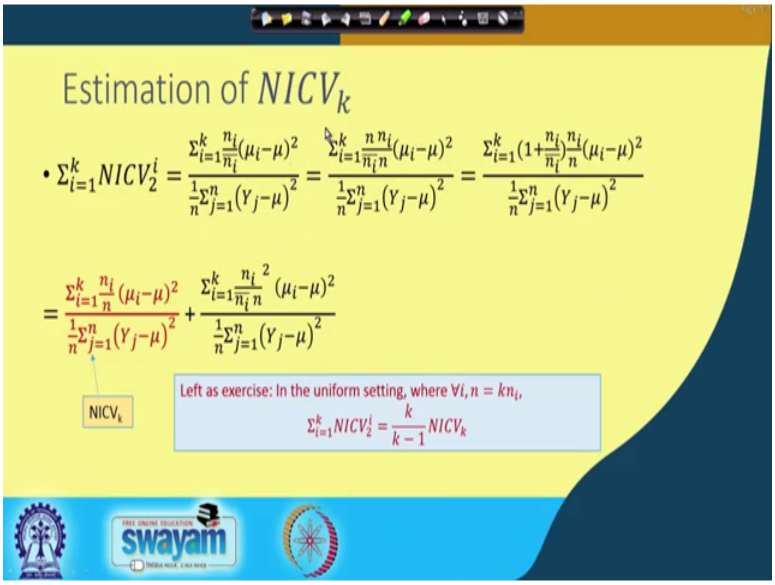
Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 42
Power Analysis - XVIII

So, welcome back. So, we shall be continuing our discussion on the homogenous test strategy.

(Refer Slide Time: 00:22)



Estimation of $NICV_k$

- $\Sigma_{i=1}^{k} NICV_2^i = \frac{\Sigma_{i=1}^{k} \frac{n_i}{\overline{n_i}} (\mu_i - \mu)^2}{\frac{1}{n} \Sigma_{j=1}^{n} (Y_j - \mu)^2} = \frac{\Sigma_{i=1}^{k} \frac{n\, n_i}{\overline{n_i}\, n} (\mu_i - \mu)^2}{\frac{1}{n} \Sigma_{j=1}^{n} (Y_j - \mu)^2} = \frac{\Sigma_{i=1}^{k} (1 + \frac{n_i}{\overline{n_i}}) \frac{n_i}{n} (\mu_i - \mu)^2}{\frac{1}{n} \Sigma_{j=1}^{n} (Y_j - \mu)^2}$

$$= \frac{\Sigma_{i=1}^{k} \frac{n_i}{n} (\mu_i - \mu)^2}{\frac{1}{n} \Sigma_{j=1}^{n} (Y_j - \mu)^2} + \frac{\Sigma_{i=1}^{k} \frac{{n_i}^2}{\overline{n_i}\, n} (\mu_i - \mu)^2}{\frac{1}{n} \Sigma_{j=1}^{n} (Y_j - \mu)^2}$$

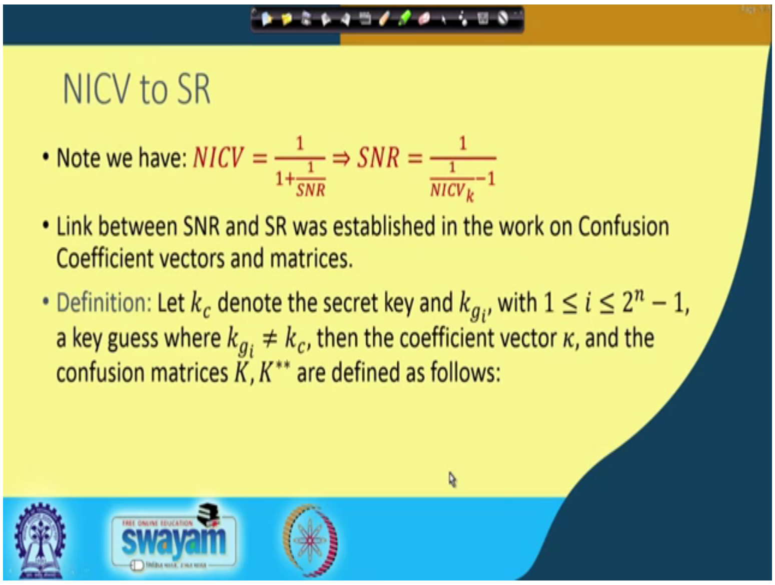
$NICV_k$

Left as exercise: In the uniform setting, where $\forall i, n = k n_i$,

$$\Sigma_{i=1}^{k} NICV_2^i = \frac{k}{k-1} NICV_k$$

So, we have been discussing about the relationship of NICV k with TVLA and this was where we stopped in the last class. So, we basically now have a pretty much relationship how to calculate the NICV or do we know how to calculate from TVLA the NICV k and essentially we know how to calculate NI, I mean we are basically related TVLA with NICV 2 and we have related NICV 2 with NICV k.

(Refer Slide Time: 00:45)



NICV to SR

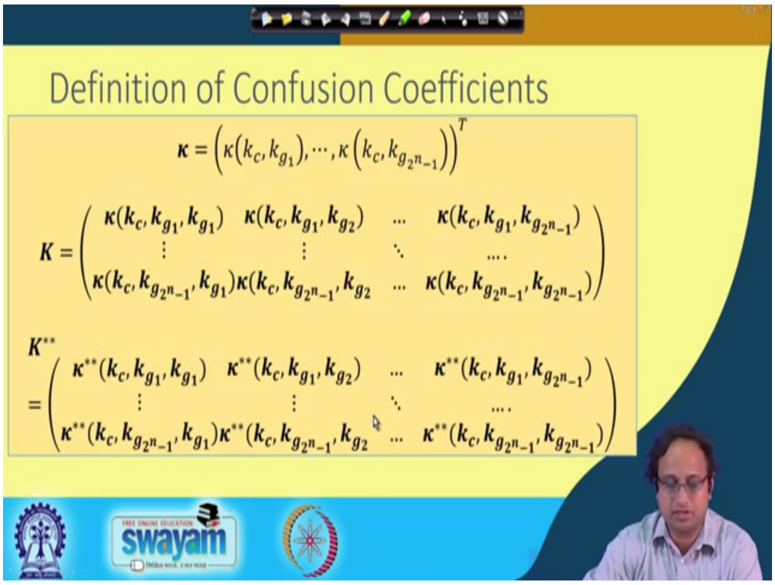
- Note we have: $NICV = \frac{1}{1+\frac{1}{SNR}} \Rightarrow SNR = \frac{1}{\frac{1}{NICV_k}-1}$
- Link between SNR and SR was established in the work on Confusion Coefficient vectors and matrices.
- Definition: Let k_c denote the secret key and k_{g_i}, with $1 \le i \le 2^n - 1$, a key guess where $k_{g_i} \neq k_c$, then the coefficient vector κ, and the confusion matrices K, K^{**} are defined as follows:

