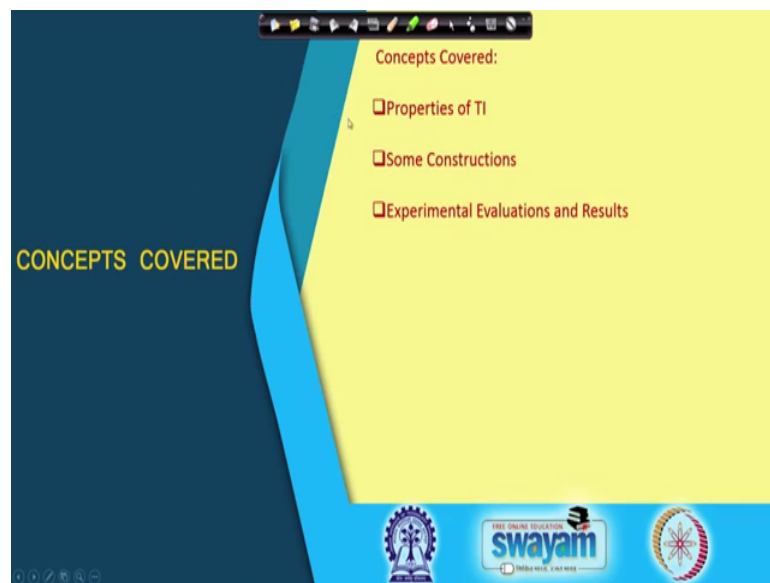


Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 45
Power Analysis Countermeasures (Contd.)

So, welcome back to this class on Hardware Security.

(Refer Slide Time: 00:21)



So, we shall be continuing our discussions on threshold implementation. So, as we have seen in the last class the idea of threshold implementations and where it is required to improve upon a basic masking scheme which was in secured against glitches. So, in today's class we shall be defining upon on the properties of TI and also see some potential constructions and some case studies based on that.

(Refer Slide Time: 00:40)

Correctness

$f(x_1, \dots, x_n) \rightarrow (a_1, \dots, a_m)$

For all $a \in F_2^m$, $A = f(\mathbb{X})$, implies that $a = \sum_i a_i = \sum_i f_i(x)$, for all x satisfying $\sum x_i = x$, $x \in F_2^n$

The diagram shows an input x being split into shares x_1, \dots, x_s . These shares are fed into a circuit with functions f_1, \dots, f_s to produce outputs a_1, \dots, a_s . The outputs are then combined to form the final output $A = [A_1 \dots A_m]$.

swayam

So, as we have defined in the last class right this is essentially the basic idea of threshold implementations where we are basically the idea is that if the input X and it has been broken up into shares then the threshold circuit right should be essentially give me the corresponding output shared. So, that if I combine them I should get a legitimate output which is corresponding to the actual data input X .

(Refer Slide Time: 01:02)

Uniform Masking

$x = (x_1, \dots, x_{s_x})$

- For all values with $\Pr[X = x] > 0$, let $Sh(x)$ denote the set of valid share vectors \mathbb{x} for x :

$$Sh(x) = \{ \mathbb{x} \in F_2^{ms_x} \mid x_1 \oplus \dots \oplus x_{s_x} = x \}$$
- $\Pr[\mathbb{X} = \mathbb{x} \mid X = x]$ denotes the probability that $\mathbb{X} = \mathbb{x}$ when the unshared input is x , taken over all the auxiliary inputs of the masking.

Uniform Masking: A masking X is uniform if and only if there exists a constant p such that $\forall x$ we have: if $\mathbb{x} \in Sh(x)$, then $\Pr[\mathbb{X} = \mathbb{x} \mid X = x] = p$, else $\Pr[\mathbb{X} = \mathbb{x} \mid X = x] = 0$, and $\sum_{\mathbb{x} \in Sh(x)} \Pr[\mathbb{X} = \mathbb{x}] = \Pr[X = x]$

Uniformity of a masking implies that the independence of the combination of any $s_x - 1$ shares, satisfying an (s_x, s_x) secret sharing scheme.

swayam

So, therefore, right as I said that there are certain important criteria which threshold implementations should satisfy, and one of the most fundamental properties is what is called as uniformity of masking.

So, let us define and see what it is it stands for. So, the idea is that for all values with probability of X equal to small x which is greater than equal to 0; that means, for all legitimate values of the input x , suppose I define $Sh\ x$ to be the you know like the set which essentially represents the valid shared vectors and the valid shared vectors will represent by the vector x , by this symbol for x .

So, that means, right the shared x will essentially consist of those shares remember that we had taken the input x . So, therefore, our representation was that we have taken x right we had taken x sorry we have taken x and broken up that or written up that as $S\ x$ shares, ok. Each x right is an n -bit value. So, therefore, it is also an n -bit value. So, every share is also an n -bit value. So, therefore, the total size of this is n into $S\ x$, and what essentially it means that $Sh\ x$ should consist of those vectors. So, this essentially this combination this tuple is denoted as this vector X for example, or written as in this way in my slides; that means, that if I take or if I XOR these components then I get back the original data which is x .

So, now we will basically try to see the conditional probability of probability of X equal to x ; that means the vector. So, this is again the random variable which is essentially denoting this particular value say you know like x or vector x . So, the probability that the shared values right for the mask or the shared vector essentially takes place takes a specific vectorial value, essentially shown over here given the input is x , given the input is x , I am trying to find out the probability that the sharing is a specific value, ok.

That means, like for example, suppose you know like that x is equal to 0, I am trying to find out the probability that my sharing, or my sharing right is essentially suppose 000 given that the input is x equal to 0, ok. Note that right if the x is equal to 0, it may also happen that the sharing right is equal to X equal to say 011 this is also a valid sharing of x equal to 0 because if I take an XOR 0 1 and 1, I get 0. So, therefore, right these are all my individual probabilities that I would like to calculate and what it turns out as that. So, therefore, I can you know like so, therefore, this conditional probability stands or denotes

a probability that X is some vectorial x when the in unshared input is x , taken over all the auxiliary inputs of the masking.

So, now the definition of uniform masking or masking X is said to be uniform if and only if there exists a constant P such that for all x ; that means, for all these values of x ; that means, for all values of x we have if this is a valid sharing, then this probability is a constant, or if this is not a valid sharing then this is condition probability is 0. As you can see right that if I ask you right, what is the probability that X equal to $0\ 1\ 0$ given that input x is 0? Now, this is not a valid sharing, right. So, therefore, it is trivial to know that this probability should be 0, right because this is not a valid sharing. But, what is important also to note that for all valid sharing this probability is a constant which is p , and the therefore, right it is important to understand why is this definition.

So, implication of this definition is actually following from secret sharing schemes, or essentially, but what it tries to say is that the uniformity of masking would imply the independence of the combination of any $S - 1$ shares; that means, right it will indicate that the value x is independent of any $S - 1$ shares; that means, right it will indicate that the value x is independent of any $S - 1$ shares ok; that means, right even if I give you $S - 1$ shares you will not get any information about the value of x if this particular property is maintained.

So, therefore, right this is an important result it is a very important result and therefore, right it is important to know or understand why this works.

