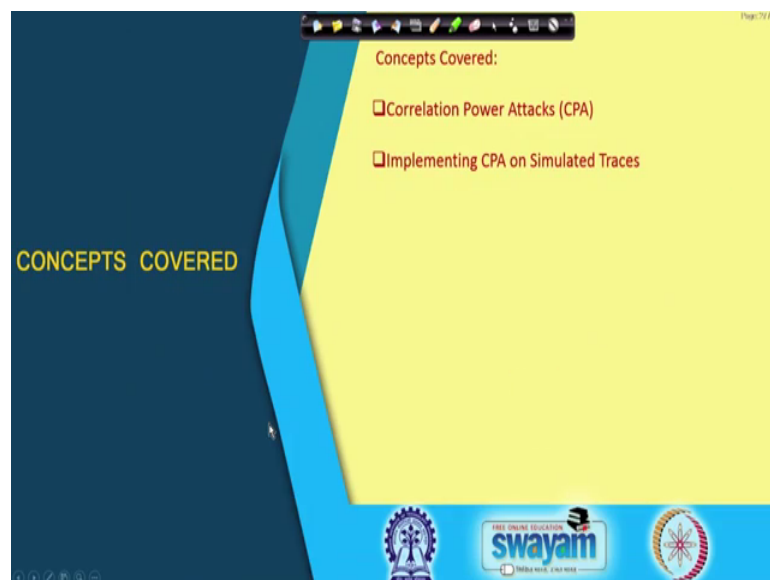


Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 31
Power Analysis - VII

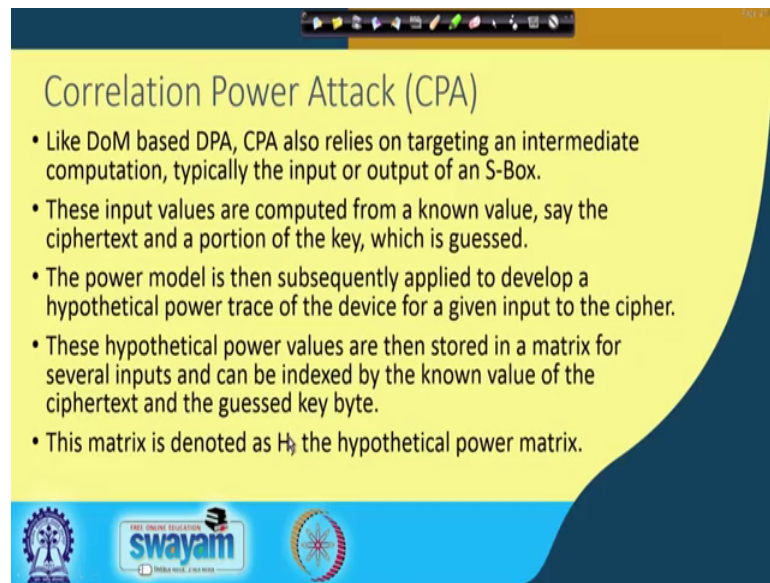
So, welcome to this class on Hardware Security. So, we shall be continuing our discussions on Power attacks. In particular, we shall be talking about how do you know to understand on CPA or Correlation Power Attacks.

(Refer Slide Time: 00:26)



And we will be trying to see how we can implement CPA on simulated traces.

(Refer Slide Time: 00:30)



Correlation Power Attack (CPA)

- Like DoM based DPA, CPA also relies on targeting an intermediate computation, typically the input or output of an S-Box.
- These input values are computed from a known value, say the ciphertext and a portion of the key, which is guessed.
- The power model is then subsequently applied to develop a hypothetical power trace of the device for a given input to the cipher.
- These hypothetical power values are then stored in a matrix for several inputs and can be indexed by the known value of the ciphertext and the guessed key byte.
- This matrix is denoted as H , the hypothetical power matrix.

Logos: Swamyam (Free Online Education), and other institutional logos.

