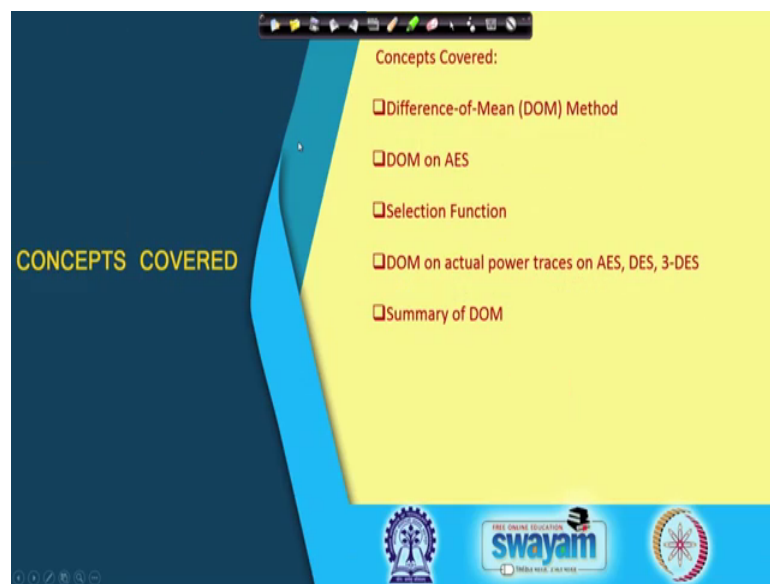


**Hardware Security**  
**Dr. Debdeep Mukhopadhyay**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 26**  
**Power Analysis- II**

So, welcome to this class on Hardware Security. So, we shall be continuing our discussion on difference of mean as we have started in the last class.

(Refer Slide Time: 00:22)



So, in particular today we shall be taking a more deeper look into how DOM works and can be applied on standard block ciphers like AES. I shall be explaining or formalizing the selection function in more details. And, then we shall be seeing some snapshots or case studies or how on what are the results of DOM being applied DOM being applied in actual power traces on AES, DES and 3 DES. And then I will summarize our discussions on difference of mean.

(Refer Slide Time: 00:48)

The slide is titled "DPA Procedure for AES" and lists the following steps:

1. Make power consumption measurement of about 1000 AES operations, 100000 data points / curve, (Ciphertext, Curve,)
2. Assume a key for an S-box of last round
3. Calculate last round S-box first bit input for each ciphertext using the assumed key
4. Divide the measurement into 2 groups (output 0 and 1)
5. Calculate the average curve of each group
6. Calculate the difference of two curves
7. Assumed correct key → spikes in the differential curve
8. Repeat 2-7 for other S-boxes

The slide also features a video feed of a presenter in the bottom right corner and logos for "swayam" and "INDIA WISE, LEAD WISE" at the bottom.

Let us see how the DPA can be adapted for AES right which is essentially a standard cipher and we have already seen how the AES design works? So, imagine that you have AES hardware and what we try to do is we basically observe now the power consumption for say 1000 AES encryptions. And, imagine that there are something like 10 to the power of 5 data points per curve.

So, this is an array in the form of a double like along with the cipher text, I also observe the power trace which is called as the curve. So, this we kind of indicate by the double Cipher text i comma Curve i to indicate that along with the cipher text you are also observing the corresponding power trace ok.

So, now we assume a key for S-box of last round of the last round. So, note that in the last round key right you have got 16 S boxes and the total key has got a dimension of 128 bits ok. Another as said previously also this attack is a is essentially a divide and conquer attack, which means you do not guess the entire key together. Because, then it will be like a brute force attack, but rather you try to do the attack part by part.

So, it basically you guess parts of the key like in this case it is a byte of the key which you guess. So, I guess a one of the key bytes, which I want to kind of determine and I guess that. So, in that case I guess one byte of the key; that means 8 bits of the key. So, now, when you essentially guess that you essentially can calculate the last round first bit input of each cipher text using the assumed key. So, note that you are observing the last

round S-box and as we will see later on is that the of the selection function, which we basically determine the last round since you have in the last term there are 16 S-boxes ok.

And, you basically target any one of these S-boxes depending upon your diffusion function. And, then you basically once you target that S-box you know that there are 8 bits which are going as an input to the S-box. So, your target bit can be can be any one of them, it can be the 0 th bit, can be the first bit, second bit or the 7 th bit which is like all 8 bits actually.

And then so, what you do is that you basically you know like you have got the cipher text, you guess the last round key, you go to the input of that S-box and you find out say the LSB of that of that bit ok. And, why you do that you basically use it to partition your traces ok, as we have seen in the previous example also. So, that bit can take either a 0 value or can take 1 value.

So, if it is 0 then you put the power trace into the 0 bit, if it is 1 you put the power trace in to the 1 bit as simple as that. So, therefore, you divide the measurements into 2 groups, depending upon the output 0 and the 1 of your selection function. And, you calculate the average curve of each group. So, each group will have you know the 0 bin will have some power traces we are going into that the one bit will have some power trace which are going into that, you take the average of each group ok.