(Refer Slide Time: 05:27)

Proof

• Define, X_i as the r.v denoting the i^{th} share. Then X_{-i} denotes the vector without the i^{th} share

• If masking is uniform $\Rightarrow X_i$ and x are independent for any i .

•
$$\Pr[X = x | X = x] = \Pr[X_{-i} = x_{-i}, X_i = x_i | X = x] = \frac{\Pr[X_{-i} = x_{-i}, X_i = x_i, X = x]}{\Pr[X = x]}$$

$$= \frac{\Pr[X = x, X_i = x_i]}{\Pr[X = x]} \Pr[X_{-i} = x_{-i} | X = x, X_i = x_i]$$

The last factor equals 1 when $x \in Sh(x)$ and zero otherwise.

Thus, $\forall x, \Pr[X_{-i} = x_{-i} | X = x] = p$

$\therefore \Pr[X_{-i} = x_{-i}] = \sum_x \Pr[X_{-i} = x_{-i} | X = x] \Pr[X = x] = p = 2^{n(1-s_x)}$

Handwritten notes on the slide include: $\Pr(A|B) = \frac{\Pr(A, B)}{\Pr(B)}$, $\sum_x p \Pr[X = x] = 1 = p$, and a diagram showing a vector x with one element x_i circled and labeled 'i'.

So, in order to know that we have to basically get into the proof of this result, so, now we will basically use a particular notation where for example, this vectorial X_i is basically representing a random variable or denoting a random variable which is denoting the i -th share for example, and if I write X_{-i} then; that means, right I am basically considering the fact that this particular or basically the vector without the i -th share.

So, therefore, what we are trying to prove is that if the masking is uniform; that means, if the previous definition is maintained, then this implies that actual data which is x is independent of this part which is essentially x_{-i} which means that the basically represents a vector without the i -th share; that means, you know $S - 1$ shares. So, for that right let us see this conditional probability this essentially used or define in my definition of uniform masking.

So, this conditional probability says that probability of X equal to x which is the vectorial X given that the actual unmasked data is say small x . So, now we can basically pretty much write this share into two parts; one part is denoting by x_{-i} vectorial i , this is the i -th share and the part without the i -th share, because the share will have two parts conditioned on X equal to small x .

So, this you can actually apply the just get the definition here and you can write this as in the denominator it will be have probability of X equal to small x . In the numerator it is a joint probability of x_{-i} of these two parts will remain along with X equal to small x , ok.

So, that is essentially written over here as $P(A|B)$ equal to $\frac{P(A \cap B)}{P(B)}$, ok. So, this is nothing, but just applying the fact that probability of probability of A given B is equal to in the denominator I have got probability of B in the numerator I have got probability of A comma B.

So, if we just apply this definition here and A is essentially given by these two parts and B is essentially just this part then you can write this ok, so, trivially. So, now, you can observe that with this particular now you can basically bring in. So, now, we bring in two parts, ok. So, one is essentially of $P(X_i = x_i)$ equal to $\frac{P(X_i = x_i | X = x)}{P(X = x)}$ in the numerator and the denominator. So, they basically cancel up, ok.

And, observe that this part is nothing, but you know like they as shown over here as the conditional probability of $X_i = x_i$ given $X = x$, ok. So, this is essentially this part and this part is essentially you know is the probability of $X_i = x_i$ in the numerator and the denominator. So, I basically just bring in this part and in the denominator I basically condition in on this part that is $X = x$, $X_i = x_i$.

Now, what is this part? So, you can observe that if x is a valid sharing; that means, you know like if this vector x is a valid sharing of x then this condition probability is 1, because if I give you the one of the inputs and if I give you the vector other than the i -th share then this particular thing is only one result right you basically know what is that value because you can suppose you consider the XOR for example, then this part is nothing, but the XOR of this part and these parts, and therefore, write this probability is equal to 1 in that case, ok. So, this will work out to 1.

On the other hand right if you find that this is an invalid share then that would imply that this is 0. So, therefore, right that is quite trivial. So, therefore, right that implies that what it implies is that since this is equal to P by my definition this conditional probability is P for. So, let us just consider the valid shares in that case for a valid share therefore, this probability or this condition probability is P and if this probability is P then that implies that this is equal to P .

So, therefore, it implies that probability of $X_i = x_i$ equal to P , ok. I mean the probability that your the part without the i -th share the vector is x_i given that $X = x$ is equal to P for all x and that would imply that if I want to calculate this probability that what is the probability that if I leave out the i -th share right the vector is

say \hat{x}_i . So, that I can easily apply the law of total probability and I can calculate this, right. I can condition this on probability of X equal to small x and multiply this with the corresponding conditional probability, right. So, and this I can do over pretty much \sum over x .

So, now note that this part or this part is always P is a constant P . So, therefore, I can write this \sum of probability of X equal to small x right over all x and this part will work out to 1, ok. So, I will get p here and what is this part? So, this you can easily see is basically right leaving out one share, ok. So, basically it is nothing, but 1 divided by 2 to the power of n into $S \times \text{minus } 1$ because this vector has got dimension of n times $S \times \text{minus } 1$ because one of the shares has been left out. So, this is the vector of the shares except that one of them has been removed.

So, therefore, this that the total vector was or dimension $n \times S$ if I remove one share then 1 goes. So, therefore, it is of n dimension $n \times S \times \text{minus } 1$, and out of all these possible values it can take only one possible value, and therefore, right this is equal to 2 to the power of $n - 1$ minus $S \times$. And, this essentially proves that you know like the fact that and actually you know like you have already proved your result because if you observe this particular fact then; that means, right this probability and this probability are same, and; that means, right this result is proved that X and \hat{x}_i they are independent, and the this proves right that even if I give you all the shares, but I do not give you one of the shares basically, right then; that means, right you are independent it does not leak any information about the actual data, in a in a very mathematical manner ok.

(Refer Slide Time: 12:01)

The slide is titled "Non-Completeness" and contains the following text:

- Masked Circuit:
 - $f_1(X_1, Y_1) = Z_1 \oplus X_1 Y_1$
 - $f_2(X_1, X_2, Y_1, Y_2) = ((Z_2 \oplus X_1 Y_2) \oplus X_2 Y_1) \oplus X_2 Y_2$
 - Note, f_2 depends on all the 2 shares. Therefore an attacker probing the corresponding wire can observe all the information required.
 - A TI on the other hand ensures that if the attacker probes d wires, it can only provide information for at most $s_{in} - 1$ shares, which is independent of the sensitive data.
- d -th order Non-completeness: Any combination of up to d component functions f_i of \mathbb{F} must be independent of at least one input share.

The slide also features logos for Swamyam and a small video inset of a speaker in the bottom right corner.