So, now we will see how we can relate the NICV to the success rate or the SNR to start with. So, we have got NICV which is equal to 1 divided by 1 plus SNR. So, you remember that this is something that I already defined in the previous class, ok. So, NICV you can actually write as 1 divided by 1 plus 1 divided by 1 plus 1 by SNR, and therefore, right you can essentially write the SNR as 1 divided by NICV k minus 1.

Note in this case NICV in the general setting can be written as NICV k. So, now, we have got a we have basically have got we have we have got the SNR. We basically can link till the SNR, but how can we get the success rate? So, the relationship between the success SNR and the SR was established in the work on confusion coefficient matrices and vectors and I will quickly guide you give an overview behind this result.

So, therefore, let us define the suppose you know like we have got k c which is your secret key and suppose I tell that k g i; that means, anything apart from the secret key is defined as k g i; where i varies from 1 to 2 to the power of n minus 1 which is the key guess which is do for k c. So, now if you do with then we define certain vectors and matrices. So, these vectors there are 3 vectors. So, first one is called as coefficient vector K or kappa and the confusion matrices are K and K star star.

(Refer Slide Time: 02:06)



Definition of Confusion Coefficients

$$\kappa = \left(\kappa\left(k_c, k_{g_1}\right), \cdots, \kappa\left(k_c, k_{g_{2^n-1}}\right)\right)^T$$

$$K = \begin{pmatrix} \kappa(k_c, k_{g_1}, k_{g_1}) & \kappa(k_c, k_{g_1}, k_{g_2}) & \dots & \kappa(k_c, k_{g_1}, k_{g_{2^n-1}}) \\ \vdots & \vdots & \ddots & \dots. \\ \kappa(k_c, k_{g_{2^n-1}}, k_{g_1}) & \kappa(k_c, k_{g_{2^n-1}}, k_{g_2} & \dots & \kappa(k_c, k_{g_{2^n-1}}, k_{g_{2^n-1}}) \end{pmatrix}$$

$$K^{**} = \begin{pmatrix} \kappa^{**}(k_c, k_{g_1}, k_{g_1}) & \kappa^{**}(k_c, k_{g_1}, k_{g_2}) & \dots & \kappa^{**}(k_c, k_{g_1}, k_{g_{2^n-1}}) \\ \vdots & \vdots & \ddots & \dots. \\ \kappa^{**}(k_c, k_{g_{2^n-1}}, k_{g_1}) & \kappa^{**}(k_c, k_{g_{2^n-1}}, k_{g_2} & \dots & \kappa^{**}(k_c, k_{g_{2^n-1}}, k_{g_{2^n-1}}) \end{pmatrix}$$

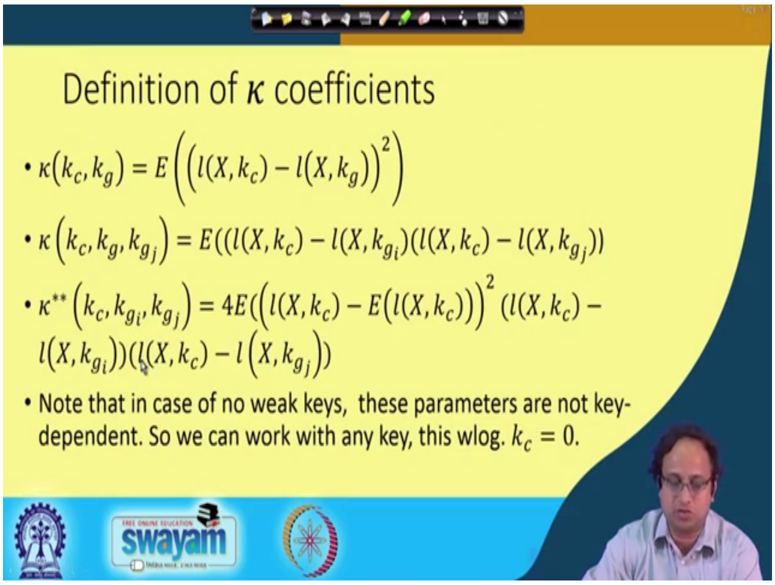
So, how they are defined is, are as follows. So, we have got this is your kappa which is your corresponding as you can see right that there are from 1 to 2 to the power n minus 1; that means, for all the wrong key parts remember that you are guessing only a part of the key ok, not the entire key. So, therefore, this exhausting analysis is possible, ok.

So, therefore, you basically have got in kappa there are 255 for example, if it stands for 250. Suppose you are doing it for AES, then this will have 255 values. So, we have basically doing this calculation of kappa with all with k c with k g 1, k c with k g 2 and so on, ok. Then you calculate the matrix k and the matrix k star again using some other definition of kappa where there are 3 parameters. So, in one parameter, there are I mean one definition of kappa there are 2 parameters and in another one there are 2 parameters, there are 3 parameters, ok.

So, again you note that how matrix has been defined. So, you have got k c as the first argument in all of them. The second argument is varying pretty much depending upon the row and the column position, ok. So, you have got k g 1, k g 1; k g 1, k g 2 so on till k g 1, k g 2 to the power of n minus 1 and in the if you see the last row it is k g 2 to the power of n minus 1 k g 1 and so on.

For K star star you have got another definition of kappa which is kappa star star for example.