And, then the idea is that you calculate the difference of the 2 curves with the hypothesis that, if your guess was correct then your differential curve; that means, when you are taking the difference of the averages, will have some place where there is a nonzero spike ok. Note that and if you are observing the differential curve, you are plotting the differential curve with respect to time ok. So, there are different instances of time when the actual computation does not take place ok.

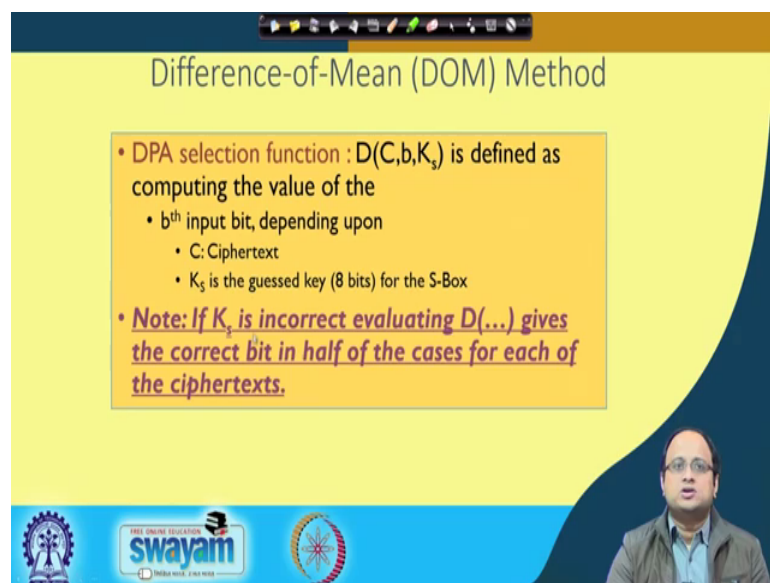
So, in that case even if your guess is correct, that difference right in that particular time instant when that when you are not doing the actual operation will still have a 0 value, will still have a you know like close to 0 value or negligible value, but when you are actually doing the operation on which you are essentially trying to correlate by getting the last bit of the S or you know like getting one of the bits of the input of the S-box, then you will find that there is a nonzero difference ok.

For example, like if you are plotting if you are if you are doing the splitting targeting one of S-boxes right and observed that when you are when you are plotting the power trace; the power trace is plotted with respect to time, but there is only one time instance when that S-box is really coming into play.

At other time instances right, even if you are taking a difference of mean with the correct key guess you will still get 0 difference of mean ok, but when you are actually doing the difference of mean right and you are doing it exactly at the time when that operate when that S-box is coming into play, then you will get a non-zero difference of mean. But for the wrong guesses you will always get close to the close to 0, you know like difference of mean.

Therefore right for the correct guess you will find that there will be difference of mean will have like small values and they repeat ok. And, that essentially will distinguish the correct key from the wrong key.

(Refer Slide Time: 05:45)



Difference-of-Mean (DOM) Method

- DPA selection function :  $D(C,b,K_s)$  is defined as computing the value of the
  - $b^{\text{th}}$  input bit, depending upon
    - C: Ciphertext
    - $K_s$  is the guessed key (8 bits) for the S-Box
- **Note: If  $K_s$  is incorrect evaluating  $D(\dots)$  gives the correct bit in half of the cases for each of the ciphertexts.**

swayam

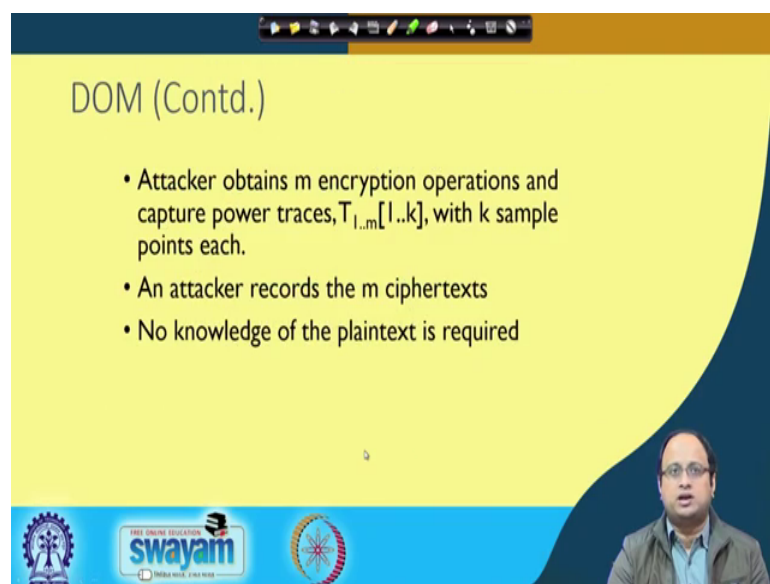
So, we say see some examples and you repeat this operation for other key bytes ok. Therefore, using divide and conquer you basically retrieve the entire secret. So, now, let us look into DOM slightly more mathematical way. So, therefore, we have a selection function as I have tried to kind of motivate. Let us kind of denote that selection function by the function D.

