

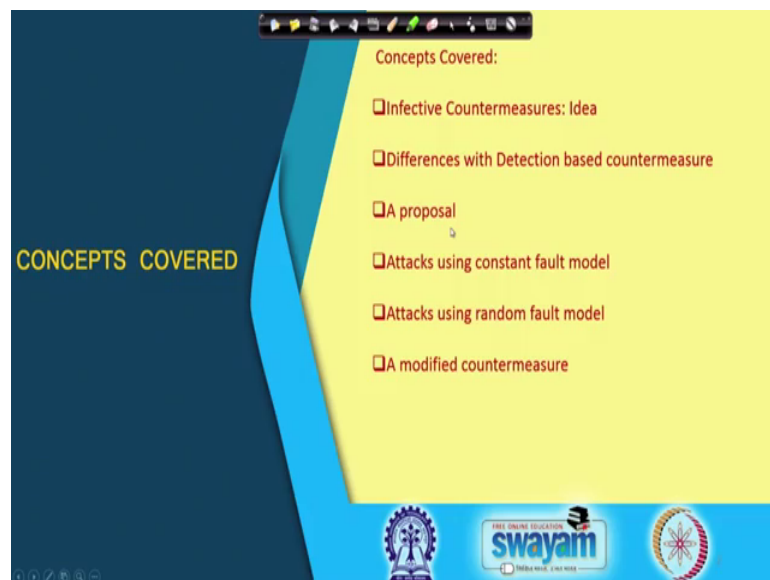
Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 52
Infective Countermeasures for DFA

So, welcome to this class on Hardware Security. So, we shall be continuing our discussions on fault attack countermeasures. In particular we shall be discussing about countermeasures, which is an improvement over needed detection based countermeasures all right this is kind of an alternative strategy to detective countermeasures or detection based countermeasures.

So, for example, right what we will be covering in today's class is we shall be going into the ineffective countermeasures idea. We shall be seeing some of the differences with detection based countermeasures, we shall be seeing a proposal of our new detection based countermeasure I mean a countermeasure which is proposed.

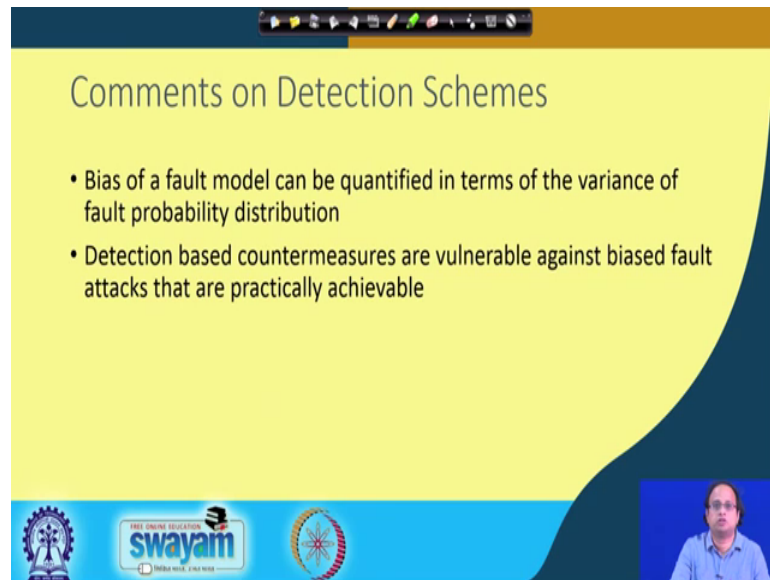
(Refer Slide Time: 00:56)



And then we will see we shall discuss about some potential attacks on the detection based countermeasures using two different fault models one of them is what is called as the constant fault model.

And the second one is a random fault model kind of similar to the differential fault attacks that we have already seen. And finally, we shall be trying to propose a modified countermeasure, which would be secured against these type of attacks.

(Refer Slide Time: 01:15)



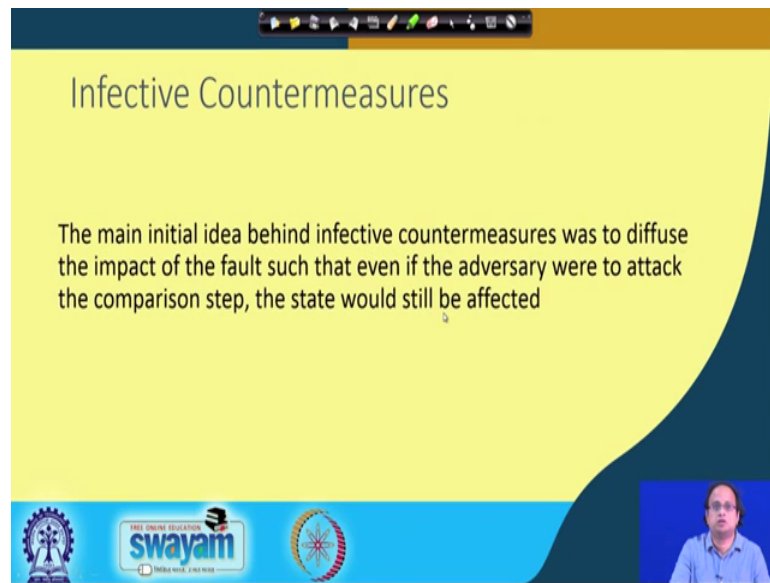
Comments on Detection Schemes

- Bias of a fault model can be quantified in terms of the variance of fault probability distribution
- Detection based countermeasures are vulnerable against biased fault attacks that are practically achievable

The slide features a yellow background with a dark blue curved shape on the right side. At the bottom, there are logos for Swamyam and other institutions, along with a small video inset of a speaker.

So, to start with I will start with the comment on detection scheme or detection based schemes, which we already seen in the previous classes. So, the bias of a fault model as we have seen right can be quantified in terms of the variance of the fault probability distribution, that is where we stopped. And therefore, right we can actually control our fault induction in a manner. So, that many of the detection based countermeasures can fail and therefore, they are vulnerable against practical attacks.