(Refer Slide Time: 03:25)



Definition of κ coefficients

- $\kappa(k_c, k_g) = E\left(\left(l(X,k_c) - l(X,k_g)\right)^2\right)$
- $\kappa\left(k_c, k_g, k_{g_j}\right) = E((l(X,k_c) - l(X,k_{g_i})(l(X,k_c) - l(X,k_{g_j}))$
- $\kappa^{**}\left(k_c, k_{g_i}, k_{g_j}\right) = 4E(\left(l(X,k_c) - E\left(l(X,k_c)\right)\right)^2 (l(X,k_c) - l(X,k_{g_i}))(l(X,k_c) - l\left(X,k_{g_j}\right))$
- Note that in case of no weak keys, these parameters are not key-dependent. So we can work with any key, this wlog. $k_c = 0$.

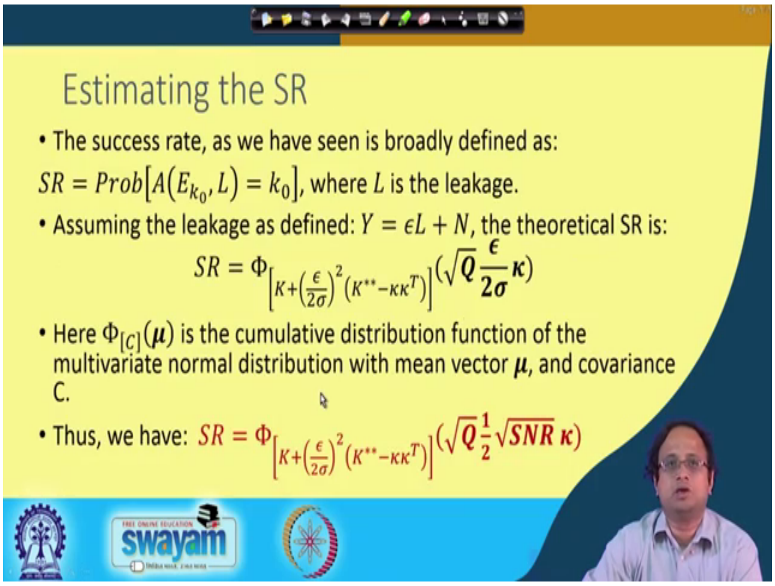
So, what how do you define? I am not going into the derivations behind them, but just writing down the results here. So, you can see that the kappa of k c comma k g is defined as the expectation of l X comma k c minus l X comma k g. Remember that l was my leakage function. So, therefore, I am using l to estimate the leakage, ok. So, this is your input and this is the corresponding key; this is the correct key and this is the guess of the key.

Again, you can calculate kappa k c comma k g, you know k g i comma k g j. So, there should be k g i here; k g i comma k g j, by using this expectation which is again you know like something like you have got the leakage of X comma k c minus leakage of X comma k g i this is one part. You multiply it by l of X comma k c minus l of X comma k g j and then you find out the expectation of this parameter.

Again, when you have got 3 terms, then you have got something like 4 expectation 4 times expectation of these whole square, again multiplied by l of X comma k c minus l of X comma k g i multiplied with l of X comma k c minus l of X comma k g j. So, at this point let us not you like, let us just try to kind of take this definitions and rather try to apply this in our context. So, idea is that here for all these parameters you can see that there is a term called k c which stands for the actual key.

But, in order to apply this, the good thing is that you need not know the key. If there are no weak keys then this parameters are not key dependent. So, therefore, right pretty much you can work without loss of generating with any key and you can actually set k c equal to 0 and do your calculation without even knowing what is the actual key.

(Refer Slide Time: 05:02)



Estimating the SR

- The success rate, as we have seen is broadly defined as:

$SR = Prob\left[A\left(E_{k_0}, L\right) = k_0\right]$, where L is the leakage.

- Assuming the leakage as defined: $Y = \epsilon L + N$, the theoretical SR is:

$$SR = \Phi_{\left[K+\left(\frac{\epsilon}{2\sigma}\right)^2\left(K^{**}-\kappa\kappa^T\right)\right]}\left(\sqrt{Q}\frac{\epsilon}{2\sigma}\kappa\right)$$

- Here $\Phi_{[C]}(\mu)$ is the cumulative distribution function of the multivariate normal distribution with mean vector μ, and covariance C.
- Thus, we have: $SR = \Phi_{\left[K+\left(\frac{\epsilon}{2\sigma}\right)^2\left(K^{**}-\kappa\kappa^T\right)\right]}\left(\sqrt{Q}\frac{1}{2}\sqrt{SNR}\,\kappa\right)$

So, now, there is a formula which you can directly apply. So, if you remember the success rate was defined by this probability, right. The probability of an adversary that if it is of operating on E k 0 and the observing the leakage L, it predicts the value of k key as correctly as k 0, then we say that the adder key successful. So, success rate is nothing, but the probability of the success.

So, if we assume that the leakage is as we have assumed Y equal to epsilon L plus N, then this theoretical SR can be found out by this cumulative distribution function. So, it is phi and you can observe that there are 2 parameters here, like there is a phi C and there is a mu, ok. So, C, so, this essentially stands for the cumulative distribution function of the multivariate normal distribution. We have seen already we have multivariate normal distribution in one of our previous classes where the mean vector is mu and the covariance matrix is C.

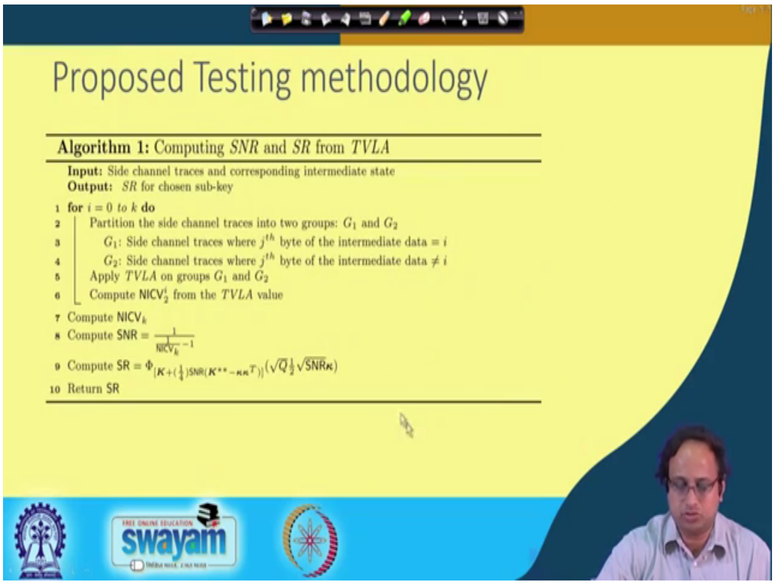
So, the covariance matrix C is defined by this formula which is K plus epsilon by 2sigma whole square into kappa into K star star minus kappa kappa transpose, ok. So, you can see that if you know K if you know K star star and if you know kappa then you can estimate this covariance matrix and all these things right are defined over here. So, these three things are defined over here using them you can know this covariance matrix.

