE ONLINE EDUCATION  
 MEDIA WISE. TIME WISE.

So, now we will see how we can you know like; so now, we will basically consider about how we can make it protected against further power analysis. For example, we will see that even though it is protects against simple power attacks this is not secured against differential power attacks, which basically tries to exploit the fact that power consume depends not only upon the operation, but also for a same operation it depends upon data, and therefore, this design would be still vulnerable against DPA. So, in order to protect against DPA we need to do something more. So, that we will see in the next class.

So, thank you for your attention.