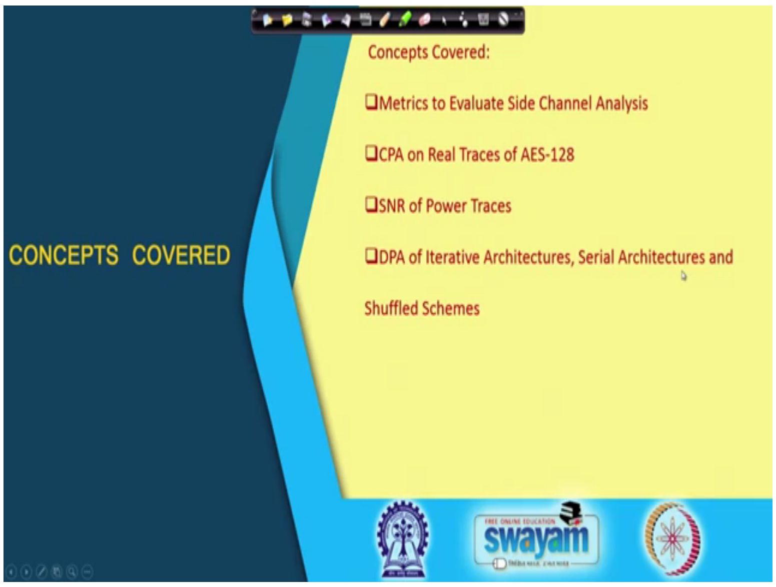
Hardware Security

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

Lecture – 32
Power Analysis - VIII

So, welcome to this class on Hardware Security. So, we shall be continuing our discussions on power attacks.

(Refer Slide Time: 00:24)



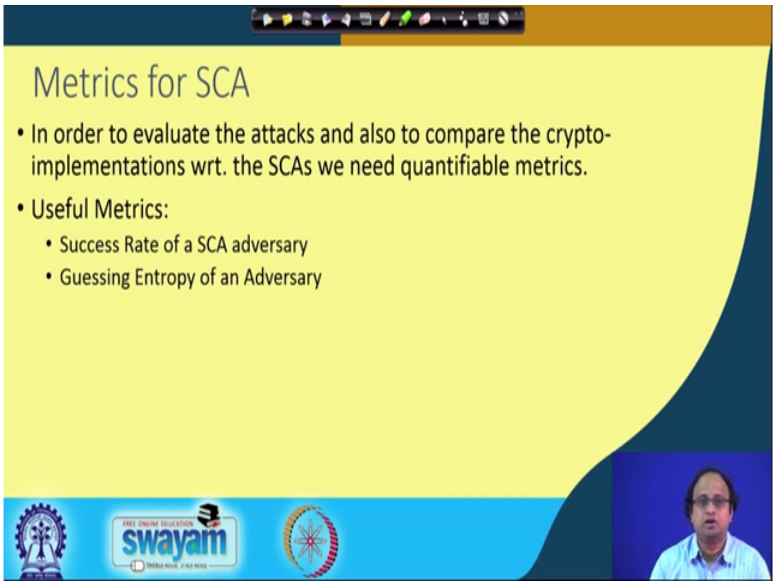
CONCEPTS COVERED

Concepts Covered:

- ❑Metrics to Evaluate Side Channel Analysis
- ❑CPA on Real Traces of AES-128
- ❑SNR of Power Traces
- ❑DPA of Iterative Architectures, Serial Architectures and Shuffled Schemes

In particular in today's talk, we shall be try to look into various metrics which are often useful for evaluating and giving a quantitative evaluation of side channel attacks. We shall be looking to CPA on real traces of AES 128 like in the last classes, we discussed about doing power attacks on simulated power traces. So, we shall be looking into some of the examples on real trace attacks. We shall be defining signal to noise ratio or SNR of power traces and we shall be using it as a guideline to we compare DPA on iterative architectures compared with C L architectures and a special scheme which is called as shuffled schemes.

(Refer Slide Time: 00:59)

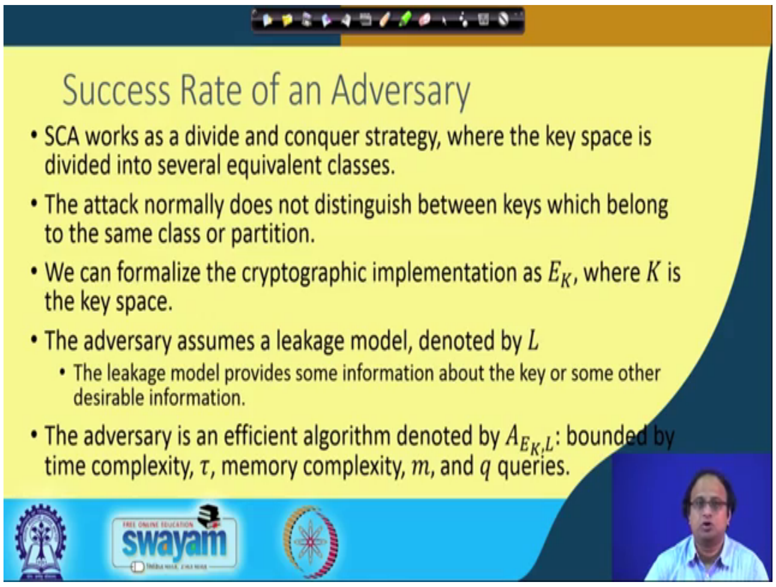


Metrics for SCA

- In order to evaluate the attacks and also to compare the crypto-implementations wrt. the SCAs we need quantifiable metrics.
- Useful Metrics:
  - Success Rate of a SCA adversary
  - Guessing Entropy of an Adversary

So, to begin with, there are different metrics for side channel attacks and metrics are very useful because they provide us a quantitative framework to compare between the success rate of two side channel attacks. It can also be useful for kind of understanding whether a countdown measured is strong enough to protect against a side channel attack. So, two of the most; two of the most useful metrics are success rate of a side channel attack adversary and the second one is called as guessing entropy of an adversary. So, we shall be trying to define these two metrics in today's class.