What about this part? So, this is your mean, ok. So, this is parameterized by root Q into epsilon by 2 sigma into kappa. So this, remember that you know like, if you remember the expression for SNR, ok, so, then what you can do is you can easily manipulate this part and you can write it as root Q into half into square root of SNR into kappa.

So, why SNR? Why SNR because you can you remember right this is the square root of SNR, ok. So, the SNR is nothing, but epsilon square by sigma square. So, therefore, epsilon by sigma you can write as square root of SNR. So, therefore, now if you know the SNR from there you can estimate the success rate by finding out this cumulative distribution function and by finding out the value of this parameter ok, by finding out the value of this parameter and finding out the you know like the essentially estimating this probability function or the rather this cumulative distribution function.

So, therefore, what we have seen here now is that we have been able to go to the SNR and once you are able to estimate the SNR from there you can know what is the corresponding success rate.

(Refer Slide Time: 07:36)



Proposed Testing methodology

Algorithm 1: Computing *SNR* and *SR* from *TVLA*

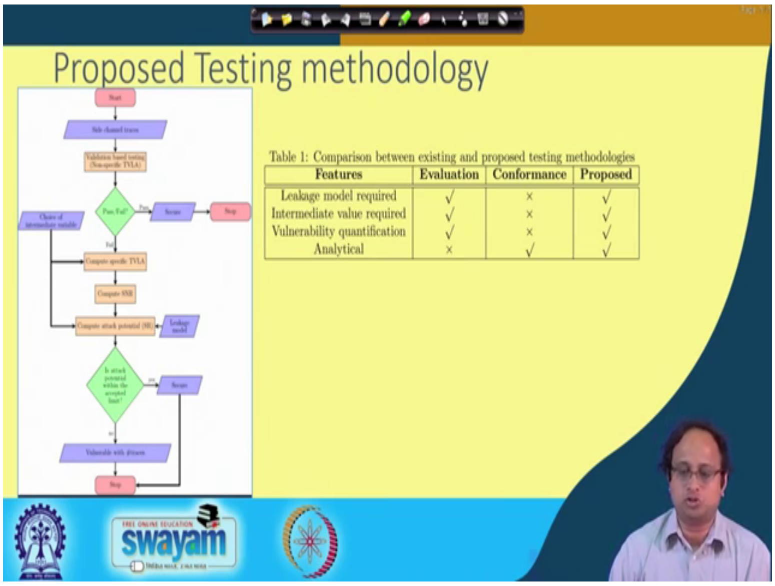
```
Input: Side channel traces and corresponding intermediate state
Output: $SR$ for chosen sub-key
for $i = 0$ to $k$ do
    Partition the side channel traces into two groups: $G_1$ and $G_2$
        $G_1$: Side channel traces where $j^{th}$ byte of the intermediate data $= i$
        $G_2$: Side channel traces where $j^{th}$ byte of the intermediate data $\neq i$
    Apply $TVLA$ on groups $G_1$ and $G_2$
    Compute $\mathsf{NICV}_2^i$ from the $TVLA$ value
Compute $\mathsf{NICV}_k$
Compute $\mathsf{SNR} = \frac{1}{\frac{1}{\mathsf{NICV}_k} - 1}$
Compute $\mathsf{SR} = \Phi_{[\boldsymbol{K}+(\frac{1}{4})\mathsf{SNR}(\boldsymbol{K}^{**}-\boldsymbol{\kappa}\boldsymbol{\kappa}^T)]}(\sqrt{Q}\frac{1}{2}\sqrt{\mathsf{SNR}}\boldsymbol{\kappa})$
Return SR
```

So, with this background we can now we all set to propose our test methodology. So, what we will do now is we will we will go from i equal to 0 to k, and we will do this partitioning we will partition the side channels traces into two groups G 1 and G 2. In G 1 we have got side channel traces where j th byte of the intermediate data takes a value i and in the G 2 it does not take the value of i, ok.

So, then we estimate, we apply the TVLA on this groups G 1 and G 2. Note that this is a this is an example of a specific TVLA test, because you are specifying the corresponding intermediate data and once you have got TVLA from there you can quickly derive what is NICV 2 i by using our formula. Once you have got NICV 2 i remember that and you have done that for all these k groups you can sigma them and from there you can estimate the value of NICV k. Once you have got NICV k you can estimate the SNR and then you can fix this SNR and from here you can get the success rate.

So, note that in this particular approach the leakage or the actual leakage model is only required at this step in the ninth step, whereas, the previous once you can do agnostic of that, ok. And, that is good because you can you know like try various leakage models and you can you can do your attack with various assumptions.

(Refer Slide Time: 08:53)



Proposed Testing methodology

Start
Side channel traces
Validation based testing (Non-specific TVLA)
Pass, Fail?
Pass
Secure
Stop
Choice of intermediate variable
Fail
Compute specific TVLA
Compute SNR
Compute attack potential (SR)
Leakage model
Is attack potential within the accepted limit?
yes
Secure
no
Vulnerable with #traces
Stop

Table 1: Comparison between existing and proposed testing methodologies

Features	Evaluation	Conformance	Proposed
Leakage model required	√	×	√
Intermediate value required	√	×	√
Vulnerability quantification	√	×	√
Analytical	×	√	√

So, therefore, this is my proposed test methodology. So, what we do is basically initially we do a non specific TVLA. We do a pass fail test. If it passes, then we say that it is fine whereas, if it fails then we still do a further analysis because as we know that you know like if there is a failed TVLA it may still mean that that will not work and in certain application scenarios like automotives or where you know like where essentially right you can afford you know like low cost counter measures or often we are you will be applying low cost counter measures.