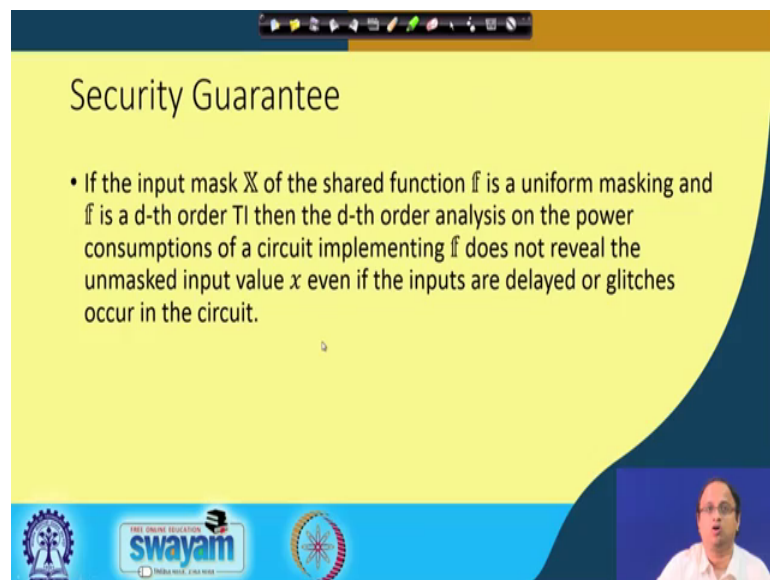
So, therefore, right I mean. So, now, what we will do is we will basically try to use this definition and you know like define another important definition which is called as non completeness. So, this was what is called as uniformity, the other result is what is called as non completeness. So, again let us take the mask circuit as we have seen in previously.

So, note that here f_2 for example, depends on all the two shares, ok. For example, here you can see that f_2 depends upon you know like all the 2-shares and therefore, right I mean you can observe that if there is an attacker which probes the corresponding wire; that means, probes corresponding f_2 can observe all the information required. So, for example, here you can see that it has got X_1 and X_2 it is also processing on Y_1 and Y_2 . So, it is processing on both on all the shares.

So, therefore, if there is an attacker who is probing the corresponding wires of for f_2 then basically it can it can pretty much observe all the information which is required and therefore, right this essentially in the proving model you can show that this is not secured in the presence of glitches because as I say that the you know like the probing captures the fact of glitches, captures the vulnerability due to glitches and here you can observe that if there is an attacker who probes right this particular circuit or this particular part of the circuit is able to observe all the required shares and therefore, right can pretty much obtain the information about the uncorrelated data because it knows all the shares if I know X_1 , X_2 then I know the value of X if I know Y_1 , Y_2 then I know the value of Y .

A TI on the other hand ensures that if the attacker probes the wires it can only observe or provide information for at most s in minus 1 shares and as we have already shown seen that this is independent of the sensitive data. So, therefore, this brings us to the definition of what is called as d -th ordered non-completeness which means that any combination of up to d component functions f_i of this vectorial function f must be independent of at least one input share, ok. So, it must be independent of at least one input share based upon that I will give my guarantees of what order of security I can protect against. So, this we can easily see right is not secured in that model because if I probe f_2 then all the shares get exposed. So, this is not this is not satisfying the you know even a first ordered non completeness requirement.

(Refer Slide Time: 14:22)



Security Guarantee

- If the input mask X of the shared function f is a uniform masking and f is a d -th order TI then the d -th order analysis on the power consumptions of a circuit implementing f does not reveal the unmasked input value x even if the inputs are delayed or glitches occur in the circuit.

THE SWAYAM EDUCATION swayam

So, therefore, right what we need is we need to kind of do something better and the security guarantees that if the input mask X of the standard function f is a uniform masking and f is a d -th order TI, then the d -th order analysis on the power consumption of a circuit implementing f does not reveal the unmasked input value x even if the inputs are delayed or if there are glitches which occurred in the in the circuit that is the security guarantee that TI would provide.

(Refer Slide Time: 14:49)

Sharing of Affine Functions $A(x) + B$

- An affine function $f(X) = A$ can be implemented with $s \geq d + 1$ component functions to thwart d -th order attacks.
- Construction:
 - $f_1(X_1) = A_1 = f(X_1)$.
 - For $2 \leq i \leq s$, $f_i(X_i) = A_i$, where f_i is f without constant terms.
- Eg, $f(X) = 1 \oplus X \Rightarrow f_1(X_1) = 1 \oplus X_1, f_i(X_i) = X_i, 2 \leq i \leq s$

Handwritten notes on the slide include:

- $f \rightarrow \{f_1, f_2, \dots, f_s\}$
- $f_1(x_1) = 1 \oplus x_1$
- $f_2(x_2) = x_2$
- $f_s(x_s) = x_s$
- $1 \oplus (x_1 \oplus \dots \oplus x_s)$

So, and again you know like it is very easy if I apply it for affine functions because a standard way of doing that is as shown here. Suppose, I have got an affine function f which I am applying on X to get A ; so, you can implement this with s shares where s is greater than equal to d plus 1 to thwart a d -th order attack.

So, what you will do is as follows. You will basically you know like write one part of the shared which is or one part of the circuit is as shown here and denoted as f_1 . So, note that since I have got s here I will have got f , I basically you know like implement f as a function of you know like f_1, f_2 and so on till f_s . So, you are basically you know like splitting the function f into shared functions.

So, you can see that the first function is shown here as just by just as $f_1(X_1)$ is equal to A_1 , and that is same as f on X_1 , ok; whereas, for the other parts right I essentially write as $f_i(X_i)$; that means, from 2 to s I write as f_i of X_i and that is equal to A_i , where f_i is nothing, but f , but without the constant terms, ok. So, note that in an affine function right you typically has got A into X plus B form, that is your form of an affine mapping.

So, consider example suppose you have got $1 \text{ XOR } X$ which you would like to implement using a TI mechanism. So, what we do is for the first component part we basically just write $f_1(X_1)$ and apply it on the same function $f(X)$, ok. So, you write that as $1 \text{ XOR } X_1$, whereas, for the other parts you basically implement using this, ok. So, as we have seen right let us consider let us consider an example to understand this. Suppose

you have got the function $f(X)$ which is equal to $1 \oplus X$ this is an example of an affine mapping.

So, what I do is basically I have got s shares. The first share I basically write is $f(1)$. So, this $f(1)$ you can actually write it as shown here as $1 \oplus X_1$, ok. So, you see that here we are using the same function as f just plugging in you know like X_1 as an input. So, if I plug in X_1 as input I get $1 \oplus X_1$ that is your $f(1)$. For the other parts of the share; that means, from 2 to s basically you basically again apply the same function, but without the constant term, ok. So, you have got $f(X_i)$ as you know like. So, in this part you have got $f_i(X_i)$; that means, $f_2(X_2)$ for example, is nothing, but X_2 and like this right you have got $f_s(X_s)$ is equal to X_s .

So, you see that in one of the shares you have got $1 \oplus X_1$ and you have got X_2 to X_s . So, if you combine the results you get $1 \oplus X_1$ with all the XORs of X_1 to X_s which is nothing, but your X . So, therefore, if you combine the results you get one XORed with the XOR of X_1 to X_s and this is nothing, but X which is essentially what you want to compute.

So, therefore, again you know like it is for affine mappings it is quite trivial and it can be done in this manner, ok.

(Refer Slide Time: 18:08)

What if the input is not uniform?

