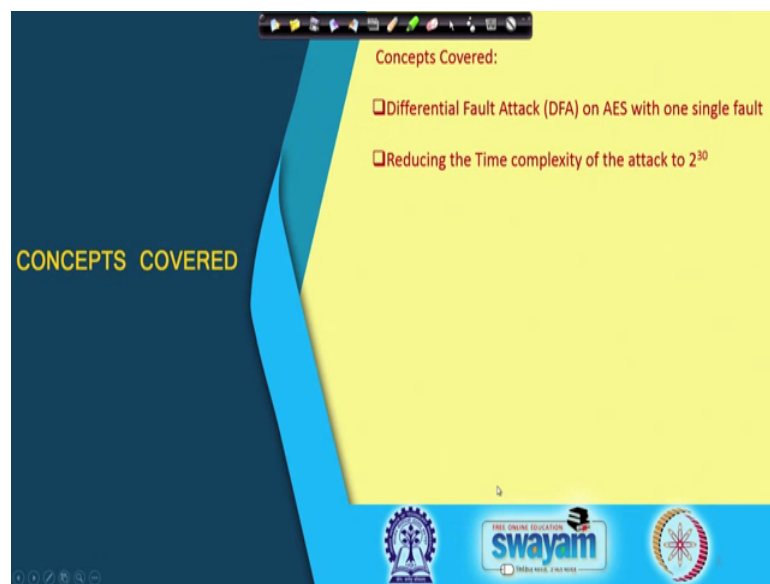


Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 47
Improved DFA of AES

So, welcome back to this class on Hardware Security. So, we shall be continuing our discussion on differential fault attacks on AES.

(Refer Slide Time: 00:25)



So, in particular we shall be trying to look into how to perform DFA attack on AES with one single faults. So, in the last class we saw that we 2 faults try 2 single byte faults, two we were you know like able to reduce the AES key size to a unique value. But now suppose you know like and we have seen that if there is one single fault, then you reduce the AES key size to 2 power of 32 values. So, the question is now that we will try to take up is that with one single fault can you reduce it further.

And in particular like we will see that in there is we can develop an algorithm through which you can do that, but it can reduce the key size to only 2 power of 8 values, but if you do that then you need a time complexity of 2 power 32. And then we will see a further improvement in which you can reduce that time complexity to 2 power of 30 values; that means, average on average like 4 times faster ok. That means, if in the

previous case you were requiring like 60 minutes to do the attack in this case you would require around 15 minutes to do the attack it is significantly faster.

(Refer Slide Time: 01:20)

Can the Attack Work with One Faulty Ciphertext?

- With one faulty cipher text:
 - Number of possible values per 4 bytes of the key is around 2^8 .
 - There are 2^{32} possible candidates for 128 bits of the AES key.
 - Brute force key is thus possible!

DebdeepMukhopadhyay, An Improved Fault Based Attack of the Advanced Encryption Standard. AFRICACRYPT 2009. LNCS 5580, pp 421-434

So, the question is that can the attack work, that we have seen with one single faulty Ciphertext we have seen already that this is a summary and even then right 2^{32} probably a brute force search is still possible, but we would try to reduce it further.

(Refer Slide Time: 01:35)

Comparison of Existing Fault Attacks

Reference	Fault Model	Fault Loc.	#Faulty CT
Blomer	Force 1 bit to 0	Chosen	128
Giraud	Switch 1 bit	Any bit of chosen bytes	50
Giraud	Disturb 1 byte	Anywhere among 4 bytes	250
Dusart	Disturb 1 byte	Anywhere between last 2 MixColumn	40
Piret	Disturb 1 byte	Anywhere between 7 th & 8 th round MixColumn	2
Mukhopadhyay	Disturb 1 byte	Anywhere between 7 th round MixColumn and last round input	2

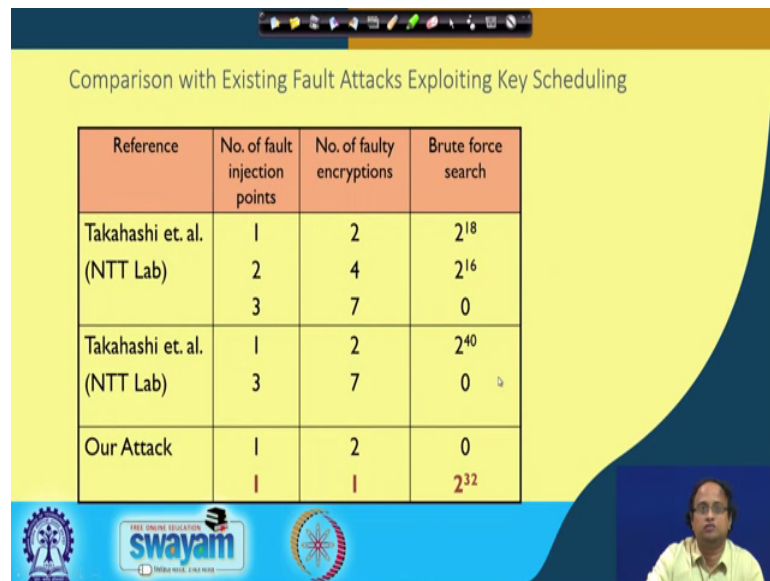
So, here is again in like some comparisons of existing fault attacks for example, there was a old work by Blomer which requires around 128 faults and then there was like 2