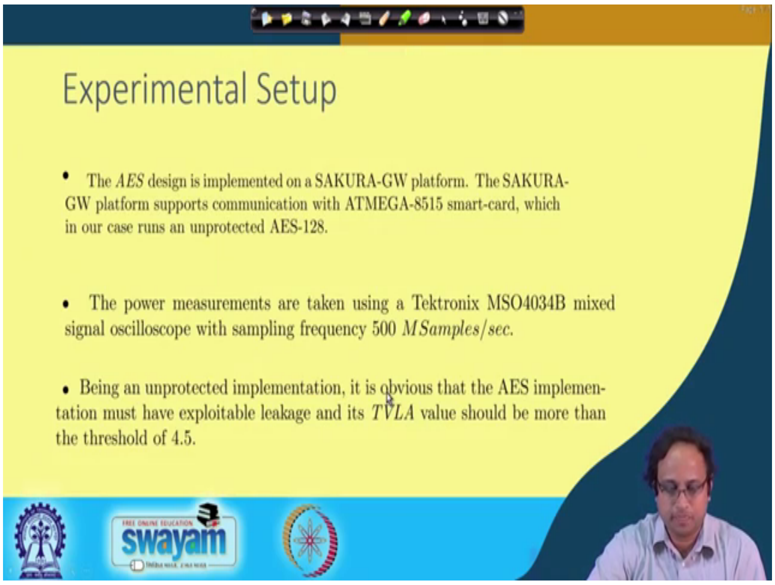
So, there we you know like if I want to do an analysis then probably you have to again apply your evaluation test methodology and you have to do the attack again and again that would be costly. Rather you can replace that part and do this approach where you if it fails you compute a specific TVLA and then you compute the SNR and then you compute the attack potential or the success rate.

So, here note that if you want to predict this you need this leakage model. So, leakage model is only required at this step. And, then again you answer whether the attack potential is acceptable within the acceptable limits, if it is fine then you it is secured; if it is not, then you again then you can say that the design is vulnerable with so many amount of traces.

If you remember that in the previous case right in order to do this attack, if you remember right there was a loop in the graph, so, because we are doing the attack again and again ok, but here that loop is gone. So, you did not do it again and again. So, this methodology quite fast in that sense. More or less right observe that the choice of intermediate variables is only require at two steps when you are computing the specific TVLA and where you computing the attack potential SR. So, this part you can actually paralyze by trying various target variables at the same time. So, you can kind of parallelize this process and you can even speed up your test methodology.

So, therefore, right if I compare with evaluation and conformance, then these are essentially some of the observations. You can see that it yes, it requires a leakage model, the intermediate value is also required, the vulnerability quantification is also done and it is also very analytical approach, ok. So, therefore, it is pretty much kind of you know like doing or rather nicely kind of combining both the approaches and is also not only doing a yes no analysis like your TVLA, but is also able to give you an actual quantification on the success rate the number of traces which you need to do that. So, therefore, it is I would say a more detailed analysis.

(Refer Slide Time: 11:25)

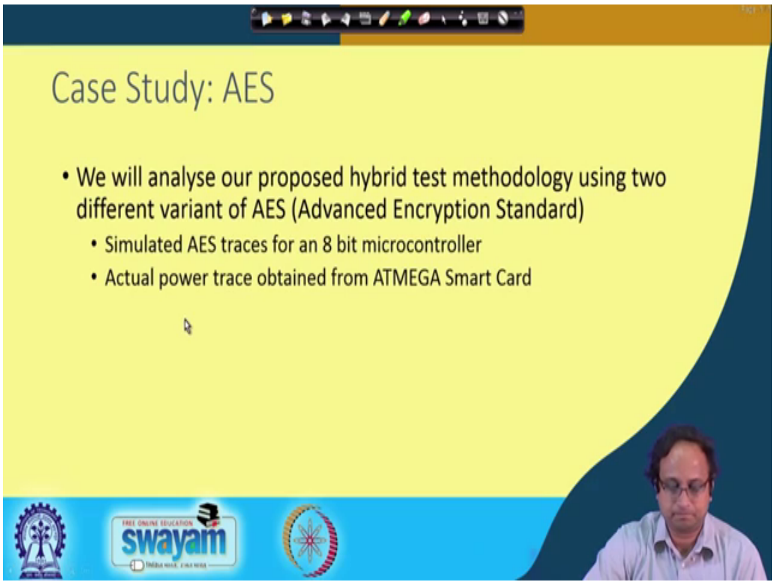


Experimental Setup

- The *AES* design is implemented on a SAKURA-GW platform. The SAKURA-GW platform supports communication with ATMEGA-8515 smart-card, which in our case runs an unprotected AES-128.
- The power measurements are taken using a Tektronix MSO4034B mixed signal oscilloscope with sampling frequency 500 *MSamples/sec*.
- Being an unprotected implementation, it is obvious that the AES implementation must have exploitable leakage and its *TVLA* value should be more than the threshold of 4.5.

So, let us now quickly look about if I have this approach and if I want to apply it on certain experiments. so, the experiment has been done on a SAKURA-GW platform where we have an ATMEGA-8515 smart-card, and we basically again use our power attack setup where we have it oscilloscope which is the mixed signal oscilloscope. And, then we basically target an AES implementation which of course, is an unprotected implementation and therefore, this should break or should be vulnerable if I do a TVLA analysis; that means, I should get a TVLA value which is probably more than say a given threshold of 4.5 for example.

(Refer Slide Time: 12:01)

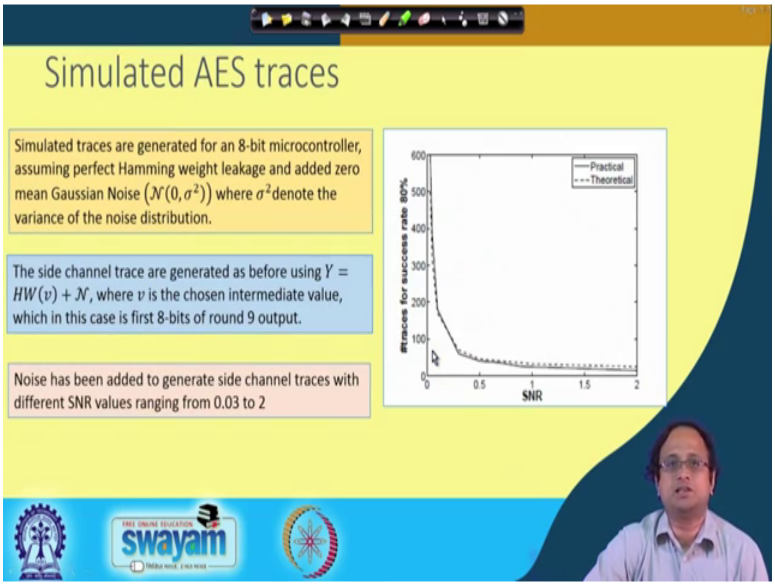


Case Study: AES

- We will analyse our proposed hybrid test methodology using two different variant of AES (Advanced Encryption Standard)
  - Simulated AES traces for an 8 bit microcontroller
  - Actual power trace obtained from ATMEGA Smart Card

So, we will analyse our proposed test methodology for 2 different variants of AES. Actually in one case I will be simulating the AES traces for an 8 bit microcontroller, in the other case we will be doing the test on an actual ATMEGA smart card.