(Refer Slide Time: 01:36)



Success Rate of an Adversary

- SCA works as a divide and conquer strategy, where the key space is divided into several equivalent classes.
- The attack normally does not distinguish between keys which belong to the same class or partition.
- We can formalize the cryptographic implementation as E_K, where K is the key space.
- The adversary assumes a leakage model, denoted by L
  - The leakage model provides some information about the key or some other desirable information.
- The adversary is an efficient algorithm denoted by $A_{E_K,L}$: bounded by time complexity, τ, memory complexity, m, and q queries.

So, to begin with, what is a success rate of an adversary? So, side channel attack as we have seen works by a divide and conquer strategy where the key space is divided into several equivalent classes like what we normally try to do in this attack is, we try to understand whether a key belongs to a specific class. So, this class could be for example, a hamming weight class or a hamming distance class. So, we try to take a key we for example, have a big key space and we try to map that key from which is a member of that key space into a partition or into an equivalence class.

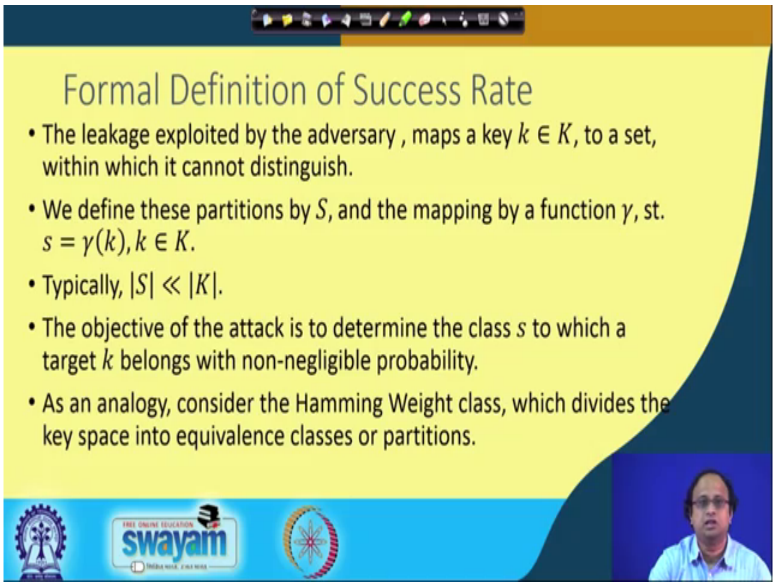
For example, we say for we may we might say depending upon hamming weight, if you are using hamming weight as a power model or a leakage model, then we take a key and we try to map that into a given hamming class ok. So, suppose you are using 8 bits then the hamming class could be 0, 1, 2, 3 4 so, until 8. So, we know that depending upon the partition for example, like if the hamming class is like 1 right then there could be a if I am considering the byte in the state matrix, then there could be 8 possible values.

Like the key could be 1 0 0 0 0 and so on or 0 1 so on and then again followed by 0s and or it could be like this till like 0 fall like contiguous 0s followed by 1 ok. So, therefore, right the attack normally does not distinguish between keys which belong to the same class or partition, we can formalize if it. So, suppose I want to formalize this notion, I want to formalize this notion of an adversary, then let us use this variable E K where K is the key space and E K essentially stands or denotes the cryptographic implementation.

So, it is for example, it is an AES algorithm which has been implemented. Now the adversary assumes a leakage model and let the leakage model be denoted as L. The leakage model provide some information about the key or some other desirable information. For example, it could be as I said the hamming weight of the key. Now, the adversary is often modeled as an efficient algorithm. So, it is often modeled as an efficient polynomial time algorithm which is denoted by A E K L and it is bounded by its time complexity tau memory complexity m and q queries.

Which means like it requires a time, I mean the adversary in order to essentially return an equivalent of the key or the equivalence partition for the key it requires time which is proportional to tau or denoted by tau and the memory complexity is bounded by m and it can make q queries. So, it is important to keep in mind that the adversary needs to be an efficient algorithm. So, we are not considering against unbounded adversaries in this case; we are considering computationally bounded adversaries.

(Refer Slide Time: 04:41)



Formal Definition of Success Rate

- The leakage exploited by the adversary , maps a key $k \in K$, to a set, within which it cannot distinguish.
- We define these partitions by S, and the mapping by a function γ, st. $s = \gamma(k), k \in K$.
- Typically, $|S| \ll |K|$.
- The objective of the attack is to determine the class s to which a target k belongs with non-negligible probability.
- As an analogy, consider the Hamming Weight class, which divides the key space into equivalence classes or partitions.

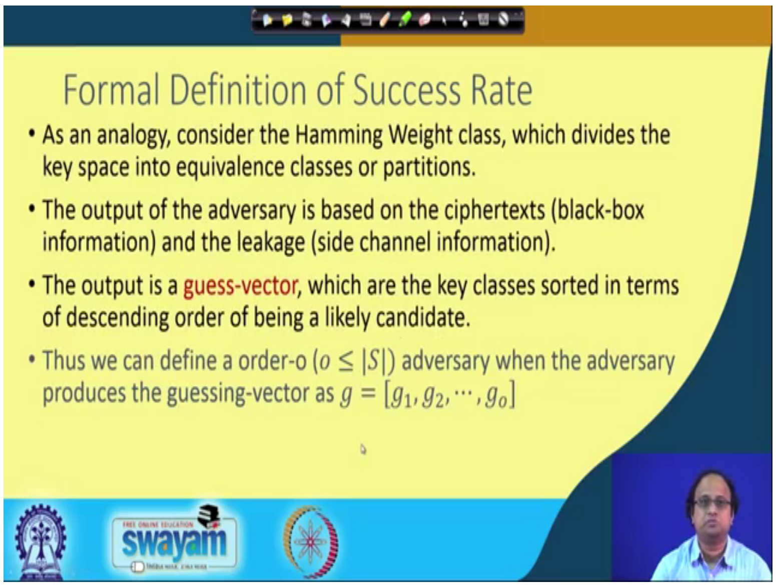
So, if you take such an adversary and imagine that the leakage, so, the adversary in this case as we know is the side channel adversary. So, it not only sees the input and the output of an algorithm, but it also tries to exploit the leakage of an encryption algorithm or encryption implementation rather. And what it does is that, it maps a key which belongs to the big key space K to a set within which it cannot distinguish. So, for example, it can say that it belongs to hamming class say 4, but within 4 there could be several possible keys whose hamming weights are 4 and it may not be able to distinguish between them or among them.