The function  $d$  operates on cipher text there is a target bit denoted as  $b$ . So, it could be the 0<sup>th</sup> bit first bit or any other bit position, and then there is a part of the key which I denote as  $K_S$  which is your guessed key. So, this is used or define you know this is used to basically get the value of the target bit. The idea is that so, as I say that these are  $b$ <sup>th</sup> input bit depending upon the  $C$  Cipher text and also the case which is the guessed 8 bits of the key.

If, your case is incorrect in evaluating  $D$ , then this will still give you the current bit in some cases. Because remember that it is in that case it becomes like a pseudo random function. And therefore, right with actual bit ok, which is there in when in your you know like which is essentially getting computed by your hardware, because your hardware knows the correct secret, because the correct secret is suppose embedded in your hardware.

So, therefore, like even if we are guessing wrong sometimes we will be able to correctly predict it and sometimes it will be wrong. So, therefore, you can imagine that is half of the cases or around half of the cases it will be correct and around half of the cases it will be wrong ok. That is why this bit will then you know like this output of the selection function will therefore, operate like a pseudo random function. But, if your guess is correct then all the time, it will correctly match with your correct you know it will correctly match and correctly predict that corresponding bit.

(Refer Slide Time: 07:41)



DOM (Contd.)

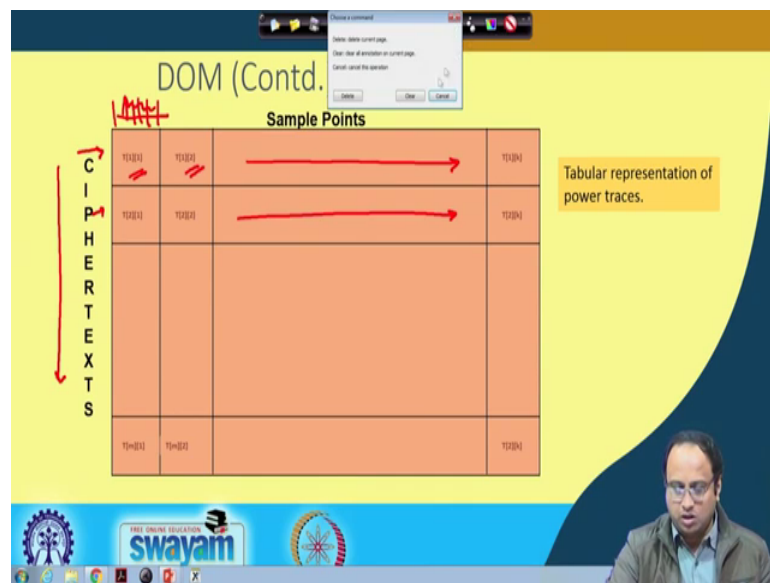
- Attacker obtains  $m$  encryption operations and capture power traces,  $T_{i,m}[1..k]$ , with  $k$  sample points each.
- An attacker records the  $m$  ciphertexts
- No knowledge of the plaintext is required

swayam  
INDIA'S GATEWAY TO KNOWLEDGE

So, therefore, that is a very important point to be noted and essentially is the reason why your attack works. So, what the attacker does now is that attacker obtains  $n$  encryption operations ok. And, captures as I said  $n$  could be like 1000 encryption operations, and captures the power traces in the form of a table.

So, you can imagine that the table is something like denoted as  $T_1$  to  $m$  to  $1$  to  $k$ . So, idea is that you have got some rows and some columns ok. And, the rows essentially are you know like denoting the various power traces or various executions that have taken place whereas; the columns are different time instances of your power traces. So, attacker records the  $n$  cipher text also because cipher text will be used. So, you can basically do the attacker in a setting where no knowledge of the plain text is required. So, it is kind of working as a cipher text only attack.

(Refer Slide Time: 08:28)



So, now they are so, this is an example of how you know like how your power board can look like. So, you can see that this essentially is nothing, but your cipher text ok. The various cipher text that has been observed for every cipher text you see that means, so, every cipher takes suppose this cipher text corresponds to an encryption operation of AES. And, what we have observed right is the corresponding power trace of AES.