(Refer Slide Time: 12:18)



Simulated AES traces

Simulated traces are generated for an 8-bit microcontroller, assuming perfect Hamming weight leakage and added zero mean Gaussian Noise $(\mathcal{N}(0,\sigma^2))$ where σ^2denote the variance of the noise distribution.

The side channel trace are generated as before using $Y = HW(v) + \mathcal{N}$, where v is the chosen intermediate value, which in this case is first 8-bits of round 9 output.

Noise has been added to generate side channel traces with different SNR values ranging from 0.03 to 2

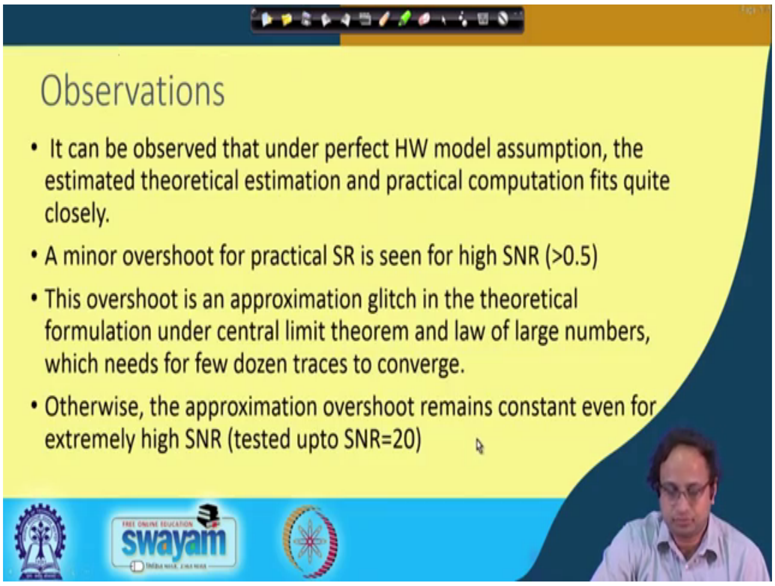
So, in the first case you can observe that this is an example on the simulated power traces again. We have basically simulated the traces for an 8-bit microcontroller assuming a perfect Hamming weight leakage and then we have added a zero mean Gaussian Noise to it.

So, the idea is that side channel traces a now generated by using this HW that is Hamming weight of v plus the noise and the this v is again you know like the chosen intermediate value which in this case is the first 8-bits of round 9 output. And, then we add a noise. So, we add noise and we kind of vary the SNR from something like 0.03 to 2.

You can observe that, so, we not only do, we basically do an estimation of the success rate or the traces required for a success rate of 80 percent. So, note that what I can do is I can go back to this equation and I can pretty much fix the success rate and from there try to estimate Q which is the number of observations. So, therefore, what I do over here is I basically estimate the number of traces required for a success rate of 80 percent; that means, I fix the success rate and then for a given SNR I basically plot our expected that is the theoretical graph or theoretical profile as well as we also do a real life power attack and try to match the results.

You can observe that the match is quite close and in fact, and if you observe very minutely you will find that when the SNR is high then the deviation is little more. So, why now because when the SNR is high then the attack is actually happening with very less traces, and therefore, right statistically you have got less samples. And, therefore, right that there is small overshoot which you can observe in this case. But, otherwise right there is a very close match and essentially our theory pretty much follows what is essentially happening experimentally.

(Refer Slide Time: 14:08)

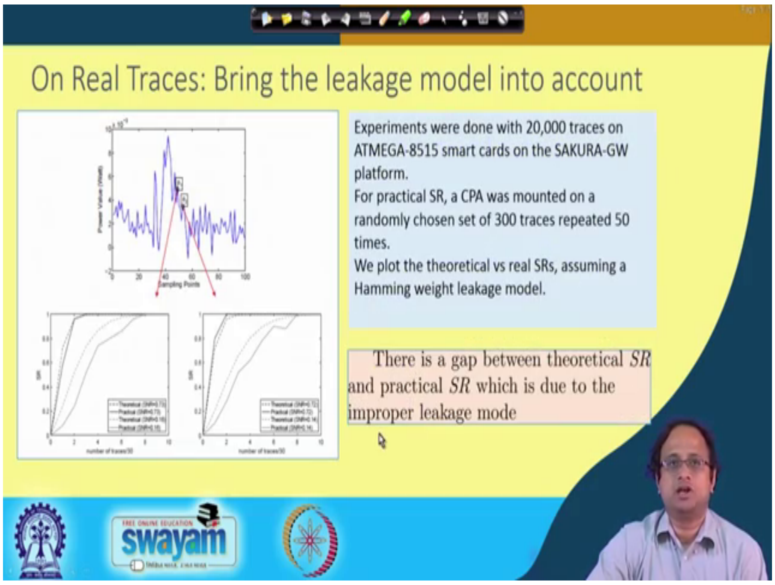


Observations

- It can be observed that under perfect HW model assumption, the estimated theoretical estimation and practical computation fits quite closely.
- A minor overshoot for practical SR is seen for high SNR (>0.5)
- This overshoot is an approximation glitch in the theoretical formulation under central limit theorem and law of large numbers, which needs for few dozen traces to converge.
- Otherwise, the approximation overshoot remains constant even for extremely high SNR (tested upto SNR=20)

So, it can be observed that under perfect hamming weight model assumption the estimated theoretical estimation and the practical computation fits quite closely a minor overshoot for practical SR is seen for high SNR points typically more than 0.5. And, the overshoot is an approximation glitch in the theoretical formulation because as we know that we have been applying central limit theory and the law of large numbers does not work when you have got few traces to observe, it does not really converge. And, otherwise right, the approximation overshoot remains constant even for extremely high SNR we have tested up to SNR equal to 20 and the match is quite consistent.