So, as we are discussing in the last class, in correlation power attacks ah, like the DoM right. You basically develop you know you go to the your target byte location but after that right, there is a slight certain change like in the difference of mean right you were basically trying to calculate one of the bits for example. But in a difference of the difference that is or the certain change, that is over here is that you are trying to apply a power model and rather than predicting or getting one of the key bits to kind of split the power traces as we are doing in DoM.

What we try is we try to develop a hypothetical power trace of the device for a given input to the cipher. So, these hypothetical power values are then stored in a matrix for several plain texts and can be indexed by the known value of the ciphertext and the guessed key byte; that means, you can kind of index the hypothetical power traces based upon your ciphertext power portion and also the portion of the guessed key byte. And this matrix is formally defined as H or the hypothetical power matrix ok.

(Refer Slide Time: 01:35)

The slide is titled "Correlation Power Attack (CPA)-Contd." and contains the following bullet points:

- The attacker also observes the actual power traces, and stores them in a matrix for several inputs.
- The actual power values can be indexed by the known value of the ciphertext and the time instance when the power value was observed.
- This matrix is denoted by T , the real power matrix.
- It may be observed that one of the columns of the matrix H corresponds to the correct key k^* .
- CPA tries to compute the **similarity** between the columns of the matrix H and the columns of the matrix T , to distinguish k^* from rest: similarity computed by Pearson's Correlation, usually.

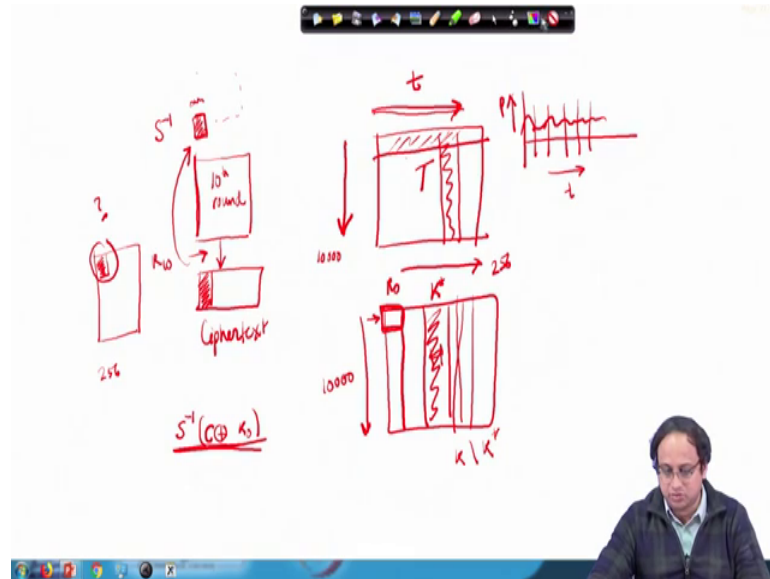
The slide also features a video feed of a presenter in the bottom right corner and logos for Swamyam and other educational institutions at the bottom.

And you also have another similar matrix which is called as the real power matrix which is denoted as T . Here the attacker also observes the actual power traces and stores them in a matrix for several inputs and the actual power values can now be indexed in this case by the known value of the ciphertext and the time instance when the power value was observed ok.

And this matrix is denoted by capital T which is the real power matrix. So, it may be observed that one of the columns of the matrix H corresponds to the correct key which is k^* ok. That is one of the columns of your matrix H ; that means, your hypothetical power matrix is where your correct key is essentially coming into play and that is essentially denoted as k^* .

So, what you try to do is, you try to in CPA you try to compute the similarity between the columns of the matrix H and the columns of the matrix T to distinguish k^* from the rest. Similarly, computed in this case is similarity is contributed computed by using Pearson's correlation usually. So, what I mean to say let me elaborate here.