works from Giraud which requires around 50 and 250 faults. So, the whole pursuit right has been like again there was a paper by Dusart to which requires around 40 faults and then my period which requires around 2 faults and then right there was an attack, that we reduced the time complexity to 2 power of 32 values, but still requires around 2 faults ok. So, therefore, the pursue has been to reduce the number of faults and we would like to continue this trend and reduce it further.

(Refer Slide Time: 02:09)

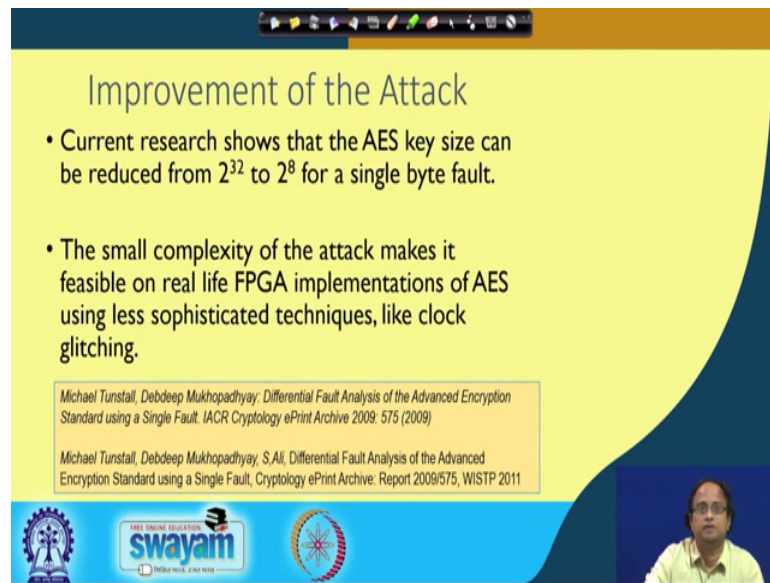
Comparison with Existing Fault Attacks Exploiting Key Scheduling

Reference	No. of fault injection points	No. of faulty encryptions	Brute force search
Takahashi et. al. (NTT Lab)	1	2	2^{18}
	2	4	2^{16}
	3	7	0
Takahashi et. al. (NTT Lab)	1	2	2^{40}
	3	7	0
Our Attack	1	2	0
	1	1	2^{32}



So, here are some more attacks that were tried by some other groups and what we are still trying to reduce it is we are still trying to optimize the fault attack.

(Refer Slide Time: 02:20)



Improvement of the Attack

- Current research shows that the AES key size can be reduced from 2^{32} to 2^8 for a single byte fault.
- The small complexity of the attack makes it feasible on real life FPGA implementations of AES using less sophisticated techniques, like clock glitching.

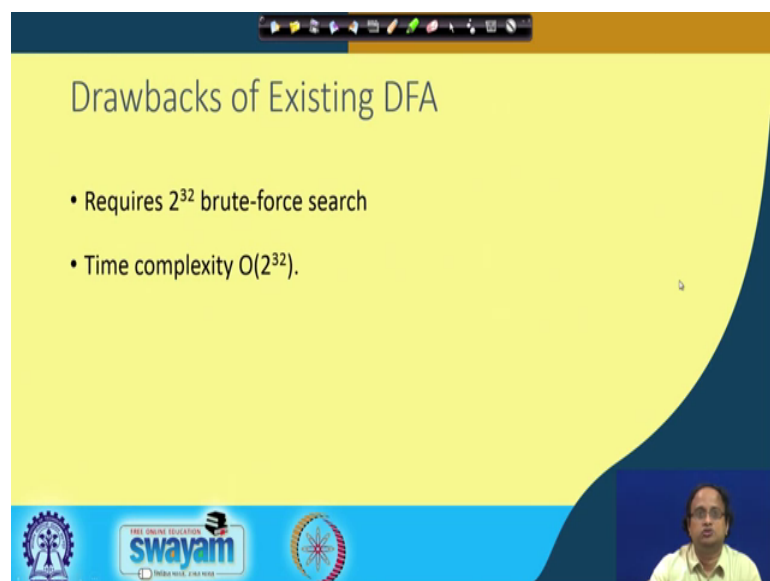
Michael Tunstall, Debdeep Mukhopadhyay, Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault. IACR Cryptology ePrint Archive 2009: 575 (2009)