So, we define these partitions by a notation S and the mapping from the space key or from the space K to the space S is denoted by a function, then function is called gamma. So, we say that let me from this key space choose K and then apply this function gamma and then I get a small s. So, small s is a essentially nothing but the mapping. So, typically of course, right as you can see that if I am considering say 128 bit space there are 2 to the power of 128 possibilities and if I map them into the hamming classes, there are 129 possible hamming classes.

So, 0, 1, 2, so until 128, so typically as we can see that the number of partitions or the number of hamming classes for example, is much smaller compared to the total number of keys. Like in the keys, we have got 2 to the power of 128 which is very large which we can which we are basically narrowing down to only 129 possibilities. So, therefore, the cardinality of S is typically much smaller than the cardinality of K and that is why our attack will be successful in that case and objective of the attack is to determine the classes to which your target K belongs with non-negligible probability, that is our objective.

So, as an analogy, I said we can consider the hamming weight class which divides the key space into equivalence classes or partition. So, hamming weight or hamming distance could be used as an analogy to understand this experiment.

(Refer Slide Time: 06:43)



Formal Definition of Success Rate

- As an analogy, consider the Hamming Weight class, which divides the key space into equivalence classes or partitions.
- The output of the adversary is based on the ciphertexts (black-box information) and the leakage (side channel information).
- The output is a guess-vector, which are the key classes sorted in terms of descending order of being a likely candidate.
- Thus we can define a order-o ($o \leq |S|$) adversary when the adversary produces the guessing-vector as $g = [g_1, g_2, \cdots, g_o]$

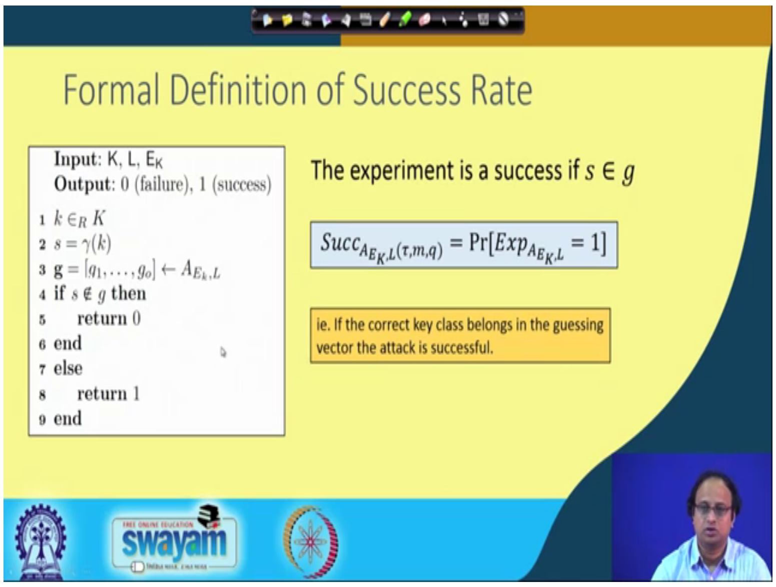
So, let us see what do we do in this experiment. So, what we do here is suppose you know like if you are; if you are considering; if you are considering right here an adversary, then the adversary based upon the ciphertext because, the ciphertext is already there with the adversary because is a black box information and the adversary has an additional side channel leakage information.

So, now, together with this the adversary comes up with a guess vector say, we denote it by g which is nothing but the key classes which are sorted in terms of descending order of being likely a candidate. For example, like it may happen that the guessing vector, if I consider the analogy of hamming weight class will tell me that essentially the key belongs to a guessing vector which is denoted as say 1, 2, 3; that means, like it could be either hamming weight class 1 or it could be hamming weight 2 or it could be hamming weight 3. But the attacker is saying that probably 1 is most likely ok. So, that is exactly what is meant by this guessing vector.

So, therefore, the guessing vector typically is again we define this an order o adversary if the guessing vector will come up with o such candidate classes. So, it will come up with vector g 1, g 2, so until g o. And of course, you can understand that o will be lesser than the number of part possible partitions. It will lesser than in you know and if I talking about a 128 bit state, then and if I am consider about considering 128 as the maximum hamming weight possible, then there are 129 hamming classes.

Then, o would be lesser than 129, because 129 is essentially the all possible cases. So, now, the adversary returns is guessing vector g ok. And the attack is said to be successful if the key class; that means, suppose you know like if I take the correct key K and if I map it to S belongs to this guessing vector, if it belongs to this guessing vector, then the attack is successful else the attack is a failure.

(Refer Slide Time: 08:44)



Formal Definition of Success Rate

Input: K, L, E_K
Output: 0 (failure), 1 (success)

```
1 $k \in_R K$
2 $s = \gamma(k)$
3 $g = [g_1, \ldots, g_o] \leftarrow A_{E_k, L}$
4 if $s \notin g$ then
5     return 0
6 end
7 else
8     return 1
9 end
```

The experiment is a success if $s \in g$

$$Succ_{A_{E_K,L}(\tau,m,q)} = \Pr[Exp_{A_{E_K,L}} = 1]$$

ie. If the correct key class belongs in the guessing vector the attack is successful.