(Refer Slide Time: 14:46)



On Real Traces: Bring the leakage model into account

Experiments were done with 20,000 traces on ATMEGA-8515 smart cards on the SAKURA-GW platform.
For practical SR, a CPA was mounted on a randomly chosen set of 300 traces repeated 50 times.
We plot the theoretical vs real SRs, assuming a Hamming weight leakage model.

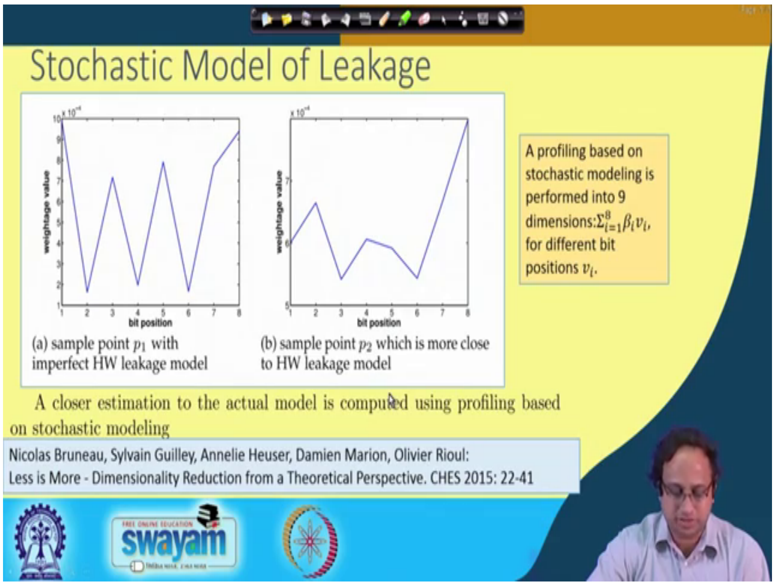
There is a gap between theoretical *SR* and practical *SR* which is due to the improper leakage mode

So, let us try to look into a real trace. So, in this case interestingly you see that this an ATMEGA experiment which is being performed. There have been several traces which has been taken from say large number of traces and here basically randomly chosen say set of 300 traces and done this analysis. So, we have basically highlighted here two points, P 1 and P 2 in order to show that there are two different points in the power trace graph, and then we do this analysis.

So, we basically add. So, this traces are typically take taken in a SAKURA-GW platform where the SNR is quite high. But in order to do our experiment we added noise to it and try to reduce the SNR and try to see how it compares in that case. You will see that when the SNR is high, then the match is quite consistent ok; like this particular trace over here is with for SNR equal to 0.73 whereas, here you see that when that SNR is something like 0.16 right, there is this deviation which you can observe.

Similarly, right in this case we are observing the point P 2 right, again you can observe that the match is high and, but still right when SNR is low, the match is not so high as probably this one, but still there is a mismatch. So, this gap between the theoretical SR and the practical SR you can study it further. It is probably because of the improper leakage model. Remember that we assume the leakage model at the end. So, there we assumed in this case hamming weight leakage model. So, somebody right can come up and give us a better leakage model and if that be so right then we can improve the accuracy of our estimate.

(Refer Slide Time: 16:22)



Stochastic Model of Leakage

(a) sample point p_1 with imperfect HW leakage model

(b) sample point p_2 which is more close to HW leakage model

A profiling based on stochastic modeling is performed into 9 dimensions:$\Sigma_{i=1}^{8}\beta_i v_i$, for different bit positions v_i.

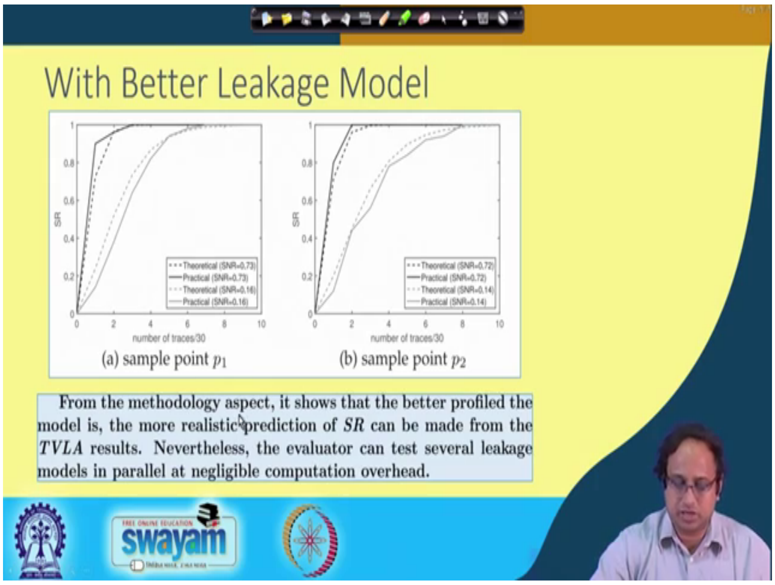
A closer estimation to the actual model is computed using profiling based on stochastic modeling

Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, Olivier Rioul: Less is More - Dimensionality Reduction from a Theoretical Perspective. CHES 2015: 22-41

So, how can we improve the leakage model? There are several works like here in one of the paper in CHES 2015, there was a work which tells you how to improve by something which is called as a stochastic modelling of the leakage. So, in stochastic modelling of the leakage right you assume that every bit has a different contribution. So, you have got say beta i into v i. So, in the Hamming weight model all the beta i values will be roughly the same.

So, this is the essentially the point p 1 and point p 2, you can see that the contribution in point p 2 for all the points are roughly the same. There are of course, an aberrations but here the aberrations are even more. And therefore, right the deviation also is more in case of point p 1, and that is exactly what you see here the deviation is high higher compared to this one when the SNR is essentially low.