(Refer Slide Time: 01:41)



Infective Countermeasures

The main initial idea behind infective countermeasures was to diffuse the impact of the fault such that even if the adversary were to attack the comparison step, the state would still be affected

swayam
INDIA WISE, LEAD WISE

So, therefore, right we come into a new style of attack or a new style of countermeasure which is called as infective countermeasures. So, the main idea behind infective countermeasures is to defuse the impact of the fault such that even if the adversary were to attack the comparison step, the state would still be affected. And therefore, right I mean the state would still be kind of uncorrelated with the actual secret key ok.

So, therefore, there is an inherent randomization which is kind of implicit in this kind of countermeasures, which tries to kind of make the output differential unusable to an attacker.

(Refer Slide Time: 02:19)

The slide features a yellow background with a dark blue header and footer. At the top, there is a navigation bar with various icons. The main content area contains the title 'Differences with Detection Based Countermeasures' and three bullet points. Below the bullet points is a light pink box with text. The footer contains logos for 'swayam' and other educational institutions.

Differences with Detection Based Countermeasures

- Presence of random dummy rounds in between the redundant and cipher computations
- Ability to detect a fault injection in any of the three types of rounds
- No explicit comparison step hence an adversary cannot attack the comparator itself without infecting the cipher state

Avoid an explicit detection step by carefully embedding it in the infection process
The effect of the fault is diffused to render the ciphertext non-exploitable

swayam
FREE ONLINE EDUCATION
INDIA WISE, LEAD WISE

So, therefore, there are some interesting or important differences with detection based countermeasures. The first is you know like we have already seen that in the normal countermeasures like the detection based countermeasures we have the concept of an encryption, or the actual encryption and there is a redundant computation that you are doing.

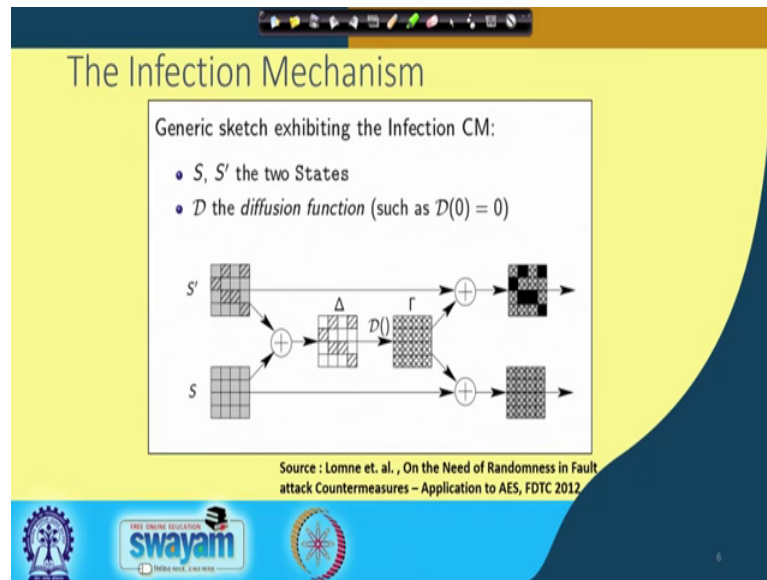
So, in this flavor of countermeasures we shall bring in a third type of round which is called as a dummy round. So, the idea would be that between the redundant and the cipher computations, there would be a random number of dummy rounds which would take place.

And the ability to detect a fault injection in any of the three rounds, which will be assumed that you will be assuming that the attacker has got the ability to detect a fault injection in any of these three types of rounds. That means the redundant round, the actual cipher computation round and also the dummy rounds.

And there is no explicit comparison step which is there and therefore, right we have seen in direction normal (Refer Time: 03:14) we have an explicit comparison step where we kind of compare either you are doing whichever type of type of detection. So, you are doing, but in this class of counter measures there are no explicit comparison steps. And hence that adversary cannot attack the comparator itself without infecting the cipher state.

So, therefore, the key point is that it avoids an explicit detection step by carefully embedding it in the infection process. And the effect to the fault is diffused in the cipher text to render it unexploitable to an adversary.

(Refer Slide Time: 03:48)



So, now we shall be kind of trying to see you know like one proposal which was published in a prior paper in every FDTC in 2012, which basically gave a nice proposal about how an infective mechanism can be built. So, the idea is that you so, as we have seen that you have got the state S and you have got the state S dash these are the two states which are there.

So, the proposal basically kind of emphasized that for a good infected countermeasure or a sound or a secured infective countermeasure, it should have some kind of inherent randomization in it. The idea would be that suppose there is a fault and as you can see that S dash is probably a corrupted state.

And therefore, the differential between S and S dash is kind of unfair is the infection. So, this is infection which is essentially obtained and this infection has to be diffused into the states. So, therefore, for this they use a diffusion function the idea is that if this XORed essentially gives you a delta which is equal to 0, which means that there is no infection. Then this function does not produce any nonzero output. And therefore, write the output does not get diffused. On the other hand when there is a disturbance; that means, when

there is a fault or there is a kind of aberration that we are doing to normal execution, then this function basically picks up that there is a disturbance created.

And now, it basically creates a random output and this random output is basically diffused into the state. And if you do that then the proposal kind of essentially ensures that or tries to ensure, that the output is not exploitable. And the output is not exploitable or the output differential is not exploitable. As we have seen right when we talked about differential fault attacks, they essentially lie upon the fact, that the one can or the adversary can compute the output differential and from there basically tries to extract out the key.

So, now, the objective here is to make the output differential randomized or essentially uncorrelated to the secret key. However, we need concrete constructions right to realize. This is just a hypothetical kind of you know scenario where, basically which gives us the idea about how to build infective countermeasures and that seems to be not very trivial.