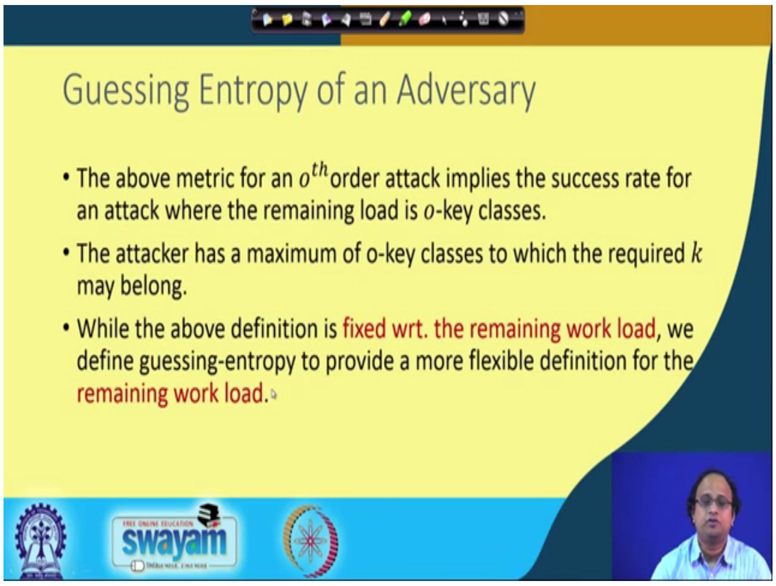
So, exactly there is modeled by this algorithm. It says that I take an input which is K L and E K. E K stands again for the encryption implementation, L stands for leakage, K stands for the possible key space and the output is 0 or 1.

Which means either the experiment is successful or the experiment is a failure. So, if you take any member of this key space say denote it by small k and if I map this k to s; that means, s is my correct key class and the attack or the adversary who is bounded by you know like tau m and q as I said that it is a polynomial timely bounded adversary which can make at maximum q queries can make a memory access which is bounded by n and can run means it is runtime is bounded by tau. So, then it basically returns a guessing vector which is denoted as g which is nothing but g 1 to g o.

So now, if s does not belong to g; that means, if this key class does not belong to g, then it returns 0 which means the attack is a failure. And if it belongs to g, then the attack is successful, so, it returns 1. So, the success rate essentially can be defined as nothing but the probability that this experiment returns a 1. If it returns a 1, then it means like that s belongs to g and therefore, right the adversary which is an o as I defined is to an o order adversary is successful. That means, if it can give o possible members of it is guessing vector, then it is guaranteed or it has got you know like I would rather say it has got a high probability that the correct key class will belong to this guessing vector ok.

This probability has to be more in order I mean ideally should be 1. So, therefore, right the correct key class belongs in the guessing vector if the attack is success. So, now, note that in this experiment right we always have a remaining work load which is defined by o. So, this maybe sometimes an inflexible definition because we are not really you know like kind of giving a metric or giving a quantification on the remaining work load and here exactly the definition of guessing entropy comes into play.

(Refer Slide Time: 11:01)



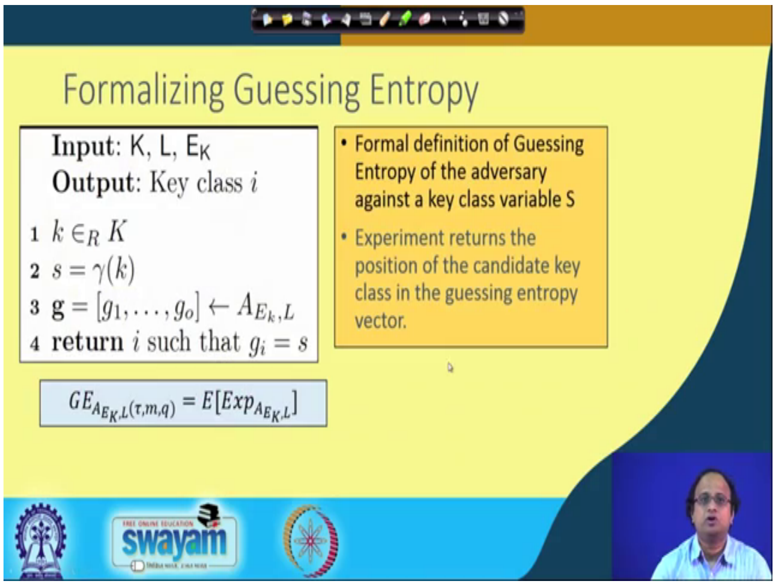
Guessing Entropy of an Adversary

- The above metric for an o^{th} order attack implies the success rate for an attack where the remaining load is o-key classes.
- The attacker has a maximum of o-key classes to which the required k may belong.
- While the above definition is fixed wrt. the remaining work load, we define guessing-entropy to provide a more flexible definition for the remaining work load.

So, therefore, right this is other definition that is of importance which is called as a guessing entropy of an adversary. So, the above metric for an oth order attack implies the success rate for an attack where the remaining load is o key classes because it comes up with the result that the guessing vector belongs to o key classes and the correct key can be belong to any one of them right. So, therefore, the attack has a maximum of o key classes to which the required key or required k can belong. While the above definition is fixed with respect to the remaining work load because the remaining work load is o is defined by o.

We define guessing entropy to provide a more flexible definition if I want to measure how the remaining work load reduces or diminishes with evolution or with maybe you know like more number of observations.