(Refer Slide Time: 17:08)



With Better Leakage Model

(a) sample point p_1

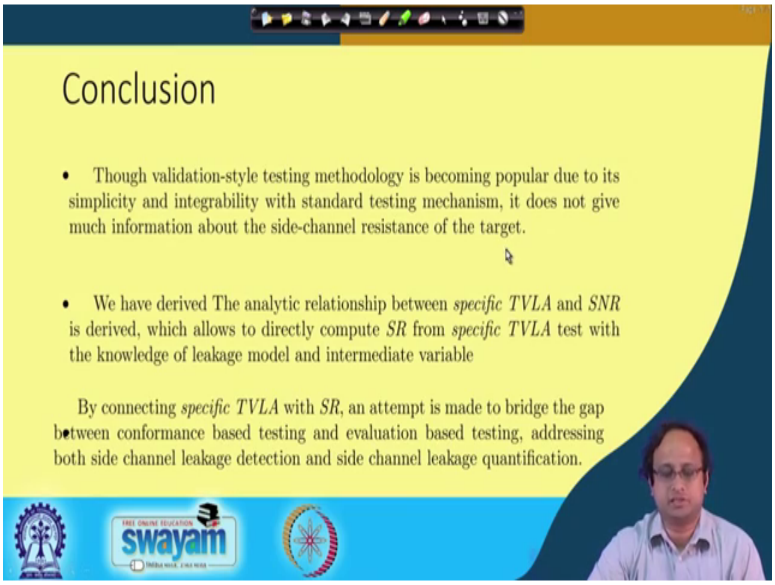
(b) sample point p_2

From the methodology aspect, it shows that the better profiled the model is, the more realistic prediction of *SR* can be made from the *TVLA* results. Nevertheless, the evaluator can test several leakage models in parallel at negligible computation overhead.

So, now what we did is we basically did a better, you know estimated the leakage model and then we again redid the experiment and now, you can see that if you compare again both point p 1 and p 2 you find that there is a drastic improvement and you can see that the match is more close.

So, therefore, from the methodology aspect it shows that the better profile the model is, more realistic prediction can be done on the of the for the success rate from your TVLA results. Nevertheless of course, the evaluated can test several leakage models in parallel and you know like an essentially can accelerate the process you know like without reducing any effect on the accuracy of your test because you can do pretty much evaluations of this leakage model in parallel and you can (Refer Time: 17:51)and see that which one matches close, and or maybe you can make you know like I would say like a (Refer Time: 17:57) case estimate depending upon various guesses on the leakage models.

(Refer Slide Time: 18:04)



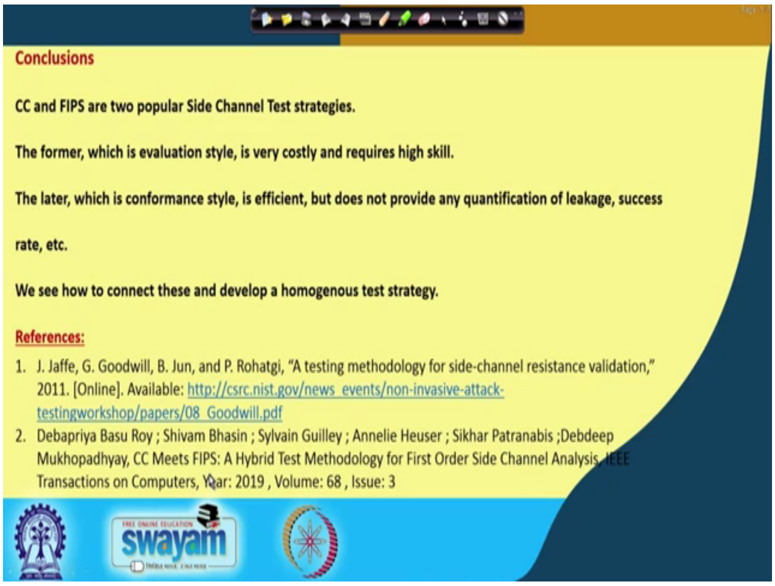
Conclusion

- Though validation-style testing methodology is becoming popular due to its simplicity and integrability with standard testing mechanism, it does not give much information about the side-channel resistance of the target.
- We have derived The analytic relationship between *specific TVLA* and *SNR* is derived, which allows to directly compute *SR* from *specific TVLA* test with the knowledge of leakage model and intermediate variable

By connecting *specific TVLA* with *SR*, an attempt is made to bridge the gap between conformance based testing and evaluation based testing, addressing both side channel leakage detection and side channel leakage quantification.

So, once you have basically this framework right, I mean essentially we are, we can discuss about some of the some of the final conclusions based on our observations.

(Refer Slide Time: 18:16)



Conclusions

CC and FIPS are two popular Side Channel Test strategies.

The former, which is evaluation style, is very costly and requires high skill.

The later, which is conformance style, is efficient, but does not provide any quantification of leakage, success rate, etc.

We see how to connect these and develop a homogenous test strategy.

References:

1. J. Jaffe, G. Goodwill, B. Jun, and P. Rohatgi, "A testing methodology for side-channel resistance validation," 2011. [Online]. Available: http://csrc.nist.gov/news_events/non-invasive-attack-testingworkshop/papers/08_Goodwill.pdf
2. Debapriya Basu Roy ; Shivam Bhasin ; Sylvain Guilley ; Annelie Heuser ; Sikhar Patranabis ;Debdeep Mukhopadhyay, CC Meets FIPS: A Hybrid Test Methodology for First Order Side Channel Analysis, IEEE Transactions on Computers, Year: 2019 , Volume: 68 , Issue: 3

So, in particular right, you see that let me just conclude this quickly. So, let me just conclude by these comments that CC and FIPS are two popular Side Channel Test strategies. The former, which is an evaluation style, is very costly and requires high skill whereas, the later which is a conformance style is efficient, but it does not provide any quantification of leakage or success rate.

What we have seen just now is that how we can combine both of them and we can get a confirmation style you know like combined with your evaluation style and although you start with TVLA you can come up with or estimating some of the matrix of an evaluation style testing methodology of course, with some you know like inputs on the corresponding leakage model. Of course, you need the leakage model, without leakage model we cannot estimate the success rate of a side channel attack.

So, here are some two important references of the discussion. The first one is actually the where the original TVLA test was proposed written by Goodwill and their collaborators and in the second paper you can find out the entire, you know the more elaborations on the proofs and derivations that we have discussed in this particular talk.

So, with I would like to say thank you. Thanks for your attention.