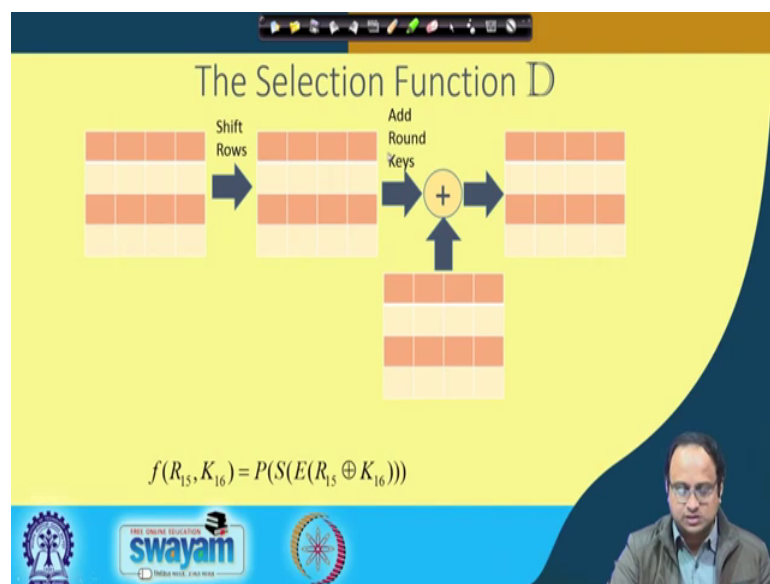
So, how do you kind of plot the power or how do you note the power trace of AES in this table, you basically sample it at different time instances ok. And, then you find out the corresponding real value and that real value is a stored in this table ok. So, imagine this T

1 1 1 stands for the fact that in the first row, at the first time instance when you are sampling it what is the corresponding power value ok. And, likewise you essentially store this in this entire column, entire row ok.

And, likewise you have got different executions like you go to the next cipher text again you take a different power trace and that is also being kind of plotted kind of noted here. So, essentially have kind of you know it represent the power trace in a tabular fashion. So, with this so, this essentially is stored in your computer, I mean is stored is captured by the oscilloscope and then again stored back into the PC where you essentially keep these data ok.

So, now, we are all so, now let us see how we can do the subsequent steps of the attack.

(Refer Slide Time: 10:01)



So, what we do basically here is let us look into AES as a specific example that we are discussing off and imagine that this is the last round of your AES algorithm. So, here you have got so, this imagine is the input of imagine that is the input of your AES basically as we take this slide rather this probably more clear.

(Refer Slide Time: 10:21)

The Selection D

Sub Bytes and Shift Rows

$S^{-1}(C_{10} \oplus K^*)$

$C_{10} \oplus K^*$

Add Round Keys

$+$

$C_{10}$

Guess  $k^*$

0-bin

1-bin

$D(C_{10}, b=0, K_{10}) = S^{-1}(C_{10} \oplus K^*)|_{b=0}$

If the key guess is correct, this matches with the correct value for all ciphertexts collected. However, if wrong it matches roughly half times, assuming sufficiently large number of cipher samples have been collected.

So, you take this so, consider AES 128 ok.

(Refer Slide Time: 10:30)

DPA Mathematically

- Attacker now computes a  $k$ -sample differential trace  $\Delta_D[1..k]$  by finding the difference between the average of the traces for which  $D(\dots)$  is one and the average for which  $D(\dots)$  is zero.

$$\Delta_D = \frac{\sum_{i=1}^m D(C_i, b, K_i) I_i[j] - \sum_{i=1}^m (1 - D(C_i, b, K_i)) I_i[j]}{\sum_{i=1}^m D(C_i, b, K_i) - \sum_{i=1}^m (1 - D(C_i, b, K_i))}$$

1-bin

0-bin

So, so, you consider AES 128 and imagine that. So, this is the last round of AES. So, the input the operation is sub bytes which is the S-box followed by the shift rows. So, imagine that I want to know. So, what do you do in so, let us first will see the just recapitulate the algorithm. You have got you know like this state matrix you perform the sub byte and the shift rows sub byte is the just byte by byte substitution and then you follow it by a shift rows.



Shift rows first row there is no shift, in the second row there is a circular left shift by one position, third row there is a circular left shift by 2 positions and in the final row there is a circular left shift by 3 positions and then you do an add round key. So, basically you XOR this secret key which is again 128 bit key and you get the cipher. So, now, imagine that as an attacker you are interested to know in this byte of the key. Likewise I will recover each of the bytes ok. So, it is that is why it is a divide and conquer attack.

So, what I do is therefore, I take the corresponding cipher text  $C_{10}$  here. And, I XOR them and I come here ok. Then I know that for this guess say, that guess is  $K^*$ , this value says  $C_{10}$  XOR with  $K^*$ , and then I go inverse, that is I take the inverse operation, and I come to this position. You see that there is a shift row operation, because of which this position corresponds to this position ok.

And therefore, what I do is therefore, I take  $S^{-1}$  of  $C_{10}$  XOR with  $K^*$  and I can target any of the bit positions here. Suppose in my selection function here  $b$  is equal to 0; that means, am targeting the LSB of the state. So, therefore, I take this  $S^{-1} C_{10}$  XOR with  $K^*$  and I found out the LSB of this particular state.

So, the idea is that if this particular LSB is 0, then the corresponding power trace that I am capturing right because I am capturing the corresponding power trace also. So, this power trace if this LSB right is essentially 0; that means, if this bit turns out to be 0, then I will put this into the 0 bin. So, I will put this into the 0 bin on the other hand if this turns out to be 1 I will take this part this power trace and put it into the 1 bin that is a simple idea.