(Refer Slide Time: 05:57)

A Proposed Infective Countermeasure

Input: P, k^j for $j \in \{1, \dots, n\}$, (β, k^0) , $(n = 11)$ for AES-128

Output: $C = \text{BlockCipher}(P, K)$

- 1: State $R_0 \leftarrow P$, Redundant state $R_1 \leftarrow P$, Dummy state $R_2 \leftarrow \beta$
- 2: $C_0 \leftarrow 0, C_1 \leftarrow 0, C_2 \leftarrow \beta, i \leftarrow 1$
- 3: **while** $\text{do} \leq 2n \text{ do}$
- 4: $\lambda \leftarrow \text{RandomBit}()$ // $\lambda = 0$ implies a dummy round
- 5: $i \leftarrow (i \wedge \lambda) \oplus 2(\sim \lambda)$
- 6: $\tau \leftarrow \lambda \cdot \lceil i/2 \rceil$ // τ is actual round counter, 0 for dummy
- 7: $R_\tau \leftarrow \text{RoundFunction}(R_\tau, k^\tau)$
- 8: $C_\tau \leftarrow R_\tau \oplus C_2 \oplus \beta$ // infect C_τ to propagate a fault
- 9: $\varepsilon \leftarrow \lambda(\sim(i \wedge 1)) \cdot \text{SNLF}(C_0 \oplus C_1)$ // check if i is even
- 10: $R_2 \leftarrow R_2 \oplus \varepsilon$
- 11: $R_0 \leftarrow R_0 \oplus \varepsilon$
- 12: $i \leftarrow i + \lambda$
- 13: **end while**
- 14: $R_0 \leftarrow R_0 \oplus \text{RoundFunction}(R_2, k^0) \oplus \beta$
- 15: **return** R_0

Source: Sikhar Patranabis and Debdeep Mukhopadhyay (Eds.), Fault Tolerant Architectures for Cryptography and Hardware Security, Springer.

$i = 1, \dots, 2n$
 $\lambda = 0 \Rightarrow \text{Dummy} \Rightarrow \kappa = 2, \tau = 0.$
 $\lambda = 1 \Rightarrow \kappa = \text{lsb}(i), \tau = \lceil i/2 \rceil$

Red. Cipher Red. Cipher
 Dummy

So, one of the proposals when one of which is kind of a pretty popular countermeasure is kind of sketched in this algorithm ok. And as you can see right this algorithm has got three different versions of ciphers.

So, there is one which is called as a state R_0 , which basically kind of stores the actual cipher computation. There is a cipher which is indexed by 1 then that essentially is your

redundant computation. And likewise there is a dummy computation ok. So, that now the dummy round right I will kind of elaborate what it means, it is basically is something which is similar to the AES round ok, but it does not take part in the actual encryption.

So, now, again you know like since you have got three variables like our 3 indices 0 1 and 2. So, we will be again having you know like 3 cipher text states C_0 C_1 and C_2 to index the outputs of the cipher round, the redundant round as well as your dummy round. So, then we kind of index a variable i so, this is a kind of a loop index that we use and initialize it to 1.

So, now, I will kind of you know like go through this algorithm in a step by step manner, you can actually read it from these reference as well. And the idea here is that there is a random bit generator which is basically producing a value which is λ . So, you can see that this loop index basically goes from 1 to $2n$; that means, i basically goes from 1 to $2n$. The idea is that out of this $2n$ there are n , which are redundant rounds and there are remaining n which are the actual encryption rounds.

So, now, there is specific order which is specified in this algorithm, how these operations take place and they are basically controlled by 3 variables. So, the first one is λ , which basically kind of tells you that the output is dummy round or whether it is not a dummy round ok. The idea is that if λ is equal to 0 then it implies that it is a dummy round on the other hand if λ is not 0; that means, λ is 1, because it produces a random bit, then it is either a cipher round or an actual round.

So, now, we again use another variable. So, therefore, right this is shown here that i goes from 1 to $2n$ λ is suppose 0, if λ is 0 then this implies, that it is a dummy round. So, then we use another variable and we call that as say K , the when λ is equal to 0 then K is equal to 2 ok.

And that is basically governed by this equation 6; that means, K as you can see that when λ is equal to 0 then it is i and λ XORed with 2 of naught λ ; that means, like if naught λ is 1 and λ is 0, then this part does not can come into effect, but this part comes. And therefore, write K becomes equal to 2.

So, κ is equal to 2 and then we use another variable which is essentially say τ for example and this τ right is equal to $\lambda \bmod 2$ and as λ is equal to 0, in this case your τ is to 0 so, τ or η 0. So, you can use so, basically this is equal to 0.

So, likewise like when λ is equal to 1 and as we said λ is equal to 1 means that it is not a dummy round, in that case κ is equal to the lsb of i . So, what I am doing i and 1; that means, I am extracting the lsb of i . So, basically right it would mean that you know like if I start with 1, then I mean i equal to 1, then lsb is 1.

In the next case when i becomes equal to 2 then lsb become 0 so, it basically kind of alternates between 1 0 1 0 and so on ok. And therefore, κ is equal to the lsb of i . And then we have got τ which is equal to $i \bmod 2$ ok. So, τ so, τ is in this case $i \bmod 2$, because λ is 1 and it is $i \bmod 2$, but since $i \bmod 2$ can be a fraction what we do is we kind of seal this.

So; that means, right it would mean that if i is equal to 1 then τ is equal to you know like 1 by 2 in that sealed. So, 1 by 2 if you seal it becomes 1. So, then it becomes 2 by 2; that means, it becomes 1. So, therefore, in a way right this kind of repeats in this manner that, this τ right essentially is like something like 1 1 then 2 2 and so on ok. That means, you can see that for every round like this is the first round, this is the second round and so on and finally, right it there will be like n and n ok; that means, that is why i goes to $i \bmod 2$ and when i is equal to 2^n we will have n here.

So, therefore, right we will have a kind of repetition of this value of τ or ψ what as shown here and this essentially is shown here; that means, right you will basically do a computation here. So, you will start with you know like a redundant computation, this is your redundant computation and then there will be an actual cipher computation.