(Refer Slide Time: 11:52)



Formalizing Guessing Entropy

```
Input: K, L, E_K
Output: Key class $i$
1 $k \in_R K$
2 $s = \gamma(k)$
3 $\mathbf{g} = [g_1, \ldots, g_o] \leftarrow A_{E_k,L}$
4 return $i$ such that $g_i = s$
```

$$GE_{A_{E_K,L}(\tau,m,q)} = E[Exp_{A_{E_K,L}}]$$

- Formal definition of Guessing Entropy of the adversary against a key class variable S
- Experiment returns the position of the candidate key class in the guessing entropy vector.

So, if you want to measure that then we will slightly modify this experiment. So, what we will say is that, suppose again your input is K L and E K, the output class right now we will say it to be a key class which is defined as i or denoted as i. So, again what I do is, I take the key which is the correct key, I map it by the gamma function to s.

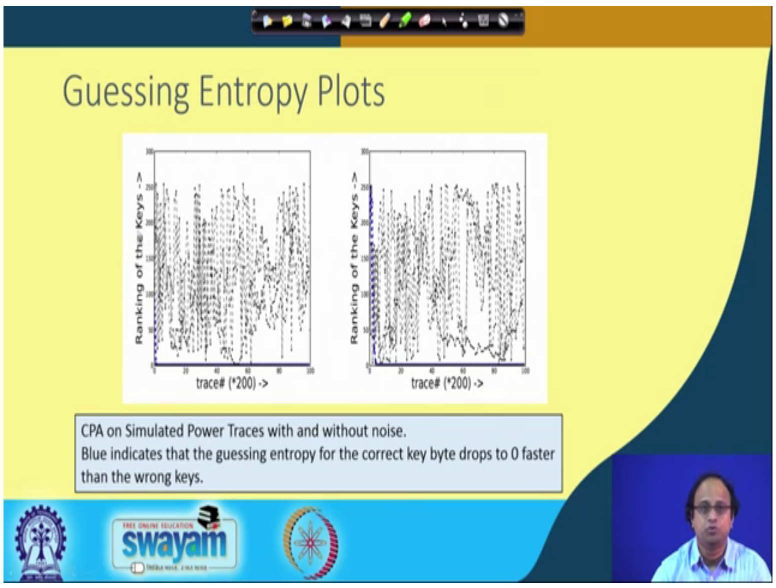
I again obtain this guessing vector which is g 1 to g o, but now I do not just say whether the attack is successful or not rather I try to return the position in this guessing vector. Because if I can tell the position in this guessing vector then it means that there are for example, if it is it starts from g 1 and if g 2 is my correct key; correct key, I mean g 2 is where you know like is equal to s, then that means, right g 1 and g 2 are both possible ok. And that means, that the remaining work load is essentially proportional to this returned i. And ideally, I will want that s which is my correct key candidate should be the first member of the guessing vector.

Because in that case right, I mean I have really resolved it, there is no remaining uncertainty about the key. So, therefore, right the attack in this case the guessing entropy is nothing but the expected value of i and that is denoted by this notation which says that it is a expected value of the or expectation of the you know like, if I denote is i to be a random variable like rather this i is an exact value, but if I denote capture this i by a random variable then the expectation of that random variable is what essentially gives us the guessing entropy of this experiment.

So, therefore, this gives me a formal definition of the adversary against a key class variable S and the experiment returns the position of the candidate key class in the guessing entropy vector. So, this is an important distinction difference between the first experiment and the second experiment. Although, they are quite closely connected there is a there is an interesting you know like I would say complementary nature of both the definitions. So, in one case it essentially gives you that the remaining entropy is o, so, it does not really bother about that. If just bother about the fact that whether you can bring the correct key class in within that o possible cases.

And in the second experiment right, we basically try to give you the position of that key candidate in the guessing vector ok. So, therefore, both of them can be useful in analyzing the side channel attacks.

(Refer Slide Time: 14:19)



Guessing Entropy Plots

CPA on Simulated Power Traces with and without noise.
Blue indicates that the guessing entropy for the correct key byte drops to 0 faster than the wrong keys.

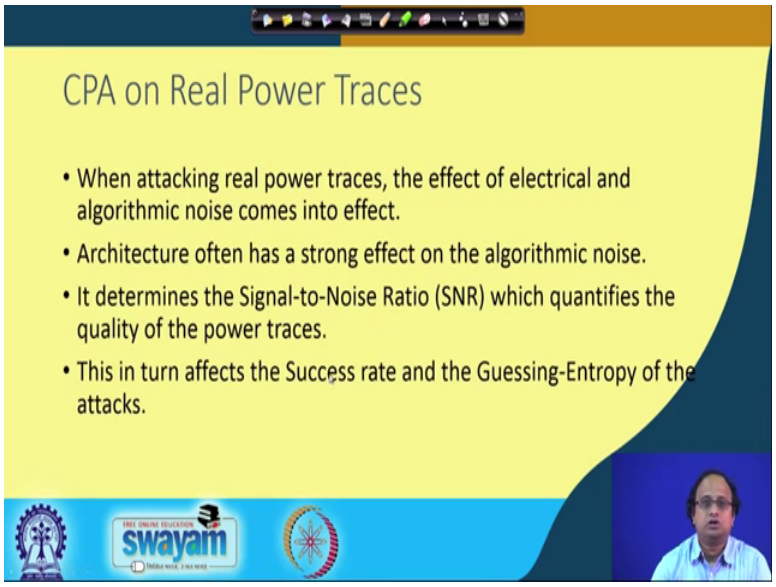
So, here are some experimental results. So, for example, you can see these are again you know CPA dump on simulated power traces as we have discussed before and it is with and without noise. So, the blue, so if you can observed eye that is a blue line and the blue line shows like there are like several wrong keys and you can see that the ranking of the key. So, if it is not very clearly, let me ask if any doubt.

So, it is basically the ranking of the key; that means, the position of the key in the guessing entropy in the guessing vector is basically plotted with the number of traces. So, what is expect is that with an evolution with more and more traces, the guessing entropy should get reduced to 0 because you know like it is at the beginning ok. So, it basically the correct key is at the top. So, therefore, what we are plotting out is the ranking of the key and the idea is that as the ranking becomes you know like from suppose it initially 10, then becomes 9 and then, gradually it becomes one.