So, that. So, now, what I do is I again take the 0 bins and the 1 bins I take the average of each of these bins, I take the difference of these bins. If my key guess was correct then I would expect a nonzero difference of mean, why now because then because; that means, that if my guess was correct, then the LSB that I had calculated will actually match with the correct value for all the cipher text.

However, if it was wrong it will match roughly half the times, as we have been sufficiently large number of cipher samples have been collected. So, therefore, right in that case when your guess is wrong, then the splitting of the power traces is done kind of in a random fashion. And therefore, you would expect that again the sum or the average

in both the bins will be roughly said and therefore, the difference of mean will be close to 0.

So, that is essentially the basic crux of the idea or our idea of why or why DOM attack works ok. And, you can actually mathematically right there right that in this fashion the same thing. So, essentially the simple idea, but written in a more formal way. So, you can see the attacker now obtains a K sample differential trace delta D. So, this is your differential trace by varying the difference between the average of the traces for which this selection function was 1 and the average for with this selection function was 0.

So, for example, this part of the computation like if you observe this part essentially, essentially you see that it turns out to be that when the selection function was 1 right, if the selection function computes to 1 then; that means, what kind of you are we are basically taking those power traces for with the selection function corresponds to 1. So, therefore, this sigma or this rather this average is the average for the 1 bit ok. On the other end this essentially stands for the 0 bit, because if this value computes to 0, then only this power trace is taken is kind of taken into in your sigma.

So, therefore, this average stands for the 0 bit and therefore, what you are computing here is nothing, but the difference of the bins ok. And the rest of the thing is essentially is exactly as we have seen in the previous discussions.

(Refer Slide Time: 15:09)

**DPA Mathematically**

- Attacker now computes a k-sample differential trace  $\Delta_D[1..k]$  by finding the difference between the average of the traces for which  $D(\dots)$  is one and the average for which  $D(\dots)$  is zero.

$$\Delta_D = \frac{\sum_{i=1}^m D(C_i, b, K_i) T_i[j] - \sum_{i=1}^m (1 - D(C_i, b, K_i)) T_i[j]}{\sum_{i=1}^m D(C_i, b, K_i) - \sum_{i=1}^m (1 - D(C_i, b, K_i))}$$

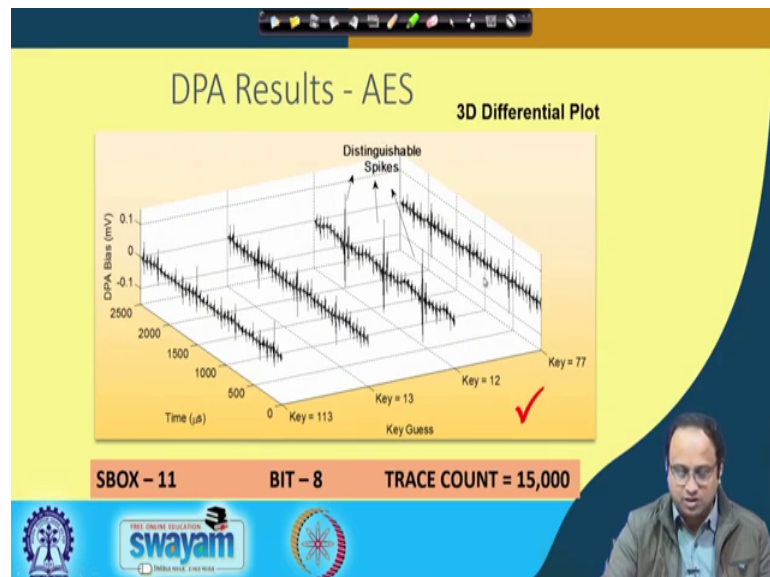
Principle: If  $K_i$  is wrongly guessed,  $D$  behaves like a random guess. Thus for a large number of sample points,  $\Delta D[1..k]$  tends to zero. But if its correct, the differential will be non-zero and show spikes when  $D$  is correlated with the value being processed.

swamyam

So, idea or the principle is that if case is wrongly guessed then DBAS like a random guess does for a large number of sample points this  $\Delta D_1$  to  $K$  will tend to 0. But, if it is correct then the differential will be non-zero and which shows spikes, when the  $D$ , when  $D$  is correlated with the value being processed, that is when your actual S-box which you are targeting for example, is being processed. At that point you will see the differential curve if your guess was correct you will see a non-zero difference of means.