(Refer Slide Time: 02:47)



So, basically what I am trying to say that is you have you know like the say the last round of AES. So, considered that this is your 10th round of AES where you are essentially calculating the cipher. So, this is your corresponding cipher ciphertext and you are targeting say one specific byte of the cipher ok. So, what you are do here is that, you are basically trying to get back but if you want to get back, then you have to guess this key right because there is this 10th round key which comes in between. Again, right what we have seen in DoM right you do not guess the entire key, but you guess only a portion of the key.

So, I guess this portion of the key. And then, you apply if you want, so, there are these bundle of S boxes. So, there are there are 15 S boxes over here. But if you want to go to the input, then you basically need to invert say one of these S boxes. So, this is your say S inverse which means the inverse S box. So, the idea is that when you are doing when this when AES hardware is you know like working at that point of time, you try to you basically kind of take the power traces and you store the power traces in the in a in the form of a matrix ok. So, this matrix is what is called as a real power matrix or T.

So, in T you are basically kind of indexing T by the number of samples like suppose you are encrypting say 1, 2, so on, may be 10000 plain text for example, So, this is essentially you know like say 0 to whatever 1 to 10000. So, these are all the number of encryptions which you are doing and the row essentially is the time instance. So, this is

that you know like your T for example. So, therefore, right any row that you basically have here you basically are some way you know like having your AES power trace and your sampling that at different time instances ok. So, this is your T and this is your power consumption and all these real values are being stored in this table ok.

So, therefore, this table is basically a you know like a representation of your actual power consumption ok. So, now, you are trying to develop the hypothetical power model. Now, for the hypothetical matrix which is essentially denoted as the matrix H, so over again you know I have got this 10000 encryptions ok. But for all of them, you are basically making a guess says.

You know that there are when you are guessing this key byte, you do not know what is this key byte. There are 256 possible values of this key right. There are 256 possible guesses which are possible of this key. For every guess, suppose you know like let me write these guess as k_0 ok, say I guess this k as say 0 as one of the key.

So, then I you know you know like I take each of these ciphertext, suppose I take the first ciphertext ok. So, let me call it as you know like C_i , XORed it with k_0 and I calculate inverse of it ok. I calculate say S inverse and then I basically find out the hamming weight say of this value ok. And that so, the I basically I can apply any power model for that matter and whatever is my favorite power model based upon that I get an integer value here and I store that values over here ok. So, like this right, I basically do it for all the possible ciphers ok. So, therefore, right here the this is indexed by say 256 because there are 256 possible values of the key ok.

So, now the question is what CPA tries to do is the CPA try CPA basically knows that you know like one of these keys is correct right. It knows that say suppose k^* is correct and therefore, the column corresponding to k^* right should correlate somewhere with these columns because there is a real power trace being observed here and this is your hypothetical power. At some point of time, there should be a correlation; that means, you know like this should correlate to either this time instance or this time instance you know for example, in the simulated power trace. There are 12 time instances at some point right there should be a correlation.

Whereas for the wrong key; that means, if I consider any other key right like k which is not k^* ok; that means, for any other wrong key, I will never get that correlation ok.

And therefore, I can distinguish the correct key from the wrong key. So, this is the broad philosophy of correlation power attacks. And now, let us see you know like how we can understand it better ok. So, the so, this similarity is essentially very vital and the you can use any other similarity matrix, but one of the most popular once is what is called as Pearson's correlation matrix.

(Refer Slide Time: 07:40)

Computing Correlation Coefficient for Simulated Power Traces for AES

- Like before, we simulate the power profile for the iterative AES, this time by using Hamming Distance power model applied on the state registers updated after each round.
- The real power matrix is stored in the array `trace[NSample][NPoint]`
 - NSample: Number of Power Traces acquired
 - NPoint: Time instances for which the power values are observed. Here NPoint is 12.

So, basically again we try to compute correlation coefficients for the simulated power traces for AES like before we simulated the power profile for an iterative AES that if AES this time by using say hamming distance because last time we were using the hamming weight.