- Let, $(X, Y) \in F_2^2, A = f(X, Y) = XY$.
- Following is a 1st order TI:
 - $A_1 = f_1(X_2, X_3, Y_2, Y_3) = X_2Y_2 \oplus X_2Y_3 \oplus X_3Y_2$
 - $A_2 = f_2(X_1, X_3, Y_1, Y_3) = X_3Y_3 \oplus X_1Y_3 \oplus X_3Y_1$
 - $A_3 = f_3(X_1, X_2, Y_1, Y_2) = X_1Y_1 \oplus X_1Y_2 \oplus X_2Y_1$

The slide also features logos for Swamyam (Free Online Education) and IIT Bombay, along with a small video inset of a speaker in the bottom right corner.

But, on the other hand right I mean this becomes more complicated when you are applying it on non you know like another kind of functions. So, now, we will see you know like before I go into that and another important criteria which essentially would help me to apply this on non-linear functions,

So, for example, right let us consider a non-linear function as shown here as XY . So, here is a first order threshold implementation of this scheme you can note that I have basically split up X into three parts X_1 , X_2 and X_3 . So, one thumb rule that you can apply is that if this degree is 2 then potentially I required three you know like three shares, and I require you know like the implementing using a 3-shares and it can. So, here what we do is that if I have got X , Y I implement them in the shares like A_1 , A_2 and A_3 .

So, the way a A_1 is working is that A_1 essentially operates on as you can see on $X_2 X_3$, but it does not apply on X_1 's or Y_1 's, and that is why that this satisfies the criteria of it is non or incompleteness, ok. So, therefore, as we have seen right in the previous case when we want the non completeness definition then at least one of the shares should be left out and that is exactly what we see here. Here for example, X_1 or the first share has been left out, similarly here the second share has been left out, here the third share has been left out and these are the corresponding functions, ok.

If you combine this I leave it to you as an exercise to see that this should be leading to X Y , and that is that is the correctness of your definition ok, but now there is a very important observation if you can see that if you do an what is called as a uniformity analysis of the circuit, ok.

(Refer Slide Time: 19:50)

Uniformity Analysis

$$X = 0 \Rightarrow X = X_1 \oplus X_2 \oplus X_3 = 0$$
$$Y = 0 \Rightarrow Y = Y_1 \oplus Y_2 \oplus Y_3 = 0$$

Distribution of (A_1, A_2, A_3)

	000	011	101	110
000	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
011	(0,0,0)	(1,1,0)	(1,0,1)	(0,1,1)
101	(0,0,0)	(1,0,1)	(0,1,1)	(1,1,0)
110	(0,0,0)	(0,1,1)	(1,1,0)	(1,0,1)

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

So, if you do an uniformity analysis of this circuit then let us try to you know like do this exercise. Suppose, I fix X equal to 0 and Y equal to 0 if X is equal to 0; that means, that the shares X 1, X 2 and X 3 remember there are 3-shares should get XOR into 0; likewise Y 1, Y 2, Y 3 should get XORed into 0.

So, therefore, if I get a table where I will make a table right with all these shares like you know like say 000, 011, 101 and 110. So, note that these are the shares which essentially will get XORed into 0, ok. Likewise you know like I basically kind of tabulate it in this fashion that these are the shares for X and these are the shares for Y, and I try to find out what are the output circuits; that means, you know like the moment I have got say 0, 0, 0 and 0, 0, 0 I feed it into these equations and find out the value of A 1, A 2 and A 3 and I annotated in this table.

So, you can observe that if you want to do that then or if you do that basically you can observe that I get many cases where I have got 0, 0, 0. I have got you know like I think around seven cases where you have got 0, 0, 0. Likewise if you observe the case number of cases where you get 1110 1, 1, 0; so, you get 1, 1, 0 here you get 1, 1, 0 here you get 1, 1, 0 here. So, there are probably three cases where you get that. Likewise if you observe 1, 0, 1 you get a 1, 0, 1 here 1, 0, 1 here and A 1, 0, 1 here which is again three cases. If you observe 0, 1, 1; so, these are you know like you know like 0, 1, 1 and 0, 1, 1 here and A 0, 1, 1 here. So, again you have got three cases where you observe that.

So, now, imagine that this particular circuit. So, therefore, you know like from the security guarantee that we have seen if the input is uniformly masked and it satisfies the definition of non completeness then this circuit alone should not leak any information about the data. So, you can you know like so, likewise you know like the way we are done the analysis for X equal to 0 and Y equal to 0 you can continue and do it for other values of X and Y. So, remember that X and Y will have four possible values because there are 2-bits here. So, if you do that then you will get a larger table where you know like you take these statistics and we just dump into that table.

(Refer Slide Time: 22:11)

Uniformity Analysis

(x, y)	a_1, a_2, a_3							
	000	011	101	110	001	010	100	111
(0, 0)	7	3	3	3	0	0	0	0
(0, 1)	7	3	3	3	0	0	0	0
(1, 0)	7	3	3	3	0	0	0	0
(1, 1)	0	0	0	0	5	5	5	1

$$\frac{6 \times 3}{16} = \frac{18}{16}$$

$$\frac{18}{16}$$

As the input masking (X,Y) is uniform, and the circuit is a first order TI, the circuit itself does not leak vs a 1st order DPA adversary.

The average Hamming weights does not depend on (x,y).
 For example, if (x,y)=(0,1), average HW=(3x2)x3/16=18/16. If (x,y)=(1,1), average HW=(5x1)x3+3x1/16=18/16

But what if this circuit is fed as an input to a second circuit?

So, what you will get is something like this right here. So, you see that there are four cases and I observe the you know like the distribution of the output values a 1, a 2, a 3. So, these are the distribution of the output shares. For example I worked out this case like for 0, 0 where this has got 7 and the other 3 cases have got 3 values whereas, these are the invalid shares which will never appear.

Now, similarly you can work it for 0, 1; 1, 0 and 1, 1. Interesting right as we can see that the input masking x, y is uniform and a circuit is a first order TI satisfying all the properties that we have seen so far the circuit itself does not leak with respect to a first order DPA adversary. For example, right if you want to observe that you can see that the average humming weights for example, in the output for example, in all these three cases

you can observe that the distribution is exactly uniform, ok. So, exactly it the average hamming weight would be the same in these cases.

In this case which slightly looks different. You will also work out that you can see right that this is an interesting exercise to work out for example, the average hamming weight here is 2 into 3, ok. So, you see that a hamming weight here is 2 and there are 3 cases where this occurs. So, that 2 into 3, 6 likewise again this is 2 into 3 which is 6, 2 into 3 which is 6. So, you have got 6 into 3 which is a equal to 18, and there are totally right if you add up it is I think 7 plus 3 is 10 plus 3 is 13 plus 3 is 16. So, the average hamming weight is 18 by 16.

On the other hand, right so, if I if I do it for this particular case you will see that it is 5 into 1 plus 5 into 1 plus 5 into 1 which is 15 plus 1 into 3 which is 18. So, you still get this 18 by 16 and this is an average hamming weight does not depend upon your uncorrelated data, and therefore, right this is fine this is kind of secured against first ordered DPA, even in the presence of glitches.

However, if you do this exercise you know like; however, if you do this exercise and for you know like for you know like for the fact that what if you know if the circuit is fade as an input to a second circuit then how in the second circuit behave, because remember now the output distribution is essentially according to this pattern of a 1, a 2, a 3, ok. So, this is the mask which is fed to the following circuit.