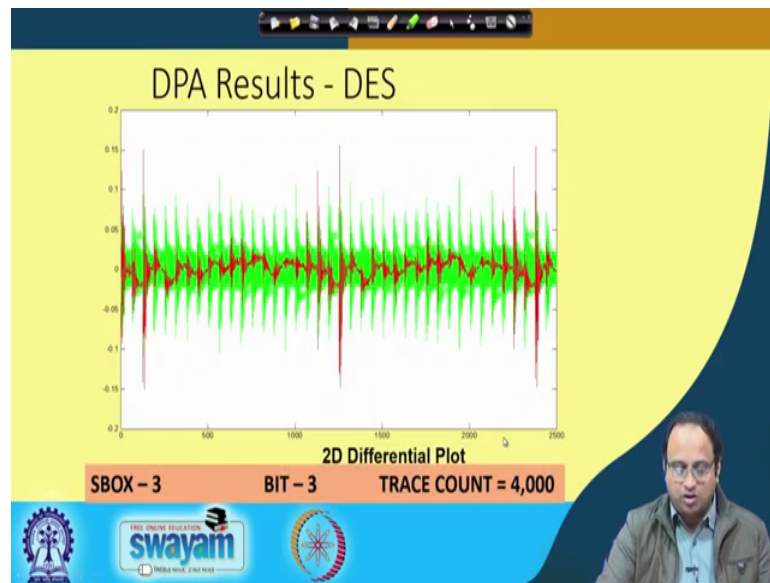
So, it is very interesting in the sense that you can do certain attack and that are more important is that the attack works in real life.

(Refer Slide Time: 15:48)



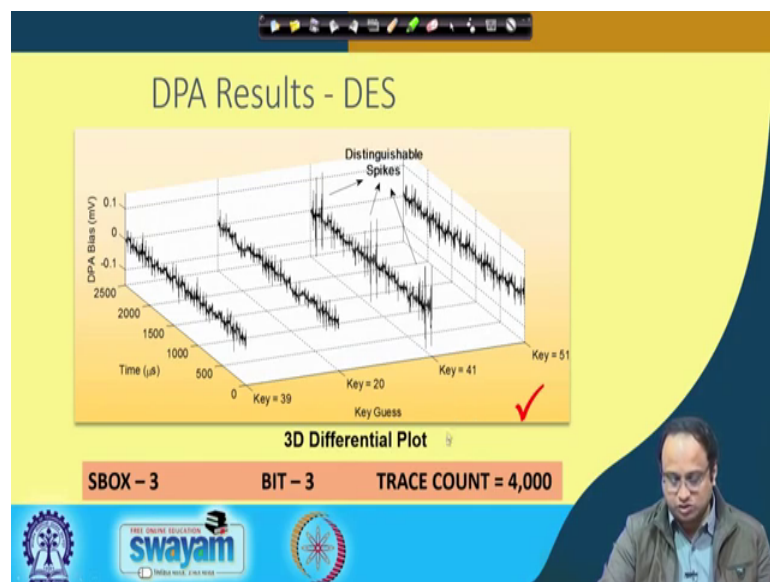
So, here are some examples on actual power traces being accumulated on AES for example, and as you can see that we know that the correct key byte that we are targeting say, you know like we are targeting say the 11th S-box for example, and I say that you can target any of the bits actually. So, in this case we are targeted say for 8 bit for example. So, something with 15,000 power traces ok. You can see that the correct key is showing up differential spikes ok. And therefore, these distinguishable spikes are not since they are not prevalent in the other key bytes, then it would it implies that this is the correct key byte.

(Refer Slide Time: 16:25)



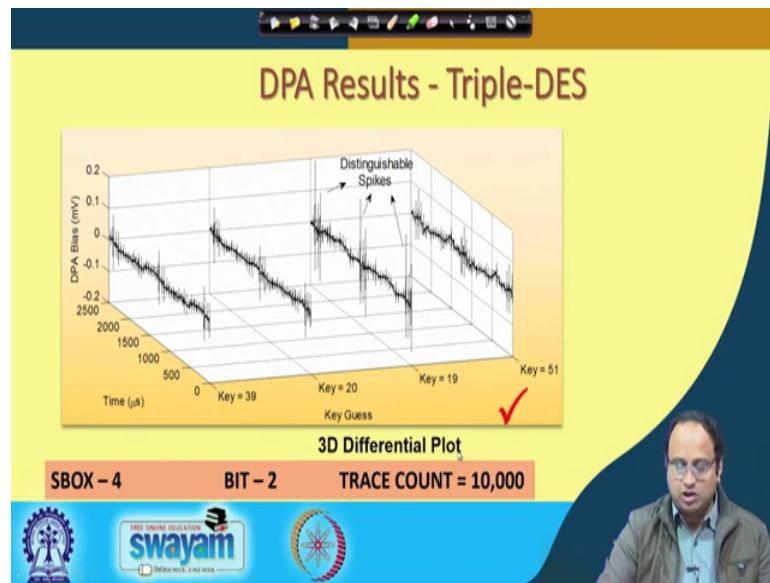
So, likewise you can do the attack on any example you can do the attack on DES also, you can do the attack on so, you know your trace counts will probably vary depending upon an implementation, depending upon your target algorithm.