So, let us try hamming distance again you know. You can again try hamming weight as well. So, we if we try hamming distance power model and we apply it on the state registers updated after every round and then we store it in again this matrix which is called as trace index by N sample and N point. So, if you remember right the real matrix that I showed, you had you know like 10000 for example, on one side that is your N samples number of encryptions you are doing and on the other hand N point basically it is in it on the, number of time instances the point where you are sampling you are power pro consumption ok.

So, in this case, in the similarity environment N point is 12 ok. So, N sample is the number of power traces acquired and N point is the time instances for which the power values are observed; here, N point is 12 ok.

(Refer Slide Time: 08:39)

The slide is titled "Calculating the Hypothetical Power". It features a grid of registers labeled R0 through R15. The registers are arranged in two rows of eight columns each. The first row contains R0 to R7, and the second row contains R8 to R15. Each register contains a pair of values, such as S0,C0 for R0, S1,C1 for R1, and so on. Red circles and arrows highlight specific registers and their components. Handwritten red text on the right side of the slide shows the calculation: $S(S_0) \oplus K_0 = C_0 \Rightarrow S_0 = S^{-1}(C_0 \oplus K_0)$ and $H(D(S,C))$, $S(S_0) \oplus K_5 = C_5 \Rightarrow S_0 = S^{-1}(C_5 \oplus K_5)$. The slide also includes a small diagram of a Hamming distance matrix and a logo for Swayam at the bottom.

So, next thing that we will see is how do we calculate the hypothetical power ok. It is quite straight forward but little tricky because of the AES shift rows ok. So, let us try to observe this how it works. So, for example, when you are finding out the hamming you know like say you are applying the hamming distance power model. So, then it matters like, you need to observe that this is your corresponding register where you are storing the input of the tenth round and also the output of the tenth round.

So, therefore, right any register, if you want to calculate the hypothetical power because of a register, then you have to basically find out what is the content before and after and take the hamming distance between them ok. So, for example, let us consider R 0. So, if you are considering R 0, then you are basically finding out the distance between C 0 and S 0 ok. And therefore, right here you know that, you know like C 0 is given you know that is basically stored here in this table. And C 0 right, if you want to C 0 and S 0 and you see that S 0, you can get from this equation because S of S 0 XORed with K 0 is equal to C 0.

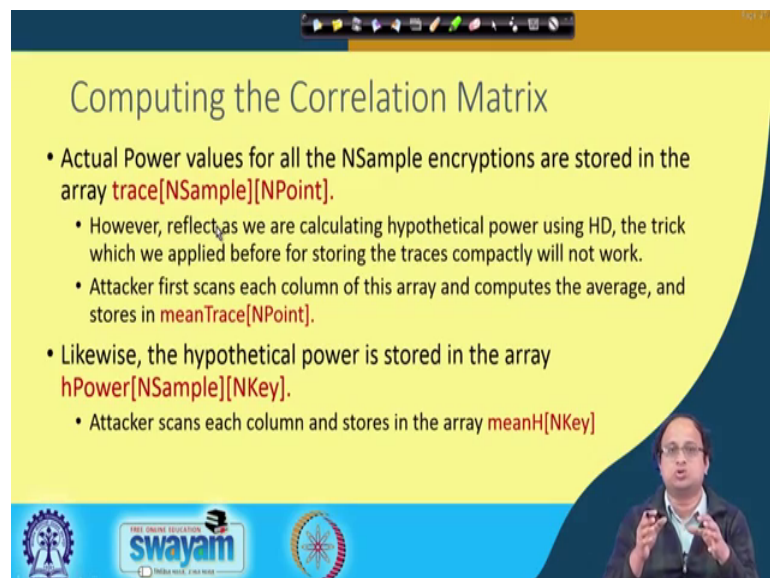
So, therefore, you want to predict S 0 right. You have to basically XORed C 0 with K 0 and find out the inverse of it ok. So, this is quite straight forward. On the other hand, if

your targeting say R 1, suppose you are target. So, this fine for R 0 right, so if you are now targeting R 1 for example. So, then that means, that you are basically targeting this right and R they that now you can observe that you have to observe or find out the hamming distance between S 1 and C 1 ok. So, you have to find out the hamming distance between S 1 and C 1 ok. But you can observe that the content of S 1 can be got from this byte can be got from C 5 ok.