(Refer Slide Time: 24:39)

Uniformity Analysis

- Let $B=g(Z,A)=ZA$, and this multiplication is implemented with similar equations.
- Assume Z is uniform, and A is the output of the previous circuit.

000	$5+5+5+5+5+1=31$
011	$5+5+1=11$
101	$5+5+1=11$
110	$5+5+1=11$

$2 \times 3 + 4 = 64$

swayam

So, if you do that right then you will see that you will basically get you know like you can do an analysis like this. So, this is what we do. So, you take Z and now you combine it with; so, this Z is again a uniform net, and you basically combine the output of this previous circuit again by applying an AND functionality ok, but this AND is again implemented in the same way as we have seen previously, ok.

So, now I assume that Z is uniform and A is the output of the previous masks. Suppose I fix x and y as 1 and 1, ok. So, that is the last row if you remember in the previous table. So, here are the possible distributions of the output mask like 001, 010, 100 and 111; you can see here 5, 5, 5 and 1 are the possible cases and z right essentially can take all these four values, ok. So, if you now do a combination you will see that the number of possible combinations are 5 into 4, because all these things can get combined with this. So, there are like 5 into 4, ok.

For example, this 5 can combine with this 4. So, it is 5 into 4 it is 20; likewise 20 into 3 20 into 3 plus this can combine with 4. So, you will have 64 such possible combinations. And, out of them right here is the possible you can observe that the, so, I am basically trying to see how many cases I get output 00. So, it turns out that there are 31 cases where the output mask will be 000, and so, basically you can work that out by just trying these combinations. So, likewise if the output is 011 it is 11 cases, here it is 11 cases and here it is 11 cases.

So, this you can get by combining these masks with these masks and again fitting into the equations of the three a 1, a 2 and a 3. So, therefore, this essentially is for the fact when x and y are 1 and 1 ok. So, you can actually take this and you can now again you know like plug it into the bigger table.

(Refer Slide Time: 26:36)

Uniformity Analysis

(x, y, z)	b_1, b_2, b_3							
	000	011	101	110	001	010	100	111
(0, 0, 0)	37	9	9	9	0	0	0	0
(0, 0, 1)	37	9	9	9	0	0	0	0
(0, 1, 0)	37	9	9	9	0	0	0	0
(0, 1, 1)	37	9	9	9	0	0	0	0
(1, 0, 0)	37	9	9	9	0	0	0	0
(1, 0, 1)	37	9	9	9	0	0	0	0
(1, 1, 0)	31	11	11	11	0	0	0	0
(1, 1, 1)	0	0	0	0	21	21	21	1

Note, for $(x,y,z)=(1,1,0)$, Average Hamming Weight= $11 \times 2 \times 3 / 64 = 33/32$, while for first 6 rows it is $27/32$. These deviations of means with inputs lead to a 1st-order DPA attack.

Note also if the function g was linear and was shared in the manner as seen previously, then circuit is still secure. The output distribution of f is carried to output of g .

And, if you do that right then you basically get a distribution as this. So, this is the specific case where I showed where I showed you know like that this is 1, 1 and z is 0. So, this is the one which I worked out here 1, 1 and 0 and you know the annotation is 31, 11, 11, right. So, this is again 31, 11, 11, 11. So, likewise you can work out the remaining cases.

So, now if you observe the hamming weights the average hamming weight for example, here is 11 into 2 into 3 because this is 0. So, this turns out to be thirty this turns out to be 33 by 32, whereas, if you try to do the exercise for the first six cases this will work out to something like 27 by 32. So, this shows that there is a deviation of the means with inputs which will lead to a first order DPA attack even now.

So, note that if this function g was linear then this problem would not have appeared because then that input distribution would have just got transformed into the output, and therefore, there would not have been any problem, but that implies that now the question is right if you have got an uniform mask and if you feed it in the next layer then how do you apply it, and that essentially something that we need to consider we need to tackle.

And, the what we will do is basically you know like if we apply this then we will see that how we can do that.

(Refer Slide Time: 27:52)

Uniform Sharing of a Function

- We need to make sure that the input of a sharing g which follows f is also uniform masking.
- **Uniform Sharing of a Function:** The d -th order sharing f is uniform if and only if:

$$\forall x \in F_2^n, \forall a \in F_2^m, \text{ with } f(x) = a, \forall a \in Sh(a), \text{ and } s_{out} \geq d + 1:$$
$$|\{x \in Sh(x) | f(x) = a\}| = \frac{2^{n(s_{in}-1)}}{2^{n(s_{out}-1)}}$$

swayam

And, for that we have got final criteria which is called as uniform sharing of a function. This says that if you want to make sure that the input of the sharing g which follows f is also a uniform masking. So, basically this leads us or this observation leads us to the fact that we need to properly cascade the non-linear functions. So, the idea is that we need to make sure that the input of the sharing g which follows f is also a uniform masking.

So, therefore, right the criteria for satisfying this is often called commonly as a uniform sharing of a function the idea is that the d -th order sharing f is uniform if and only if this criteria is satisfied therefore, we are not only bothered about input shares, but we also consider the output shares in this property, or in this criteria.

So, for that consider for all x which belongs to F_2^n and for all a which belongs to F_2^m ; these are the possible input values and the output values where x results in a ; that means, $f(x)$ is equal to a then for all the shares a which belongs these are the output shares a and s_{out} satisfying greater than equal to $d + 1$, then the cardinality of this particular set which basically tells me that if I take the input share x and if I obtain the output shares a , the number of input shares for which this is valid. Essentially is given by this ratio 2 to the power of $n s_{in} - 1$ divided by 2 to the power of $n s_{out} - 1$.

So, the rationale behind this particular criteria will soon be cleared if we were just getting to the proof of why this leads to an uniform sharing, ok. I mean why it leads to a uniform distribution of the output value, ok. So, remember that the problem was that in

the previous case we had a uniform distribution of the input which is essentially the you know like the input x , but the problem was that we did not get that uniform distribution in the output a . So, now we will see that why it works and for that let us see the corresponding proof.

(Refer Slide Time: 29:49)

Proof

- If the masking \mathbb{X} is uniform and the circuit f is uniform, then the masking \mathbb{A} of $a = f(x)$, defined by $\mathbb{A} = f(\mathbb{X})$ is uniform.
- We show: $\Pr[\mathbb{A} = a | \mathbb{A} = a] = \sum_{\substack{x \in Sh(x) \\ x, f(x)=a}} \Pr[\mathbb{A} = f(x) | \mathbb{A} = f(x)] \Pr(\mathbb{X} = x, X = x)$ Pr[X=x|X=y]=p
- The inner probability in the summation x=x
 $\text{term} = 2^{n(s_{in}-1)-m(s_{out}-1)} \Pr(\mathbb{X} = x | X = x) \Pr[X = x] = 2^{n(s_{in}-1)-m(s_{out}-1)} 2^{-n(s_{in}-1)} = 2^{-m(s_{out}-1)} \Pr[X = x]$ x=x
- Thus, we have $\Pr[\mathbb{A} = a | \mathbb{A} = a] = p = 2^{-m(s_{out}-1)}$, if $a \in Sh(a)$, and 0 otherwise.