So, this is again showed here. So, first we will do a redundant computation, then you will be a cipher combination, then you will do a redundant computation, then you will be doing a cipher computation.

But then between a redundant computation and between a cipher computation and between this right it may happen that λ becomes equal to 0 and when λ becomes equal to 0, then you do dummy operations ok. So, therefore, what will happen

is kind of depicted here is that you will be having you know like, these kind of operations and you can see that these are all dummy operations.

And therefore, right you do not know like what is essentially exactly happening. So, from the (Refer Time: 11:28) point of view right these are all indistinguishable. And you know like therefore, right I mean what the adversary will probably see is the large number of rounds that will be taking place, but it will not be able to kind of distinguish, whether it is a redundant round or whether it is a cipher round or whether it is a dummy round, which is under clay.

(Refer Slide Time: 11:48)

A Proposed Infective Countermeasure

Input: P, k^j for $j \in \{1, \dots, n\}, (\beta, k^0), (n = 11)$ for AES-128
Output: $C = \text{BlockCipher}(P, K)$

1: State $R_0 \leftarrow P$, Redundant state $R_1 \leftarrow P$, Dummy state $R_2 \leftarrow \beta$
 2: $C_0 \leftarrow 0, C_1 \leftarrow 0, C_2 \leftarrow \beta, i \leftarrow 1$
 3: **while** $\text{do} \leq 2n$ **do**
 4: $\lambda \leftarrow \text{RandomBit}()$ // $\lambda = 0$ implies a dummy round
 5: $\kappa \leftarrow (i \wedge \lambda) \oplus 2^{(-\lambda)}$
 6: $\xi \leftarrow \lambda \cdot \lfloor i/2 \rfloor$ // ξ is actual round counter, 0 for dummy
 7: $R_\xi \leftarrow \text{RoundFunction}(R_\xi, k^\xi)$
 8: $C_\kappa \leftarrow R_\xi \oplus C_2 \oplus \beta$ // infect C_κ to propagate a fault
 9: $\varepsilon \leftarrow$
 10: $R_2 \leftarrow$
 11: $R_0 \leftarrow$
 12: $i \leftarrow i + 1$
 13: **end while**
 14: ShiftRow, MixColumn, in 0th round, and a dummy MixColumn in the last round
 15: $R_0 \leftarrow R_1$
 16: **return** R_0

Source: Sikhar Patranabis and Debdeep Mukhopadhyay (Eds.), Fault Tolerant Architectures for Cryptography and Hardware Security, Springer.

$i = 1, \dots, 2n$
 $\lambda = 0 \Rightarrow \text{Dummy} \Rightarrow \kappa = 2, \tau = 0.$
 $\lambda = 1 \Rightarrow \kappa = \text{lsb}(i), \tau = \lfloor i/2 \rfloor$

Red. Cipher Red. Cipher
 Dummy

And one point should be kept in mind is that it may be emphasized here, that all the rounds are equivalent with respect to the side channel footprint, by introducing dummy sub byte shift row mix columns to the 0th round. Because, we know that in the 0th round it is just a key XOR. So, you will be adding these dummy operations. And then the final round does not have a mix column. So, we will be putting in a dummy mix column. And therefore, will have kind of 11 rounds ok. All of them are indistinguishable even along with the dummy operation.

So, now, we have to understand how the infection takes place and those are depicted here from equation 8 to 13 and so on and that is what we will kind of discuss subsequent to this. So, let me continue here.

(Refer Slide Time: 12:31)

A Proposed Infective Countermeasure

Cipher and Redundant Rounds

Redundant
Cipher

SNLF

SNLF (Some Non-Linear Boolean Function) is a Boolean Function.
It operates on a byte and maps to a non-zero value, except $SNLF(0) = 0$. Candidate function is inverse in $GF(2^8)$

swamyam
FREE ONLINE EDUCATION
INDIA WISE, LEAD WISE

So, therefore, right I mean how does the infection take place? In order to understand that, I will kind of depict this with the diagram. So, you say as I say that there is a redundant operation there is a cipher operation. So, what we do is in order to understand, that whether there is a kind of you know like fault which has been introduced. We will be using a special Boolean function and that is called as SNLF or abbreviating for some non-linear and Boolean function. A very common way to implement this would be by using the X inverse operation in GF 2 power of 8.

So, that if there is a 0 input that gets mapped into a 0 output ok. So, therefore, right you have got the cipher and the redundant rounds and you basically XORed them and you feel it into an SNLF Boolean function.

(Refer Slide Time: 13:14)

A Proposed Infective Countermeasure

Cipher and Redundant Rounds

In case of a fault, SNLF poisons the cipher state!

Note this check happens only when we are computing the cipher round.

$$\epsilon \leftarrow \lambda(i \Delta 1) \cdot SNLF(C_0 \oplus C_1) \quad // \text{ check if } i \text{ is even}$$
$$R_2 \leftarrow R_2 \oplus \epsilon$$
$$R_0 \leftarrow R_0 \oplus \epsilon$$

The diagram shows a cipher round where a redundant value C_1 and a cipher value C_0 are XORed. The result is fed into an SNLF block. The output of the SNLF block is XORed with the cipher value C_0 to produce the final cipher state. The SNLF block is designed to output zero when the input is zero, and a non-zero value when the input is non-zero.

So, now this is your proposed infective countermeasure and as you can see right there are 2 types of fault infections that can take place here. So, we basically write measured this difference between the ciphered and between the redundant and the cipher by using this function. So, the idea here is that there is you know like this is a redundant computation noted in C_1 .

And this is your cipher which is essentially denoted in C_0 , the XORed between C_0 and C_1 is what is fed into your SNLF equation. And then right you see that if there is the idea here is that for SNLF right, the property is that if you get a 0 input it gives you a 0 output whereas, if you gets nonzero input then it basically gives you a non-zero output.