Because if you want to get S 1, then because of shift row, you have the equation which is S of S 1 XORed with K 5 is equal to C 5 and that implies that S 1 is equal to S inverse of C 5 XORed with K 5 ok. And that is basically written over here that you have to target C 5 and K 5 ok. So, likewise you can work out the remaining things but it is important that when implementing, you have to take care of the shift rows.

Otherwise, you will get wrong results. When you are in particularly, when you are applying hamming distance power models, when you are applying hamming weight power models it is probably more straight forward it. You need not bother about this. But when you are trying with the hamming distance power model, you need to take care of the shift rows that that is easy.

(Refer Slide Time: 11:34)



The slide is titled "Computing the Correlation Matrix" and contains the following bullet points:

- Actual Power values for all the NSample encryptions are stored in the array `trace[NSample][NPoint]`.
 - However, reflect as we are calculating hypothetical power using HD, the trick which we applied before for storing the traces compactly will not work.
 - Attacker first scans each column of this array and computes the average, and stores in `meanTrace[NPoint]`.
- Likewise, the hypothetical power is stored in the array `hPower[NSample][NKey]`.
 - Attacker scans each column and stores in the array `meanH[NKey]`

The slide also features a presenter in the bottom right corner and logos for Swamyam and other institutions at the bottom.

So, now so once we have done that, we basically you know like we can go to the next step which is computing the correlation matrix. So, here we have got the actual power values for all the N sample encryptions which are stored in the matrix trace N sample N

point as we have seen. However, you should reflect on the point that we are calculating the hypothetical power using hamming distance and therefore, the trick that we saw in previously right when we were talking about the hamming weight power model will now not work ok.

You cannot just store a based upon the only a byte ok. And that is why, you probably need a bigger data structure. So, let us use again all the N sample points ok. And the other point is like that that attacker right first scans each column of this array and computes average of each of this columns. So, basically what now the attacker does is that attacker wants to calculate the correlation between each of these columns like basically wants to calculate the correlation between a column of the real matrix t with your hypothetical power matrix ok.

So, basically for every column if you want to calculate the correlation therefore, you need their individual means ok. And their individual means are here denoted as say mean trace N point note that. Why N point, because every column right will have one mean and there are N point such columns in the real matrix. So, therefore, mean trace will have N points ok. Likewise for the hypothetical power matrix, you know that it is indexed in the column by N key that is the number of possible key bytes which is set 256. And therefore, the attacker can scan again each column and therefore, the average for that column is stored in the data structure or array which is mean H and the number of possibilities is N key.

So, you have got mean H denoting the average for the columns of the hypothetical matrix and mean trace storing the average for your real power matrix which is t ok.

(Refer Slide Time: 13:24)

Correlation Matrix

NSample

Hypothetical
Power matrix
H

NKey

NSample

Real Power
matrix T

NPoint

NPoint

Correlation
matrix C

NKey

$$C[i][j] = \frac{\sum_{k=0}^{NSample} (hPower[i][k] - meanH[i])(trace[j][k] - meanTrace[j])}{\sum_{k=0}^{NSample} (hPower[i][k] - meanH[i])^2 \sum_{k=0}^{NSample} (trace[j][k] - meanTrace[j])^2}$$

So, then we are all set to calculate the correlation matrix. So, here is basically what we are doing. You are you have got the hypothetical power matrix, you have got the real power matrix ok. So, let this denote this as your you know like a one of the columns and the of your hypothetical power matrix and you are trying to correlate this with another column of the real power matrix ok. And this you are storing in this correlation matrix, so you are basically taking this and indexing here as the row and you are taking this column position and indexing that as the column of this correlation matrix.