Then; that means, it is at the beginning right and therefore, the entropy is minimum in that case. So, in this case, right there are several wrong keys which has been plotted the rankings of several wrong keys and there is a small blue line which basically tells us the guessing entropy of the correct key. So, if we can observe here, then with around something like more than 40 into 200 traces, you will find that the blue line shows that the correct key trace essentially, I mean the; I mean the ranking of the correct key goes to 0. That means it is identified.

Now, the interesting thing is that if you add noise, if you add you know like more noise to it, then you will find that this will becomes sluggish. And therefore, you know like what will happen is that it will take more number of traces to get down to 0 ok. And therefore, essentially that implies that if you add more noise, then the attack becomes difficult and the experiment or the attack takes more number of traces to reduce the guessing entropy to 0 or to essentially bring the correct key to the top of the least we or the guessing vector or. So, that is the objective behind the experiment.

(Refer Slide Time: 16:31)



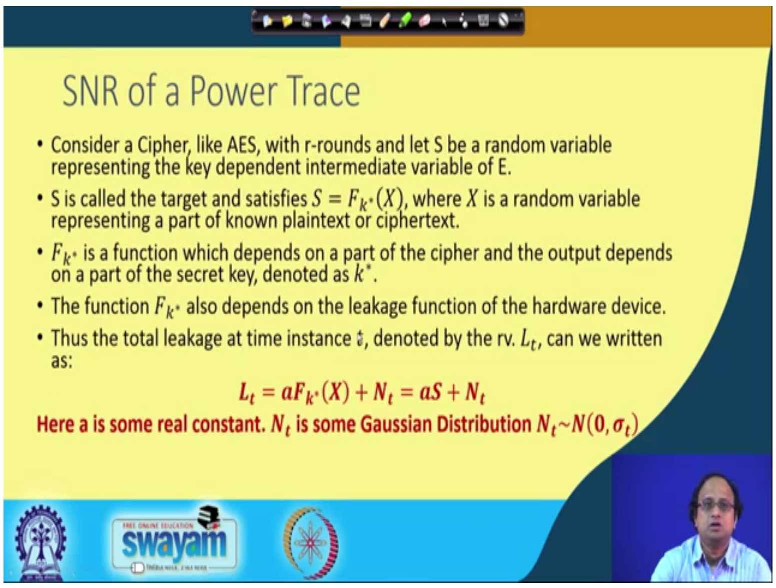
CPA on Real Power Traces

- When attacking real power traces, the effect of electrical and algorithmic noise comes into effect.
- Architecture often has a strong effect on the algorithmic noise.
- It determines the Signal-to-Noise Ratio (SNR) which quantifies the quality of the power traces.
- This in turn affects the Success rate and the Guessing-Entropy of the attacks.

So, therefore, now let us see that how we can what will happen if I you know like replace this simulated power traces with real power traces. So, when attacking real power traces of course, right the effect of electrical and algorithmic noise will come into effect which we have trying to kind of model with our Gaussian noise. But in real experiments there will be more phenomena and the architecture often as you got a strong effect on the algorithmic noise ok. So, as we will see in our discussion. It is often determine I mean, it often determines something which is called as a SNR or the Signal to Noise Ratio which quantifies the quality of the power traces.

This in turn effects the success rate and the guessing entropy of the attacks ok. So, this in turn has got a strong effect on the success rate as well as the guessing entropy of the attacks.

(Refer Slide Time: 17:20)



SNR of a Power Trace

- Consider a Cipher, like AES, with r-rounds and let S be a random variable representing the key dependent intermediate variable of E.
- S is called the target and satisfies $S = F_{k^*}(X)$, where X is a random variable representing a part of known plaintext or ciphertext.
- F_{k^*} is a function which depends on a part of the cipher and the output depends on a part of the secret key, denoted as k^*.
- The function F_{k^*} also depends on the leakage function of the hardware device.
- Thus the total leakage at time instance t, denoted by the rv. L_t, can we written as:

$$L_t = aF_{k^*}(X) + N_t = aS + N_t$$

Here a is some real constant. N_t is some Gaussian Distribution $N_t \sim N(0, \sigma_t)$

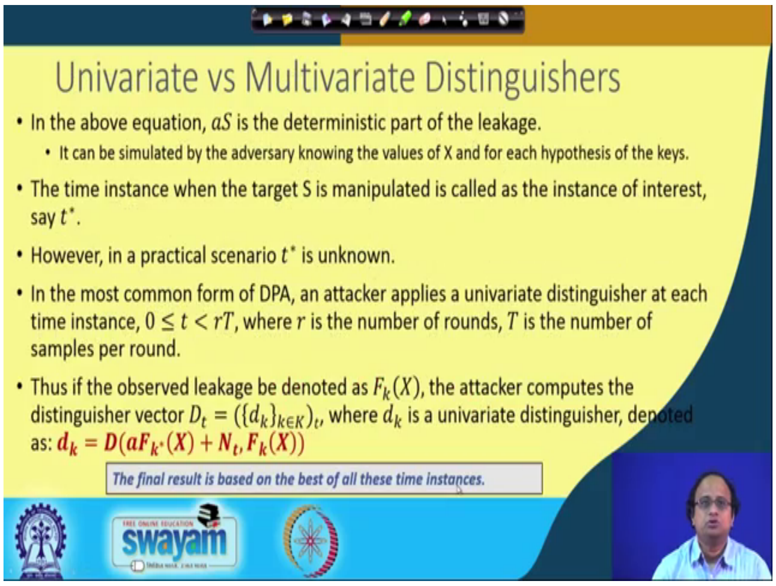
So, what is an SNR of a power trace ok? So, consider a cipher which is like AES which has got r rounds and let S be a random variable representing the key dependent intermediate variable of E. For example, it could be the target you know like S box input for example or the target S box output if you are doing the attack from the plaintext.