So, therefore, right the objective here is that if it gets a non-zero if C_0 n C_1 are not equal for example, if C_0 and C_1 are not equal, then there will be an epsilon error which is basically getting produced and that will be a result of applying this SNLF function to this difference ok. Note that this computation of epsilon will be effective when lambda is not equal to 0; that means, not when you know like when you are doing the dummy round operation.

In fact, right this will this computation will take place when you are doing the actual cipher round operation ok; that means, if you are doing the redundant operation ok, then it will not come into play because remember that I write becomes is like 1 2 3 and so on and if I is equal to 1 right. So, you can see here there is also another additional parameter

here, we just naught of i and 1; that means, right the if the lsb of i is 0 only then this part becomes effective; that means, when i becomes even only then this check is done.

So, this check is done only if i is even and remember right that i basically go something like 1 1 I mean 1 2 3 4 and so on. And this is where you know like the actual cipher computations take place. So, the check basically is done here at these alternate positions and it is not done at every iteration or every increment of i .

So, then how do you defuse this disturbance? So, you basically defuse this disturbance into these 2 registers R_2 and R_1 and note that R_2 is essentially stands for the dummy operation and R_0 stands for the actual cipher operation. So, therefore, right what will we do is that we will take this XORed difference this will gives basically give me a nonzero difference in this case, and this will basically pollute my cipher operation for example. However, when this is 0; that means, if this is 0, then this is also equal to 0 and therefore, write the cipher does not get contaminated.

So, this is the you know like idea which is basically kind of applied over here to do the infection when there is a default ok. So, therefore, in case of faults the SNLF poisons the cipher state ok.

(Refer Slide Time: 16:10)

A Proposed Infective Countermeasure

Redundant

Cipher

Dummy

β

SNLF

ϵ

$\beta \oplus \epsilon$

Dummy Rounds

```

 $\epsilon \leftarrow \lambda(-i \wedge 1) \cdot SNLF(C_0 \oplus C_1)$  // check if  $i$  is even
 $R_2 \leftarrow R_2 \oplus \epsilon$ 
 $R_0 \leftarrow R_0 \oplus \epsilon$ 

```

Thus after this, DummyRound State is not β .

And actually right so, this is a it also poisons the dummy operation as I said that there is also a line which is R_2 equal to R_2 so, R epsilon. So, therefore, the same epsilon is

basically used to kind of pollute the dummy operation ok. So, thus after this right the dummy round state is not beta. So, remember so one thing I would like to also mention here is that the dummy round is initialized 2 beta and therefore, right when there is an error. Then the dummy round right is essentially right I mean because of this extra output of the SNLF, which is that I mean the non-zero output of the SNLF.

If, the initialization here is beta that means you know like if this is initialized to beta and if you get a non-zero output here then this becomes B plus epsilon or B XORed epsilon. And therefore, the dummy down does not remain in it is beta state ok. And this is a very important point which will be kind of which we need to keep in mind to understand the future discussions.

(Refer Slide Time: 17:11)

A Proposed Infective Countermeasure

Or, the fault can be in the Dummy Round itself.
In both cases, this will further contaminate the Redundant and Cipher Rounds through:

$C_k \leftarrow R_k \oplus \beta$ // infect C_k to propagate a fault

Dummy Rounds are a RoundFunction made of dummy SubByte, ShiftRow, MixColumn operations.
It operates on a secret value β and an idempotent secret key k^0 , st:

$RoundFunction(\beta, k^0) = \beta$

swayam

So, let us see you know like what is the consequence of these infections and actually right I mean there is also another way the infection will happen for example, the fault can be also in the dummy round.

So, we saw here that if the fault is in the cipher round or in the redundant round in the previous case what we saw is that there is an epsilon which is generated, which is polluting your cipher or it is polluting your dummy operation. It can happen also happen that a fault can be in the dummy operation and if that happens ok. So, what will happen is that the redundant and the cipher rounds throughout right or following this will also get you know like corrupted.

So, when you know in consequence of whatever happens like, in the previous case as we have seen that the fault was in the cipher or in the redundant round. Because of which the dummy round got corrupted or the fault can also be in the dummy round itself. And in both the cases we will see right that this difference will kind of propagate forward ok. And we will propagate the and contaminate the future redundant and cipher round computations.

So, here it is kind of a depiction of that. So, if you see right you see the algorithm right you will see that there is a line, which basically does this operation. So, this operation is nothing, but it shows that C_k is equal to R_k XORed with C_{k-2} XORed with beta. So, remember that this Kappa right can be either 0 1 or 2 ok. So, in case when the dummy round is not corrupted, then this C_{k-2} contain right or this content of this C_{k-2} is beta. And therefore, what happens is that this beta gets cancelled with this beta. And therefore, C_k is kind of you know like R_k is basically assigned to C_k ok.

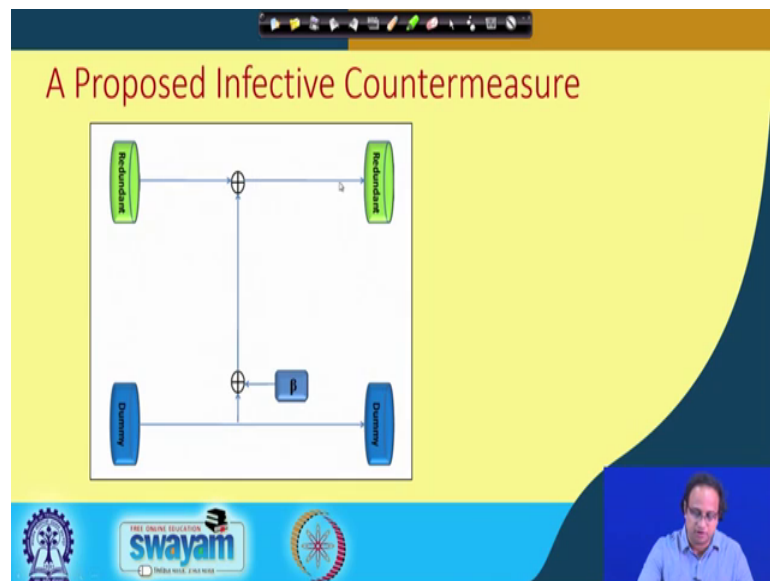
So, therefore, right I mean in this case there is no problem, on the other hand right when this C_{k-2} is not equal to beta then write this nonzero value here will kind of corrupt the corresponding round; that means, whether it is a cipher round or if it is a redundant round right in both cases there will be a contamination. So, therefore, right this kind of brings us to the question right that what is the definition of dummy round ok.