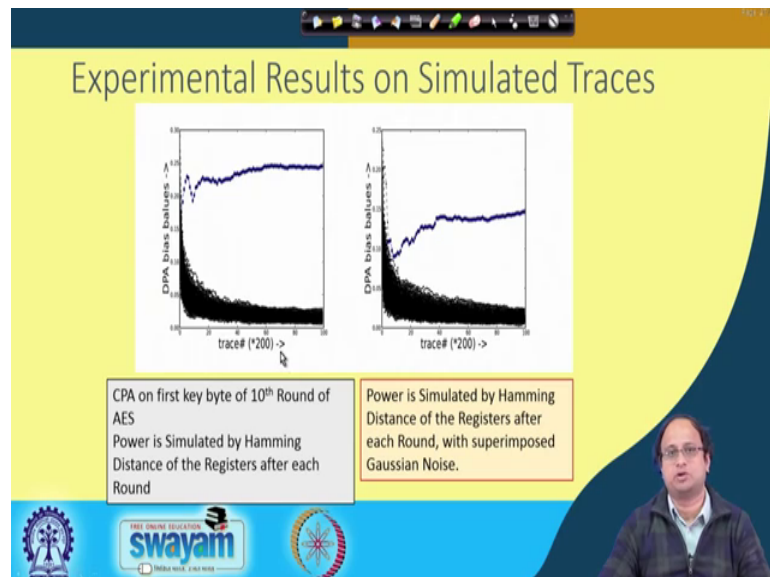
And the value here is essentially nothing but this. So, basically right it stores. So, $C[i][j]$ will mean that you are taking the i th column of the hypothetical power matrix and your correlating with a j th column of your real power matrix. And as you know, that if you want to correlate this vector with this value, you have to just you can just it has apply Pearson's correlation which means you have to basically find out the covariance and divided by the product of the variances. So, you have to calculate the variance of each of them and then you have to divide by their product ok.

So, that is exactly done here. The numerator stores the covariance and the denominator stores the product of the variances. So, you know like for if you are very accurate may be you have to do a square root and sum, but more or less you get the same value. So, you know you get the same proportionality. So, you need not bother about you know like applying the square root here. So, therefore, I, so, therefore, now you basically observe

this correlation matrix and ideas is that you have got N keys. So, remember that the rows are indexed here by N keys and the idea is that what you expect basically is that for the correct key that is suppose your correct key is k^* , you will find out one non trivial value here in the correlation.

Whereas if you are key is wrong, then probably what you will see is that all of these values will be less much lesser ok. So, therefore, you basically try to identify the correct key, not the correct key row where you get a non trivial value of the correlation of correlation and that basically distinguishes you are correct key from the wrong key.

(Refer Slide Time: 15:35)



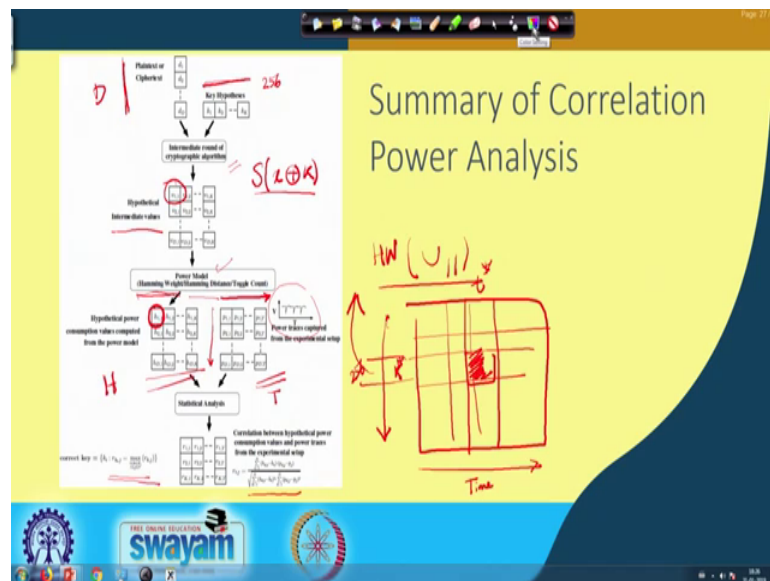
So, here is an experimental result which has been developed on simulated power traces. So, what we have done is basically again done our you know like simulation. So, you have basically you know like simulated the power traces by hamming distances here of the registers after every round.