(Refer Slide Time: 16:30)



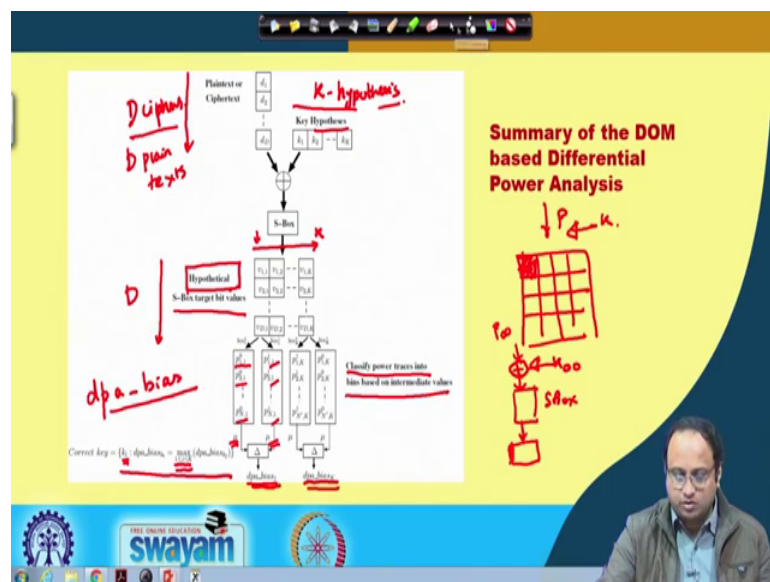
But what is what stays you know like as a central truth is that all these implementations, if they do not take care of these kind of attacks, they are vulnerable to such kind of techniques.

(Refer Slide Time: 16:51)



So, you can also try to apply it on triple DES and even they are you know like you will you can find these distinguishable spikes, which will distinguish the correct key byte from the wrong key bytes.

(Refer Slide Time: 17:03)



So, here is a summary of what we have already studied in a more I would say systematic fashion. So, I am in a more generic fashion, you can if you understand this then you can virtually implement these on any block cipher or any algorithm for that matter.

So, here you see that we have observed what we what we note here is you know like the various plain text or the cipher text. So, you can do the attack in any mode actually. So, imagine that they are all cipher text. So, you have got say  $D$  capital  $D$  cipher texts that you have observed. So, you have got capital  $D$  ciphers and there are  $K$  values for the key byte ok.

So, in our in the case of AES there are 256 values for the key byte. So, all these are your key hypothesis ok. So, there are  $K$  hypothesis which are possible. So, the  $K$  can take I mean the key can take capital  $K$  values. So, now, what do you do is basically you do an exhaust and then you do an S-box. So, you can do an inverse as box when you are operating from the cipher, you can also do the attack from the plain text, if you are doing that from the plain text right. So, this could be either  $D$  ciphers or it could be  $D$  plain text also ok.

The plain text means the message that you are encrypting. So, suppose if we are so, we have already seen when you are doing the attack with respect where you are where you are operating from the cipher. So, you can also do the attack from the plain text ok. So, suppose these are all  $D$  plain text, I kind of exhaust all my key hypothesis. I target any one of the S-box ok.

So, for example, in AES so, imagine that this is your state matrix ok. And, the plain text comes in between right is your plain text and it kind of gets exhaust with your secret key, and you operate your S boxes. So, there are 16 S boxes I can target any one of the S-box suppose I target this S-box.

So, if you target this S-box then; that means, that suppose I guess the  $P_{00}$  which is like 1 byte of the plain text I exhaust it with my secret key say  $K_{00}$  and then I apply my S-box ok. So, this S-box will produce again an 8 bit of output. And these are essentially what are called as the hypothetical S-box target bit values ok.

So, it is interesting to that we have you know like use this one hypothetical which will probably make more sense when we discuss about our next attack method which is called as correlation power attacks, but these are term which we can generate in a use a more generic setting ok. So, that means. So, therefore, the idea is that, you have got a hypothetical. So, the reason why am saying hypothetical because it depends upon the key

hypothesis ok, whether your hypothesis was correct or not depending upon that your bit value will be correctly estimated.

So, here you observed that for all the  $D$  values, we are basically you know like guessed the key and how many guesses do I have I have got  $K$  guesses ok. So, therefore, right I have got  $K$  guesses here which stands for the columns here. And, the rows stands for all the  $D$  observations that I do. So, imagine that I guess make a guess here and then I do it for all the corresponding  $D$  plain text, I evaluate the output of the S-box and I target any one of the bits. So, I target maybe the you know like now let us target the MSP for example. So, then I get so, what I do is that for all these values right.