So, therefore, dummy round basically right as I said that should be indistinguishable from the cipher round and from the redundant round. So, the attacker right by observing a side channel cannot understand that this is a dummy round getting into execution ok. And therefore, right it will consist of the same components as that we have in AES operation; that means, they will have a sub byte, shift row and mix columns or mix column operations.

And what we will do here is that we will take an input which is a secret input beta, we will take a secret key which is k_0 . And such that this k_0 is idempotent, which means like if I take beta as an input this dummy operation also produces beta as output. So, that if there is no fault then the beta remains as beta ok. And therefore, right it never contaminates any future operation, because this C_{k-2} is beta, then beta always gets cancelled with beta and there is no future you know like contamination.

However, if there is a fault then this we as we discussed right that this you know like dummy round output can get corrupted and if it gets corrupted then this C_2 will not be equal to β . And therefore, C_k will also be not equal to R_k , but will get infected with an accompanying fault. So, let us see you know like how this works you know like in more details.

(Refer Slide Time: 20:33)



So, basically right what will happen over here is that shown over here, that if this is β and this is dummy is also β then I say that β gets cancelled with the β . So, the redundant round does not get corrupted. However, if there is a non-zero value then it kind of pollutes the redundant now in computation.

(Refer Slide Time: 20:45)

A Proposed Infective Countermeasure

One final point in this description is that there is a compulsory dummy round at the end:

$$R_0 \leftarrow R_0 \oplus \text{RoundFunction}(R_2, k^0) \oplus \beta$$

And likewise right so, this is how again you know like you also have the cipher round, also kind of getting contaminated. If, there is a non-zero value as a result of this dummy round getting either faulted or getting corrupted, because of you know like a previous infection in a cipher or redundant round 1.

Final point in the description is that there is after you have done your you know like all your operations; that means, you have come to your final cipher round; that means, when your i is equal to $2n$ in your execution. You come to a compulsory dummy round ok. In this compulsory dummy round you do this operation; that means, you have got R_0 which is basically storing your actual cipher output, you XORed it with the round function R_2 comma k_0 XORed with beta.

You remember that if there is no fault then R_2 has beta here. And therefore, again this results in beta and therefore, beta gets canceled with beta and therefore, R_0 remains is R_0 without any problem. However, out; however, if there is you know like a fault in R_2 and because of that you know like there has been a previous fault in the cipher or in the redundant round.

Because, because of which as we know that R_2 is not equal to beta and therefore, right this will result in something different kind of random. And that will basically get XORed with beta will produce another random and that will lead to another level of infection that will happen in the cipher operation.

(Refer Slide Time: 22:09)

FDTC 2013 Attack

- Fault f in first byte of the second row in the input of 10th cipher round of AES128
- Countermeasure infects the faulty computation twice
 - After the execution of 10th cipher round
 - After the execution of compulsory dummy round

So, therefore, right this is kind of the overall description, but unfortunately right this particular countermeasure can be attacked. So, the first attack that I will be discussing here is essentially due to a work which is published in FDTC in 2013, which showed you know like that infective countermeasures are more difficult to construct than believed ok.

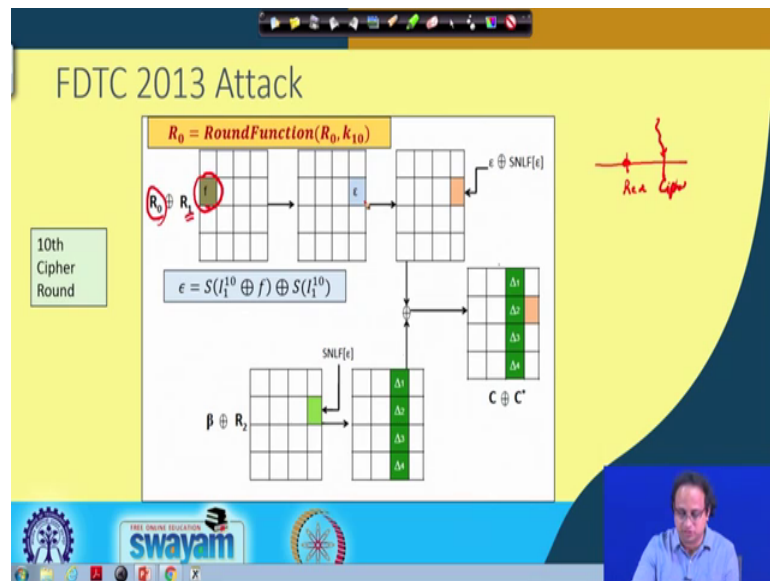
So, for example, right the fault f . So, we will basically assume the fault f is in the first byte of the second row in the input of the 10th cipher round of AES 128. And therefore, right if you remember that in the normal execution we always have you know like when i becomes equal to $2n$.

So, we finally, end with the cipher round because previous to that there has been somewhere where we have done the redundant round ok. And it may happen that within this write there are some number of dummy rounds ok. So, I do not know that whether you know like how many dummy rounds has actually taken place.

So, here we are assuming that the fault f is in the first byte of a second or the input of this 10th round cipher; that means, before the final computation has been done. So, therefore, the counter measured with infect or we will result you will infect the faulty computation twice ok. The first time will be it will execute is after the execution of the 10 cipher round and the second time right because there is a compulsory dummy round so, that also will kind of lead to the infection.

So, there will be 2 times that will have an infection.