So, you basically calculating the hamming distance with a of a particular round with the previous round. And then, you are basically trying to find out the corresponding you applying CPA to develop you know like to distinguish the correct key from the wrong keys. This is again is like a similar analysis, but here we have basically added noise and therefore, although there is a separation, but the separation comes little later little later compared to the in this case where there is no noise. So, of course, like when we will see

later on when you are trying to do this analysis with actual power traces, then you have got more phenomenon which you need to kind of you know like accommodate.

But at the same time, the technique remains the same and if you understand this. And if you are able to apply it on simulated power traces, then we have a fair good of good amount of understanding about how the attack works ok.

(Refer Slide Time: 16:44)



So, therefore, let me summarize this discussion on correlation power analysis. So, basically right here we have got. So, you know like say D number of; D number of plains. So, let me you know like. So, again you can also visualize this attack say from the perspective of the plaintext, you can do it either from the plain text or the cipher texts.

So, we have got say D number of plain texts or cipher texts that you have taken, these are your corresponding key hypothesis; that means, again in our context of AES. There are 256 possible hypothesis, you calculate the intermediate round. So, this is the intermediate round of your cryptographic algorithm. So, basically when you are doing it from the plaintext you will not calculate the inverse S box, but you will apply the forward operations.

So, you have basically take the plain text portion XORed it with your guess and calculate the S box ok. And you will find out say one of you basically you will get the state and that state essentially is denoted here as the hypothetical intermediate values.

Hypothetical why, the because you are guessing the key that is why it is hypothetical you do not know whether that is actual state or not. But then, what you do is that you basically in order to understand whether its a correct one or whether it is wrong. You basically apply your power model on the hypothetical state ok. So, when you applying your power model, suppose you are applying the hamming weight or the hamming distance or any other thing for example, whichever is your favorite power model.

So, when you are applying say the hamming weight, so now, you basically takes a V_{11} which is your hypothetical intermediate state and then you basically apply you should take V_{11} and say you will like calculate the hamming weight. So, this hamming weight essentially gives you the hypothetical power value ok.

So, therefore, you get H_{11} and you would do it similarly for the entire operations ok. But how many such matrices will you get ok? You should you have basically you know like getting it right I mean you are getting an hypothetical power value, but observed that the hypothetical power value right which you are; which you are observing is for a specific byte ok.

But then, if you change the input; that means, if you change the you know if you do it for all the D possible values of the plain text, you will get D possible values of the hypothetical power trace and you will get it for every key guess that you do and that essentially is nothing but your matrix H .

So, this is your hypothetical power matrix H ok. And now you basically observe the actual power consumption. As we discuss the actual power consumption is denoted by the matrix T ok. So, note that T is indexed again in the row by the number of encryptions you are doing that is D encryptions which you are doing and then the column is the time instance. So, here is the power trace for example, you can see it is the voltage versus time plot and this is the time which you are observing ok.

And therefore, now we have got two matrices and now you basically try to kind of you try to correlate the you know like one of the columns of the hypothetical power matrix with your real power matrix and that gives you the i comma j position of your correlation matrix ok.

And that essentially as we have discussed is noted over here and then the correct key is returned by finding out the max of these correlation value. So, basically what you do is, you try to find out the maximum value in the correlation matrix the corresponding row gives you or the corresponding index of the row gives you the correct; gives you the correct candidate key right.

Because what you are expecting is that in your correlation matrix, you have got you know like. So, the correlation matrix will be indexed by you know like you. So, you, so you have got you know like. So, it is indexed here by the number of possible key for example, in AES there are 256 possible keys and it is indexed in this direction by time right.