So, the idea is that if you are masking X is uniform and the circuit f is uniform then the masking a of a equal to $f(x)$ which is defined by A equal to $f(X)$ is also uniform. So, now what we will want to do is that; so, likewise we had you know like we have seen that the criteria for a uniform mask given by this conditional probability as shown over here. So, in the previous case right if you remember right that a criteria was given by this notation right it was probability of X hat equal to say this right this is the our output this is the shared this thing shared value and X equal to small x . So, we basically calculated this probability and we saw that if this is a valid sharing then this is a constant p , that was my criteria.

If I want to get the similar property for the output right then I also need to look into this and I basically this is my corresponding conditional probability because here my actual value is A equal to small a I want to find out the conditional probability that there is a sharing which is occurring given the value of a equal to small a and I want to basically calculate this probability and I want to see that this is a uniform distribution.

So, this you can easily you know like obtained by you know like applying over all the possible input x 's. So, you know these are all the possible input shares of x , and you are basically pretty much calculating this by taking all the possible you know like. So, basically if I want to calculate this particular condition probability then I will be calculating it over all values of X equal to small x and their corresponding shares, ok.

So, X is equal to this and X hat is equal to small x hat, ok. So, this is my or mathcal x . So, these as so, I am basically calculating about the joint probability of these two these two these two terms, and I calculate. So, if this be so; that means, if my input X is equal to small x and my input sharing is denoted as mathcal X then I can feed it into my equation, or my shared or my TI implementation and I get an output share.

So, this output share essentially right essentially is giving denoted as A hat and that is equal to be right if I take f hat or you know like the TI implementation of f and apply x hat I mean apply it on x hat then essentially it turns out to be this conditional probability, where the input x is essentially you know like and I basically do a sigma such that over all x where $f x$ is equal to a , because $f x$ essentially has to give me a and that is why right I get that this a hat is equal to vectorial a .

So, therefore, right I mean if I now if you observe the inner probability in this summation the inner probability in this summation is 2 to the power of n times and this is essentially given by my criteria just we have seen previous to this it is 2 to the power of $n s$ in minus 1 minus m into s out minus 1, multiplied by this probability by this conditional probability into probability of X equal to small x because this again you know like this particular thing has been broken up into these two terms probability of X equal to small x multiplied by this conditional probability.

So, now what is this essentially has been already given by the fact that a uniform masking was obtained for input, and therefore, this is nothing, but 2 to the power of minus 1 a minus n sin minus 1. This is already what we have derived in the previous context of uniform masking and therefore, right if I multiply this with this these two terms cancels and I have got only this part remaining which is 2 to the power of minus m s out minus 1.

So, note that this part right if I do a sigma over all values of X then this part sums up to 1 and therefore, right I get this result that this condition probability of probability of A

equal to a hat given be you know like A equal to small a is equal to now 2 to the power of minus m s out minus 1 and is a constant p, and therefore, right it shows that the output masking is also now uniform. So, the input masking is uniform and as well as the output masking is uniform. Since the output masking is uniform now if I apply this as a input to a to a following circuit that also remains nicely intact ok. So, there is also no leakage as we have just now seen because of that, ok.

So, therefore, that essentially pretty much kind of helps us to ensure that we can cascade the non-linear functions.

(Refer Slide Time: 34:28)

Non-completeness: Example

$$S(x,y,z) = x \oplus yz$$

$$= (x_1 \oplus x_2 \oplus x_3) \oplus (y_1 \oplus y_2 \oplus y_3) (z_1 \oplus z_2 \oplus z_3)$$

$$S_1(x_2, x_3, y_2, y_3, z_2, z_3) = x_2 \oplus y_2 z_2 \oplus y_2 z_3 \oplus y_3 z_2$$

$$S_2(x_1, x_3, y_1, y_3, z_1, z_3) = x_3 \oplus y_3 z_3 \oplus y_3 z_1 \oplus y_1 z_3$$

$$S_3(x_1, x_2, y_1, y_2, z_1, z_2) = x_1 \oplus y_1 z_1 \oplus y_1 z_2 \oplus y_2 z_1$$

First order non-complete: Any one sub-function is independent of at least one share. We shall be dealing with First order TI throughout this presentation.

To protect a function with degree d , at least $d+1$ shares are required

The slide also features logos for IIT Bombay and Swamyam, and a small video inset of a speaker.

In fact, right what we will do is you know like just being more careful we will see. So, therefore, right what we do is as here is an example of non completeness. So, again you know if we take the example of $x \text{ XOR } yz$; so, you have got three 3-shares for x like x_1, x_2, x_3 and again y is here into y_1, y_2, y_3 ; z_1, z_2, z_3 .

Here you can see that it has been realized by three corresponding functions. So, this is a first order non-complete because you can see that every input right does not depend upon one shares and therefore, right it is satisfy in the requirement of a first order non completeness. And, also right we observe also observe that we can make a general comment here that is to protect the function with degree d at least d plus 1 shares are required, ok.

So, here the degree is 2 and therefore, we require at least 3-shares to realize this.

(Refer Slide Time: 35:18)

Composition of nonlinear functions

Separate non-linear functions with registers to prevent propagation of glitches

But, further right if you want to compose non-linear functions up a technique that we follow is that we will separate non-linear functions with registers to prevent propagation of glitches, ok. So, therefore, if you have got different layers then these layers are separated out by a registered layer so that the glitches here does get transformed into the next layer, ok, just to ensure that there is no accidental decays which happens because of you know combining cones or you know like circuit paths which are getting combined.

(Refer Slide Time: 35:51)

TI Example: TI of 2-input XOR Gate

- XOR is a linear function
- Let $c = a \oplus b$
- Let a_1, a_2, a_3 be the shares of a and b_1, b_2, b_3 be the shares of b i.e.
 - $a = a_1 \oplus a_2 \oplus a_3$
 - $b = b_1 \oplus b_2 \oplus b_3$
- Let c_1, c_2, c_3 be the output shares:
 - $c_1 = a_1 \oplus b_1$
 - $c_2 = a_2 \oplus b_2$
 - $c_3 = a_3 \oplus b_3$
- This is non-complete, uniform and correct
- In fact all the three properties can be achieved using just 2 shares!
- To summarize, TI design for linear functions are EASY!
- XOR + AND is functionally complete, so let us look into TI design of AND gate

So, finally, right I mean so, here is an example or just to illustrate that for example, if you take the case of AND gates right, why you essentially cannot realize it? So, the AND gate essentially you know like you cannot realize using two shares, as you can do for an XOR. For example, if I have got a 2-input XORs so, here it is shown as a XOR b; if I want to realize a in two shares, then I just write a_1 XOR a_2 . So, there is like their 3-shares like a_1 XOR a_2 XOR a_3 is equal to a; b_1 XOR b_2 XOR b_3 is equal to b and likewise right you can see that you know like you can also the idea here is that for XOR right you can easily do that because you can realize it using 3-shares of course, you can also realize using 2 shares.