(Refer Slide Time: 23:35)



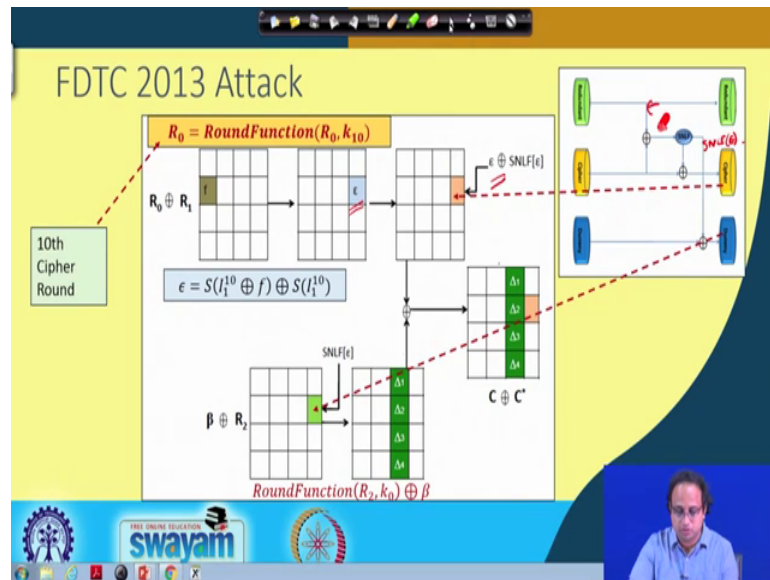
So, let us see you know like how that works and here is a depiction of the how that works. So, as you can see that the fault is in the 10th cipher round. So, R_0 is equal to round function R_0 comma k_{10} and here is the fault. The fault has been shown over here as this is the first byte of the second row and that has been faulted by value and so, showed here as f .

And because of the shift row right essentially this will get shifted this error will come here. So, this epsilon is nothing, but you know like because suppose the original state here was I_{110} ok. And in the actual computation right this will be I_{110} and S applied on that and in the faulted computation right, this I_{110} will be XORed with f and then will be doing an S box.

Remember that the redundant computation in the 10th round has already taken place, the redundant round is done we are basically doing the actual cipher round and when you are doing this there is a fault ok. Because of this fault and is the 10th round cipher this computation right if I say that this is my reference point R_1 and I am basically calculating R_0 and there is a fault.

So, you basically see that there is a fault over here and because of this fault right you will basically have an infection. So, let us see you know like how the infection will basically look like.

(Refer Slide Time: 24:51)



So, therefore, right you will have you know like for example, what will happen is if you remember this diagram. In this diagram right there is a cipher which is being computed and this cipher is nothing, but you know like the XORed of you know like the output of the SNLF function. So, the as we know that there is an error which is being generated so, therefore, the so, if we get epsilon over here like if we get epsilon over here.

So, if we get epsilon over here, then this will basically generate an SNLF epsilon. And this SNLF epsilon will basically get XORed remember that already you have this infection epsilon and that will get augmented with SNLF epsilon. And so, you will have epsilon XORed with a SNLF epsilon ok. So, also remember that there is this dummy round you know like over here and because of this dummy round there will be a further effect on the dummy round operation.

So, this initially right this dummy round at this dummy round has beta. So, if I take an XORed of beta with R 2. So, this right was kind of generating 0 over here I mean beta XORed with R 2 was 0, but because of this infection SNLF epsilon this will come over here and therefore, right it will affect this particular byte as well.

Now, remember that this is this round function right is just normal and like a normal AES encryption or round. And therefore, there will be a shift row operation and because of which right this will get shifted and it will kind of affect this column ok, shown here as delta 1, delta 2, delta 3 and delta 4.

So, now, if you take an XORed between these two things to understand what is the final differential which you obtain, you will see that this in this kind of gets shifted over here, this part this error which is epsilon XOR SNLF epsilon gets exposed to the attacker. So, the attacker is able to see this diffusion even now ok.

And therefore, right the attacker can start guessing the key and from there we will try to kind of reduce and obtain the actual key value. So, this exactly is basically the idea behind the attack. And we can see you know like the more details of the attack as follows.

(Refer Slide Time: 26:54)

FDTC 2013 Attack : Final Difference

• Infection caused by compulsory dummy round does not affect ϵ .

$$C \oplus C^* = \begin{pmatrix} 0 & 0 & \Delta_1 & 0 \\ 0 & 0 & \Delta_2 & \epsilon \oplus SNLF[\epsilon] \\ 0 & 0 & \Delta_3 & 0 \\ 0 & 0 & \Delta_4 & 0 \end{pmatrix}$$

• Infection SNLF[ϵ] caused by 10th cipher round is ineffective.

• Attacker uses the value of $\epsilon = S(i_1^0 \oplus f) \oplus S(i_1^0)$ to make hypotheses on i_1^0 and key byte k_{13}^{11} .

• Repeat this process with two more pairs of faulty and correct ciphertexts, using constant byte fault model.

• The attack targets **last three rows** of the 10th round input.

• Recover remaining 4 bytes of top row using brute force search.

Last Round Key: k^{11}

k0	k4	k8	k12
k1	k5	k9	k13
k2	k6	k10	k14
k3	k7	k11	k15

So, what therefore, what we will do is basically write this will infection will kind of result in a differential like this. And as we know the corresponding key right to this is shown over here in the last round key and shown in red that is this is your k 13 for example.

So, therefore, now if I basically you know like know this differential. So, this differential means I will know the value of epsilon XOR, SNLF epsilon remember that this is SNLF

is something like an x inverse function. And therefore, if I know this right I will get some possible values of epsilon.

So, therefore, right what I will try to kind of do from here is that I will basically attacker you know like what I will basically do is I will use the value of epsilon remember that epsilon is nothing, but the error and that is basically as we have seen is equal to S of I_{10} XORed with f XORed with I_{10} ok. So, this is where I wrote here epsilon is equal to S of I_{10} XORed with f XORed with S of I_{10} .