So, S is called the target. And it satisfies often S is equal to you can denote it as S is equal to F k star X. So, F k star X could be something like you know like that X is like is a random variable which is representing say a part of the known plaintext or the cipher text, we have got the correct key which is k star. And therefore, you apply them to get the value or estimate of F k star X. Now, note that F k star also depends upon the leakage function of the hardware device ok. So, and this is called as a deterministic portion of leakage. Why deterministic because you can fix it by fixing the values of X as well as k star. You can pretty much assume that this part of the leakage remains constant over several runs of the encryption.

So, therefore, the therefore, you know like the, but at the same time the total leakage will the something more. So, therefore, right there will be some portion which is proportional with this F k star X and that is denoted as this a F k star X where a is some real constant to indicate the proportionality. Now, this is basically superimposed with a is a portion which is called as N t ok. So, N t is nothing but the you know like it is basically a Gaussian. And as we have discussed right that every time instance there is an amount of electrical and algorithmic noise which is coming into place and that is essentially modeled as N t.

So, therefore, you can simplify this and write it as a S plus N t where F k star X is nothing but S. So, here is you know like I have as I have said that N t is some Gaussian distribution N t which belongs to say N 0 sigma t. Of course, you can say that it is not 0, but then if you say it is mu, then you can bring that component into a F k star x and you can make the noise component meaned around 0 ok. So, where is without loss of generality, we can write that the N t belongs to the normal distribution and 0 comma sigma t.

(Refer Slide Time: 19:40)



Univariate vs Multivariate Distinguishers

- In the above equation, aS is the deterministic part of the leakage.
  - It can be simulated by the adversary knowing the values of X and for each hypothesis of the keys.
- The time instance when the target S is manipulated is called as the instance of interest, say t^*.
- However, in a practical scenario t^* is unknown.
- In the most common form of DPA, an attacker applies a univariate distinguisher at each time instance, $0 \leq t < rT$, where r is the number of rounds, T is the number of samples per round.
- Thus if the observed leakage be denoted as $F_k(X)$, the attacker computes the distinguisher vector $D_t = (\{d_k\}_{k \in K})_t$, where d_k is a univariate distinguisher, denoted as: $d_k = D(aF_{k^*}(X) + N_t, F_k(X))$

The final result is based on the best of all these time instances.

So, therefore, right I mean there is a very important concept which is essentially when we talk about distinguishers which is called as univariate and multivariate distinguishers.

So, now let us understand what is meant by these two terms. So, in the above a S say is a deterministic part of the leakage, it can be simulated by the adversary knowing the values of X and for each hypothesis of the keys, like I make the you know like it is I am basically guessing on a portion of the key. So, I may make a guess on the key. Like there could be for example, as we have seeing 256 possible guesses of a part of the AES key when we are guessing say the 10th round bite for example. And we can essentially simulate the distribution, you know like simulate these portion.

So, now the time instance when the target when the target S is manipulated is called as the interest of is called as the instance of interest. Note that you know like if you are observing the power profile, not every time instance is when that particular you know like if you are targeting for example, a specific state that state comes into play, there could be other time instances when that state is not at all into; at all into the picture ok. It is not participating in the computation. So, therefore, right there is a specific time instance which we called as a instance of interest. So, I denote it as t star when it is you know like this particular state; this particular state is participating in the computation.

Note that were most practical cases, I do not know what is t star right we do not know what is the value of t star. So, therefore, in the most common form of DPA or in the some of the forms of what is called as you know like which is most practical and we as we have seen even in the context of CPA and difference of mean based DPA, an attacker applies a univariate distinguisher at each time instance. So, what is the univariate distinguisher? Univariate distinguisher is as we have seen the Pearson's correlation for example, you are applying the distinguisher on a single point in the trace.

You are applying it on a single leakage point in the trace. You are not applying it on multiple leakage points on the trace. So, if we apply it on a single leakage point on the trace, then you call that as a univariate distinguisher. So, what I do is therefore, at all time instances say t 1, t 2, t 3 ans. So, on I apply this distinguisher ok. Note that, if I have got a block cipher and block cipher is basically you know like iterating over say r rounds and every r round there are t instances when you are basically sampling or t is a number of samples per round, then the total value of small t can range from 0 to r T right.

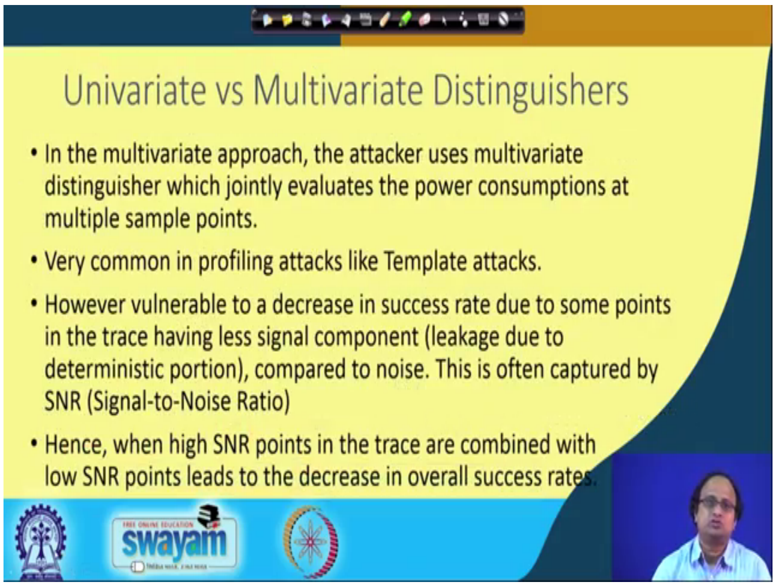
Because it can basically go on from 0 to you know like more precisely 0 to r T minus 1. So, therefore, the if the observed leakage be denoted as F k X, the attacker computes the distinguisher vector which is denoted as D t. So, what I do is therefore, I evaluate this distinguisher at every point on the leakage trace, every point on the leakage trace. So, therefore, as you can see these D t comprises of the result of this univariate distinguisher for all the possible key hypothesis, if there are 256 hypothesis, then you basically guess d 1, d 2, d 3 till, so until d to 256.