So, for example, right what I will do is I will just write here as I mean what I do here is I basically take a as a_1 XOR a_2 XOR a_3 ; b as b_1 XOR b_2 XOR b_3 . If I want to write c_1 , c_2 , c_3 as your corresponding output shares then what I just do is that I write c_1 as a_1 XOR b_1 , ok. Note that I have left out the second one on the shares here. Likewise I can replace I can realize c_2 as a_2 XOR b_2 I can realize c_3 as a_3 XOR b_3 , this is non-complete and also uniform and also correct. In fact, all the three properties can be achieved by using 2 shares, although we have worked with 3 shares, but you can also work it with 2 shares.

To summarize TI design for non-linear functions are easy, but at the same time right it would be interesting to see that whether you can do the similar thing for an AND gate.

(Refer Slide Time: 37:30)

TI Example: TI of 2-input AND Gate

- AND is not a linear function
- Let $c = ab$
- Lets first see whether we can design a 2-share TI
- Let a_1, a_2 be the shares of a and b_1, b_2 be the shares of b i.e.
 - $a = a_1 \oplus a_2$
 - $b = b_1 \oplus b_2$
- The two output shares must contain the following 4 terms:
 - $a_1 b_1$
 - $a_1 b_2$
 - $a_2 b_1$
 - $a_2 b_2$
- In no way can these 4 terms be combined into 2-shares without violating non-completeness.
- So, 2 share TI of AND gate is not possible
- We need to increase the number of shares

Logos at the bottom: IIT Bombay, Swamyam, and another circular logo.

But, it turns out that it is not possible for AND gates because if you want to realize an AND gate with 2-shares then; that means, that you have broken up a into $a_1 \oplus a_2$ as in $b_1 \oplus b_2$ and therefore, the corresponding output shares must contain the following four terms $a_1 b_1$, $a_1 b_2$, $a_2 b_1$ and $a_2 b_2$ and you can do as an taking that exercise to see that these four terms right must be combined into 2-shares, but you cannot do in any way without violating the non-completeness definition, ok.

So, therefore, we put in right for example, $a_1 b_1$ and $a_2 b_1$ you can observe that this depends on both a_1 and a_2 therefore, it does not satisfy the non completeness definition. In fact, any possible combination would leak in would lead in to this conclusion and therefore, you need to increase the number of shares.

(Refer Slide Time: 38:13)

TI Example: TI of 2-input AND Gate

- With three shares even though non-completeness is satisfied, no three sharing can achieve uniformity without using extra randomness
- To achieve uniformity, non-completeness at the same time without using extra randomness, we need at least 4 shares
- Shown on the right is a uniform, non-complete sharing of the 2 input AND gate using 4 shares

$$A = X \cdot Y$$

$$X = x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

$$Y = y_1 \oplus y_2 \oplus y_3 \oplus y_4$$

$$A = a_1 \oplus a_2 \oplus a_3 \oplus a_4$$

$$a_1 = (x_2 \oplus x_3 \oplus x_4) \cdot (y_2 \oplus y_3) \oplus y_3$$

$$a_2 = ((x_1 \oplus x_3) \cdot (y_1 \oplus y_4)) \oplus (x_1 \cdot y_3) \oplus x_4$$

$$a_3 = (x_2 \oplus x_4) \cdot (y_1 \oplus y_4) \oplus x_4 \oplus y_4$$

$$a_4 = (x_1 \cdot y_2) \oplus y_3$$

So, you see here that if you want to make a TI of 2-input AND gate here is a solution worked out with 4-shares, ok; x_1 , x_2 , x_3 and x_4 . I leave it to you as an exercise to kind of judge that this satisfies the definition of uniformity, non-completeness and at the same time you know like that is correct at the end of the day. You can actually realize a circuit with less number of shares, but at the at the cost of extra randomness for example, I can reduce the shares from 4 to 3, but I need an extra I need actually extra random bits as indicated as r_1 , r_2 and so on. It is a challenge on how to realize a circuit with less shares and also with less randomness, ok.

(Refer Slide Time: 38:35)

TI Example: TI of 2-input AND Gate

- With three shares we can achieve uniformity using extra randomness
- The first example on the right uses 2-bits of randomness
- The second example uses one bit of randomness

$$A = X \cdot Y$$

$$X = x_1 \oplus x_2 \oplus x_3$$

$$Y = y_1 \oplus y_2 \oplus y_3$$

$$A = a_1 \oplus a_2 \oplus a_3$$

$$a_1 = (x_2 \cdot y_2) \oplus (x_2 \cdot y_3) \oplus (x_3 \cdot y_2) \oplus r_1 \oplus r_2$$

$$a_2 = (x_3 \cdot y_3) \oplus (x_1 \cdot y_3) \oplus (x_3 \cdot y_1) \oplus r_2$$

$$a_3 = (x_1 \cdot y_1) \oplus (x_1 \cdot y_2) \oplus (x_2 \cdot y_1) \oplus r_1$$

r_1 and r_2 are 2 bits of randomness

$$A = X \cdot Y$$

$$X = x_1 \oplus x_2 \oplus x_3$$

$$Y = y_1 \oplus y_2 \oplus y_3$$

$$A = a_1 \oplus a_2 \oplus a_3$$

$$a_1 = (x_2 \cdot y_2) \oplus (x_2 \cdot y_3) \oplus (x_3 \cdot y_2) \oplus r$$

$$a_2 = (x_3 \cdot y_3) \oplus (x_1 \cdot y_3) \oplus (x_3 \cdot y_1) \oplus (x_1 \cdot r) \oplus (y_1 \cdot r)$$

$$a_3 = (x_1 \cdot y_1) \oplus (x_1 \cdot y_2) \oplus (x_2 \cdot y_1) \oplus (x_1 \cdot r) \oplus (y_1 \cdot r) \oplus r$$

r is a unit of randomness

swayam

For example here you can see a circuit where we can still realize an AND gate with 3-shares, but now with the less amount of randomness requirement. For example, here there is only one random bit which is required and therefore, right this is a better implementation in with respect to cost because any random bit right essentially is not easy to develop or easy to generate and therefore, it comes with an accompanying cost, ok.

(Refer Slide Time: 37:18)

A Case Study of Lightweight TI based S-Box

$$f = XZW \oplus YW \oplus XY \oplus Y \oplus Z$$

$$b_1(X, Y, W) = X \oplus Y \oplus XW \oplus YW$$

$$b_2(X, Y, Z) = Z \oplus XY \oplus XZ$$

$$b_3(X, Z, W) = X \oplus W \oplus XZ \oplus ZW$$

$$f(X, Y, Z, W) = b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \oplus b_{12} \oplus b_{13}$$

$$b_1 = b_{11} \oplus b_{12} \oplus b_{13}$$

$$b_2 = b_{21} \oplus b_{22} \oplus b_{23}$$

$$b_3 = b_{31} \oplus b_{32} \oplus b_{33}$$

$$X = X_1 \oplus X_2 \oplus X_3$$

$$Y = Y_1 \oplus Y_2 \oplus Y_3$$

$$Z = Z_1 \oplus Z_2 \oplus Z_3$$

$$W = W_1 \oplus W_2 \oplus W_3$$

swayam

So, here is another example that you can observe for a lightweight TI based s-box. So, in order to realize that consider that we have got and this is these are some popular tricks which people have developed for implementing lightweight TI implementations. For example, if you consider this function f here you can see it is $XZW \oplus YW \oplus XY \oplus Y \oplus Z$.