The idea is that out of all these 256 keys, there is one key which is the correct k^* ok. And note that k^* the operation because of k^* will not correlated all time instances, but rather it will correlate at a specific time instance which is say T^* . And that is my point of interest ok. And the idea is that this value or the assumption is that this value will be the maximum value ok.

So, therefore, I basically try to see this matrix find out the maximum value and then just written the corresponding index as my correct candidate key and that essentially is expected to be the correct key ok. So, that is essentially pretty much what we have to discuss here and.

(Refer Slide Time: 21: 26)



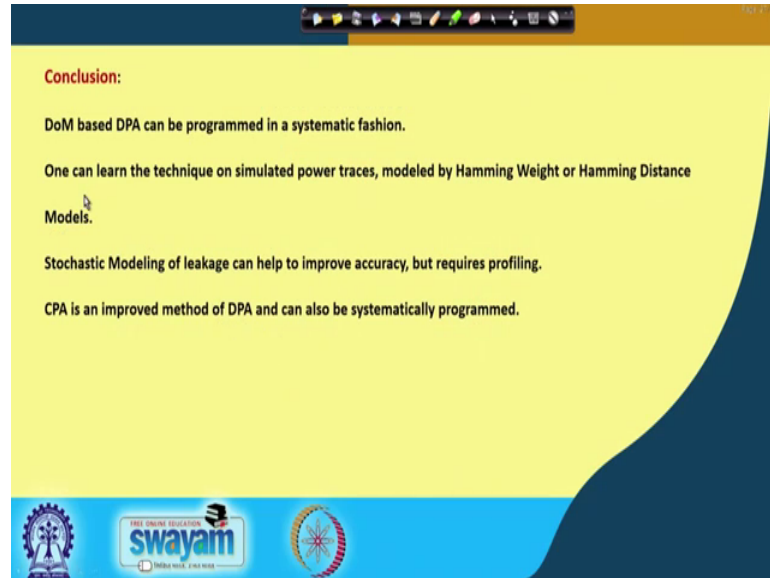
The slide is titled "References" in a stylized yellow font on a dark blue background. The main content area is yellow and contains a list of references. On the left side of the yellow area, there is a small image of the book cover for "Hardware Security: Design, Threats, and Safeguards" by Debdeep Mukhopadhyay and Rajat Subhra Chakraborty. The bottom of the slide features logos for IIT Madras, Swayam, and another organization.

References:

- Debdeep Mukhopadhyay and Rajat Subhra Chakraborty, *Hardware Security: Design, Threats and Safeguards*, CRC Press
- D. Stinson, *Cryptography: Theory and Practice*, Chapman & Hall/CRC
- Lawrence C. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC
- Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, *An Introduction to Mathematical Cryptography*, Springer.

So, so, basically right I mean let us. So, this is the reference that we have been following for our discussions.

(Refer Slide Time: 21:30)



And should now to conclude this, so what we discussed is are the difference of mean based DPA can be programmed by a systematic fashion we have seen how to write programs for DoM, that is the difference of mean technique. One can learn that the technique on simulated power traces modeled by you know like hamming weight or hamming distance models without even getting access to complicated setups ok, you can learn and even develop these techniques you can try to develop better statistical techniques without even going to you know like the going to a costly setup ok.

Likewise, right we have also seen how to develop that you will able to seen that it is important to you know like to also learn the leakage models and that you can do by you know like modeling the leakage and that essentially brings us to what is called as a stochastic models of leakage which can help us to improve accuracy, but almighty requires a profiling stage and.

So, the difference of mean and the CPA that we discussed are both non profiling attacks. I supposed to this we are something which are called as template attacks which are examples of profiled power attacks ok. And CPA is an improved method of DPA and can also be systematically programmed. You can write again similar programs and therefore, you can compare the performance of CPA with difference of mean techniques and you

can try to understand how they you are like trade off with each other in terms of efficiency, you know as well as success rate.

So, we will see the next class about how to you know define matrix and understand how to obtain the success rates of our side channel experiments.

So thank you for your attention.