So, if you know this right, then you basically what you will try to do is you will try to make hypotheses on I_{10} ok. And from there you will basically try to gather the value of or obtain the value of $k_{13,11}$ is this key. So, therefore, you will of course, you will not get a unique value. So, therefore, you have to repeat this process with two more pairs of faulty and correct cipher text, but remember in this case that the fault model is constant, because if the fault model is not constant then this f will give you multiple values and there could be like 256 possible values for this f and therefore, right you will not get any reduction of entropy.

So, therefore, right this is not linked to the reduction or revelation of the key. So, therefore, what we will be trying to do is we basically try to keep this fault constant and from there we will try to come up with a solution, which will give me values for I_{10} and hence will leak me information about $k_{13,11}$. So, one of the restrictions behind this attack right is that this attack is kind of you know like restricted to the last three rows of the 10th round and it will not work if the fault is there in the first row for example.

(Refer Slide Time: 28:00)

Flaws Exploited by FDTC 2013 attack

- The last cipher round is always the penultimate round: The attacker can verify target round using side channel.
- A fault in last three rows of 10th round \implies Infection caused by compulsory dummy round does not affect the erroneous byte.

Remark
What happens if the infection caused by compulsory dummy round affects the erroneous byte of 10th round??

swayam

So, why will need not work can be easily observed here and this is the question you know like if the infection is in the first round, then what will happen.

(Refer Slide Time: 29:08)

Extending FDTC 2013 Attack to the Top Row

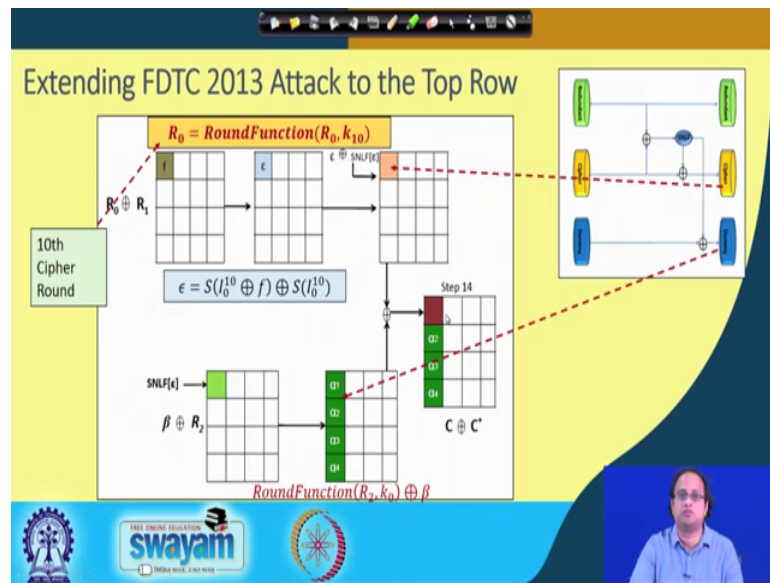
- Fault f in I_0^{10} , i.e., first byte of the top row in the input of 10th cipher round
- Countermeasure infects the faulty computation twice
 - ▶ After the execution of 10th cipher round
 - ▶ After the execution of compulsory dummy round

Harshal Tupsumudre, Shikha Bisht, Debdeep Mukhopadhyay:
Destroying Fault Invariant with Randomization - A Countermeasure for AES Against Differential Fault Attacks. CHES 2014: 93-111

swayam

So, why will it not work? So, this will not work and this is kind of elaborate in this work in just in 2014.

(Refer Slide Time: 29:16)



And that essentially can easily be observed if we again see the depiction of what happens in the fault is over here. So, if the fault is over here like previously we have an infection and this will propagate and we will get. So, this fault is again at the input of the 10th round, 10th cipher round.

So, therefore, what will happen is that this will get to be epsilon XOR is SNLF epsilon, this will you know like kind this is result of the cipher round infection and likewise there will be a dummy round encryption infection and that is shown here in this column like alpha 1, alpha 2, alpha 3 and alpha 4.

So, now if you take an XOR between these two things remember you know like this is the effect on the dummy round and this is the effect on the actual cipher round. So, if you take an XOR, because remember that there will be a final step in the equation, where there is an effect of the compulsory dummy round ok.

So, the compulsory dummy round will again pollute the actual cipher round. So, therefore, this differential will get mixed with the cipher differential ok. And this will result this shown here as this step 14 which will basically give you the final output differential ok. Now, in this differential if you observe you see right is showing your values alpha 1, alpha 2, alpha 3, alpha 4, which trivially I have got no idea about it so, then it kind of looks pretty much random.

And, if they are random you can see that this epsilon XOR epsilon which I was previously able to see quite clearly is now getting kind of mixed with this alpha 1 value. And therefore, I cannot you know like segregate it I cannot separate it out from this infection ok.

And therefore right I cannot make you know like hypothesis of the I 1 10 and which essentially led to the revelation of the corresponding key byte. So, therefore, right that that will not straightforward work and therefore, this attack at this point we can just do you know like reveal these 12 bytes, but the first round I will not be able to leak and one of the reasons why I this will not work in the first round is because of the shift row operation.

If you go back and just see you know like the state transfer happened here, this got exposed because this column right. Essentially is you know like getting shifted here, you can see that this differential here is getting shifted because of the shift rows. And therefore, right this particular thing is not affected it kind of gets exposed.

On the other hand right he does not work over here, because right here in this particular state, in this particular transformation remember that in the first round there is no shift ok. And therefore, this byte does not get shifted and therefore, this column overlaps with this column ok. And therefore, this gets hidden and therefore, it does not leak.

So, this is the basic idea behind the attack and you know like we will see remember that it still works on only 3 bytes and it does not work on the first row. So, we need to kind of do something. So, that we can kind of extend it to a solution, which works on pretty much the entire state matrix we will also see another extension where we basically kind of relax the fault model and make it work on random fault models ok. So, I will stop here and we will continue from the next class on this point ok.

Thank you.