And the and how do you get any d k is as simple as that you basically apply your univariate distinguisher on F k X. So, F k X is the actual leakage because k is the actual key which you do not know as an adversary and this is the part which you are modeling. So, you are saying that ok, I know say I can simulate F k star X plus there is a Gaussian noise that I am trying to model. So, therefore, I if I basically take; basically take this component and if I say you know like for example, you know like correlate with F k X or if I apply any other distinguisher for that matter I can apply any you know like distinguishing function and then I get the value of d k ok. And therefore, the final result is based on the best among all these time instances.

So, therefore, by basically what I try to do is, I tried to kind of distinguish the correct key from the wrong key right. So, if I know that you know like that I want basically see that in which particular time instance, the separation between the correct key and the second correct key is maximum and that I tried to kind of find out at all these time instances and I find and I want to basically tell that at which time instance this separation is maximum. And that would probably be written as my correct key I mean correct time instance or t star.

So, this is one approach. This is typically what is called as a univariate way of you know like actually doing differential power attacks.

(Refer Slide Time: 24:33)



Univariate vs Multivariate Distinguishers

- In the multivariate approach, the attacker uses multivariate distinguisher which jointly evaluates the power consumptions at multiple sample points.
- Very common in profiling attacks like Template attacks.
- However vulnerable to a decrease in success rate due to some points in the trace having less signal component (leakage due to deterministic portion), compared to noise. This is often captured by SNR (Signal-to-Noise Ratio)
- Hence, when high SNR points in the trace are combined with low SNR points leads to the decrease in overall success rates.

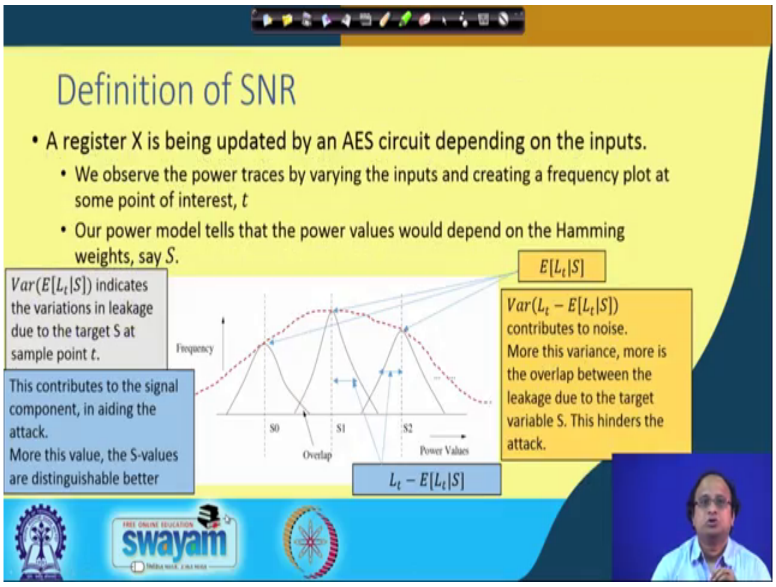
Now, as oppose to these right, in the multivariate approach what you can do is that the attacker will use multivariate distinguisher which jointly evaluates the power consumptions at multiple time samples ok. So, therefore, you know like and this very common in profile attacks, as we have seen the like that the DPA, we have been studying is more in example of a non profile attack in a profile attack this is brought you more common where you try to kind of apply this multivariate distinguisher on several points or multiple points on the traces.

This can be for example, very useful when suppose you are doing a computation not at one time instance but it is spread across time instances ok. So, these are you are doing say in a sequential manner, you are doing a computation. So, then what happens is that a computation is basically spread across different time instances in the power trace. So, in such scenarios, this is a more common form of doing attacks ok. However, you know like this is we have to basically apply this in a careful manner. So, if you basically the objective of applying a multivariate distinguisher is actually to amplify your signal or you know like we reduce the noise component.

But at the same point, you know like this technique could be vulnerable to a decrease in success rate due to some points in the trace having less signal which were trying to combine with the high SNR points. So, why is it kind of more common template attacks? Because, in template attacks when you are doing a profiling at that point, you know the key because you have full control on the device ok. And therefore, you can estimate the SNR properly and therefore, you can suitably you know like combine those points.

Whereas, in a non profile setting, you do not know the key and therefore, right there is a chance that if you are arbitrary applying this technique, then it may lead to you know like an overall degradation of the SNR. So, hence when high SNR points in the trace are combined with low SNR points, this could lead to a decrease in the overall success rate and therefore, right we have to carefully apply this technique.

(Refer Slide Time: 26:34)



Definition of SNR

- A register X is being updated by an AES circuit depending on the inputs.
  - We observe the power traces by varying the inputs and creating a frequency plot at some point of interest, t
  - Our power model tells that the power values would depend on the Hamming weights, say S.

$Var(E[L_t|S])$ indicates the variations in leakage due to the target S at sample point t.

This contributes to the signal component, in aiding the attack.
More this value, the S-values are distinguishable better

$E[L_t|S]$

$Var(L_t - E[L_t|S])$ contributes to noise.
More this variance, more is the overlap between the leakage due to the target variable S. This hinders the attack.

$L_t - E[L_t|S]$

So, this essentially brings us to something is called as a definition of SNR and that is the very vital definition we will take up that in the next class so.

Thank you for your attention.