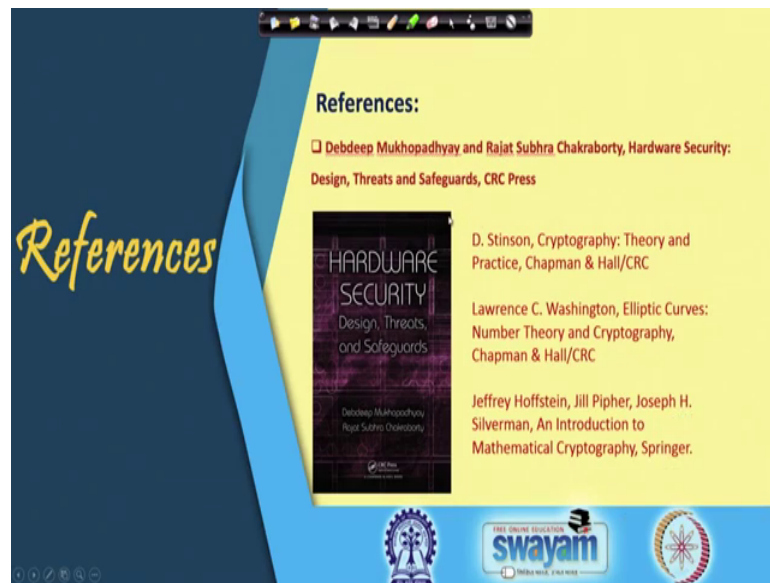
I will so, therefore, so, therefore, this value can be either 0 or 1 ok. And depending upon that I put I classify the power traces into the bins based on this intermediate values. So; that means, by the corresponding power trace  $P_{11}$  and  $P_{21}$  and so on till  $P_{N1}$  kind of indicates those power traces, which has gone into the 0 bin ok.

And likewise these power traces indicate right those were those were those power values which has got into the 1 bin. And, then I take the mean so, I take this mean  $\mu$  I take this mean  $\mu$  here and I take the difference. So, this essentially is also called this difference of mean is also called as dpa bias ok. So, I calculate the dpa bias for all my individual key hypothesis.

So, I have got  $dps$   $dp$  a bias 1 to dpa bias  $K$  ok. And, my correct key computation is calculated by this  $R$  max function. So, what it does is that it takes all the dpa biases compares them and you can see that there are  $K$  such candidates finds out the maximum maximum, and the corresponding key value is returned as my candidate key. So, you can easily observe that since this is the statistical attack with less power traces you will have lot of candidate keys ok. But, as you are kind of progressing and observing with larger number of observations, the correct key hypothesis will stand whereas; the other ones will fall off ok. And therefore, the separation will be more will be better.

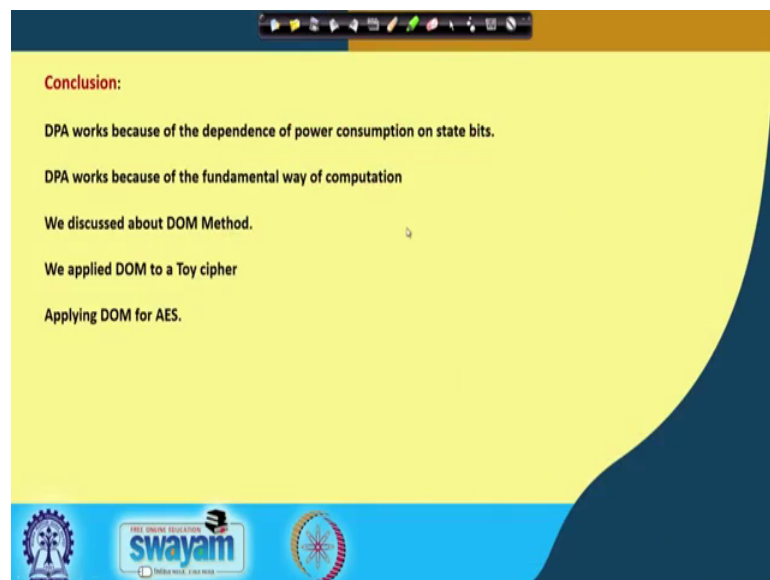
So, that essentially is basically more or less the summary of you know like what essentially we had to discuss about difference of means.

(Refer Slide Time: 22:19)



And, so, let me just you know like so, this is a reference that we have used so, this is the standard textbook.

(Refer Slide Time: 22:25)



So, just to summarize what we have discussed is the DPA works because of the DPA works because of the dependence of power consumption on state bits. DPA works because of the fundamental way of computation as we have seen that it works independent of the algorithm, but it works more because of the way in which we do computations ok.



And therefore, right it is more fundamental and therefore, right if you really want to stop DPA working you need to understand how the counter measures should be built. So, we discussed in particular about the difference of mean which is one of the classical forms of doing a side channel attack. And, we also applied DOM to a toy cipher which is essentially which was implemented using the (Refer Time: 23:06) routine and we also applied to DOM for AES.

So, in particular we apply the DOM for the advanced encryption standard and how it can work both from the cipher text as well as from the plain text. So, in we also kind of tried to understand, how the DOM works on a very generic setting. And therefore, right with this background we should be able to write a simple program to automate the calculation of the difference of mean. So, let me stop here and.

Thank you for your attention.