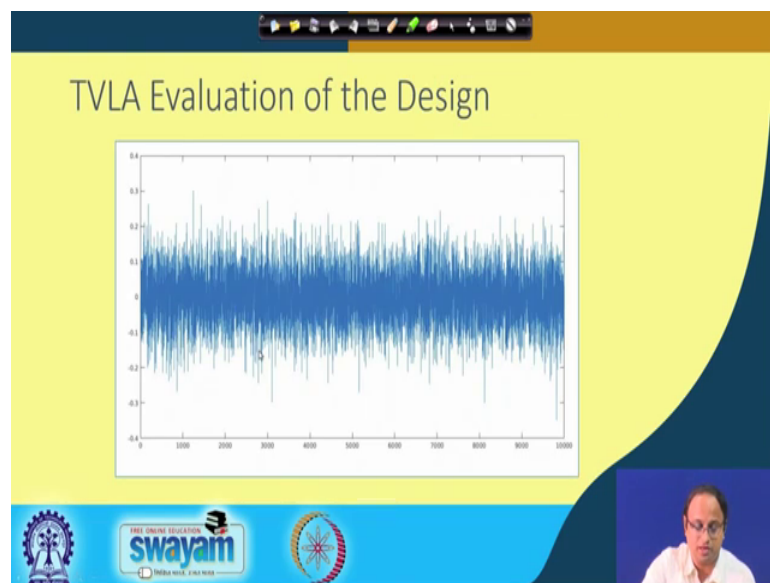
So, if you just straight forward apply threshold implementation on this you will see that you will require 4-shares because it has got three degrees and therefore, you will require 4-shares. But, you can do it cleverly by for example, in this case what you can do is you can observe or kind of define some intermediate variables say b_1 , b_2 and b_3 as follows. So, b_1 is nothing, but XOR of $X \oplus Y \oplus XW \oplus YW$ ok, b_2 is $Z \oplus XY \oplus XZ$, b_3 is $X \oplus W \oplus XZ \oplus Z \oplus W$.

You can see that if I combine b_1 , b_2 and b_3 right then so, therefore, you will observe that you know like you can basically you know like these are the three parts in which the circuit has been broken up for example, that is the three intermediate or you know like I would say like intermediate variables which has been defined. The purpose is that now you can write f in terms of b_1 , b_2 and b_3 . So, you can write f as b_1 applied on the inputs b_1 , b_2 and b_3 because you will see that now f you can write in terms of b_1 and b_2 and b_3 in this way because f is $b_1 \oplus b_2 \oplus b_1 \oplus b_3 \oplus b_2 \oplus b_3$. Again I leave it to you as an exercise to verify that this correct. You can observe that this form is exactly equal to this form, and this is nothing, but applying b_1 on the inputs b_1 , b_2 and b_3 . So, you can write this as b_1 applied on b_1 , b_2 and b_3 .

So, now you can realize you know like b_1 by applying you can apply TI on b_1 , but the advantage now is that since it has got a degree of 2 you can realize it with 3-shares. So, therefore, right I will break b_1 into b_{11} , b_{12} and b_{13} likewise b_2 into b_{21} , b_{22} and b_{23} ; b_3 into b_{31} , b_{32} and b_{33} , and you can basically realize b_1 , b_2 , b_3 using these equations. You can observe again this satisfies the required properties for TI, and likewise I can write b_{21} , b_{22} , b_{23} and b_{31} , b_{32} and b_{33} and since I know how to realize b_1 , now I can realize f also using these 3-shares where b_1 right will be broken up into these 3-shares b_{11} , b_{12} , b_{13} and b_2 will be broken up into b_{21} , b_{22} and b_{23} and likewise for b_3 .

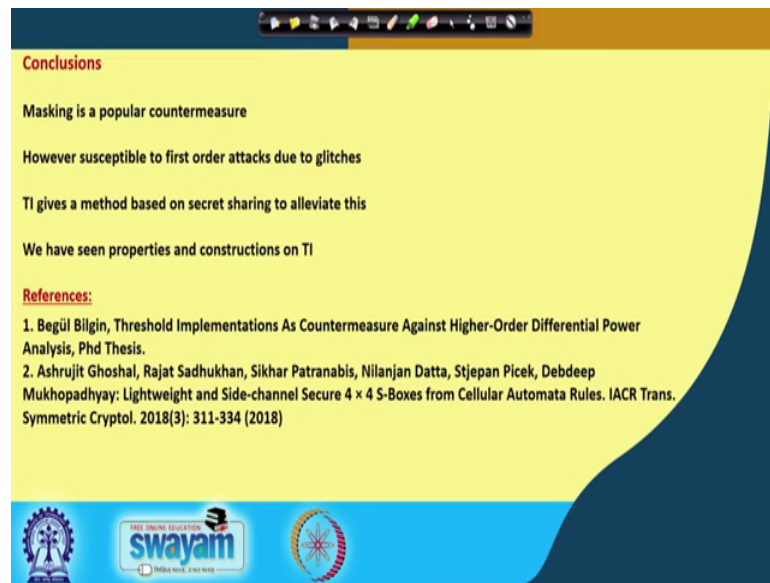
So, therefore, you can realize a entire TI in a controlled manner by not allowing you know like the circuit to blow up and the cost essentially because you have to ensure that at the end of day right that the cost for implementing the counter measure should be suitably controlled. So, you can kind of compare this with you know like an effort to straightforward apply a TI on this function where you will be requiring at least 3 plus 1 that is 4-shares, and therefore, this d and also I this non-linear circuit will significantly you know like contribute to the blow up of the area requirement.

(Refer Slide Time: 42:37)



So, if you realize this in this suitable way then at the end of the day if you remember right that we had discussed about TVLA evaluations. So, if you take this S-box and do a TVLA and then you can see that nicely that the leakage is within your threshold of 4.5 or plus minus 4.5 and therefore, this indeed gives you a security and you can see that there is potentially no leakage with respect to this kind of power attacks. So, this is this an evaluation for a first order DPA requirement.

(Refer Slide Time: 43:03)



Conclusions

Masking is a popular countermeasure

However susceptible to first order attacks due to glitches

TI gives a method based on secret sharing to alleviate this

We have seen properties and constructions on TI

References:

1. Begül Bilgin, Threshold Implementations As Countermeasure Against Higher-Order Differential Power Analysis, Phd Thesis.
2. Ashrujit Ghoshal, Rajat Sadhukhan, Sikhar Patranabis, Nilanjan Datta, Stjepan Picek, Debdeep Mukhopadhyay: Lightweight and Side-channel Secure 4×4 S-Boxes from Cellular Automata Rules. IACR Trans. Symmetric Cryptol. 2018(3): 311-334 (2018)

swamyam
FREE ONLINE EDUCATION
INDIA WISE. LEARN WISE.

So, to conclude write masking is a popular countermeasure. However, these are susceptible for first order attacks due to glitches. TI or threshold implementation gives a method based on secret sharing to elevate this. We have seen some properties and constructions on TI and here are some popular here are some references in particular I would take your notice to the first reference which is essentially a PhD thesis, but nicely gives and you know like having anecdote about how to implement TI and also see various implications and further implications on threshold implementations if you are interested to continue on this.

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