Michael Tunstall, Debdeep Mukhopadhyay, S.Ali, Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault, Cryptology ePrint Archive: Report 2009/575, WISTP 2011

swayam

So, therefore, right I mean this is the current status right now there where you see that the AES key size can be reduced on to power of 32 to power of 8 values only for a single byte fault. And this essentially makes that are very practical because; that means, right you just need one single shot to reduce the key size to only 256 guesses which is essentially very easy. And so, these are the two references that I will be using and elaborating in this work.

(Refer Slide Time: 02:44)



Drawbacks of Existing DFA

- Requires 2^{32} brute-force search
- Time complexity $O(2^{32})$.

swayam

So, what are the drawbacks of the existing DFA? So, basically requires a brute force search of 2^{32} values and it has got a time complexity of 2^{32} . So, you would like reduce this space complexity if I say that the brute force search of 2^{32} , we would also write we try to reduce a time complexity further.

(Refer Slide Time: 03:03)



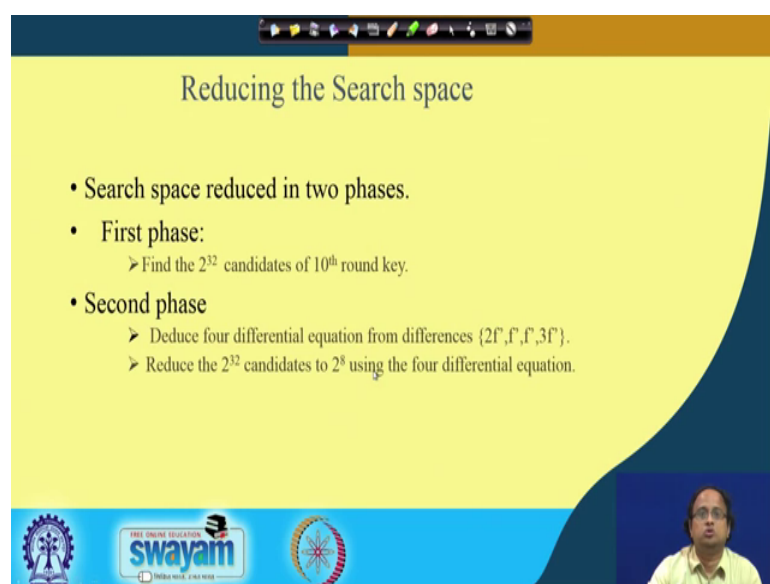
Improving The DFA

- The attack is improved in two ways:
 - Reduce the search space of the attack
 - Reduce the time complexity of the attack

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 educational institutions, along with a small video inset of a man in a light green shirt.

So, therefore, right we will try to improve both on the search complexity as well as the time complexity of the attack.

(Refer Slide Time: 03:10)



Reducing the Search space

- Search space reduced in two phases.
- First phase:
 - Find the 2^{32} candidates of 10^{th} round key.
- Second phase
 - Deduce four differential equation from differences $\{2f, f, f, 3f\}$.
 - Reduce the 2^{32} candidates to 2^8 using the four differential equation.

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 educational institutions, along with a small video inset of a man in a light green shirt.

So, how can we reduce the search space of attack? In the first phase we have seen that we require we have basically you know like we find the 2^{32} candidates of the 10th round key, but in the second case second phase we shall reduce 4 further differential equations which we had neglected in the previous attack ok. So, you remember right that state the propagation there was like in the 9th round, there was also like certain equations among the faults which we did not write we did not exploit in the attack we just considered about the final equation sets which we explore it in the attack.

So, therefore, right they also had relationships like 2^f dash f dash f dash and 3^f dash only right if you want to go and exploit those differentials, then you have to work behind the 9th round and go there; that means, you have to work behind the 9th round key. So, therefore, right you have to understand the relationship between the 9th round key and the 10th round and that can be slightly complicated. And then finally, right you would like to reduce the 2^{32} candidates right.

Once you have these equations further to 2^8 values ok. Remember that if you can do this, then this filter also will have a probability of 2^{-24} ok; that means, these 2^{32} candidates will get charmed will get filtered and get reduced to 2^8 values because it is like 2^{32} multiplied with another factor of 2^{-24} which will give on average around to power of 8 values ok.

(Refer Slide Time: 04:35)

Reducing the search space

• Find 2^{32} candidates K_{10}

2^{32} Differential Equation

$$2F_1 = S^{-1}(x_0 \oplus k_0) \oplus S^{-1}(x'_0 \oplus k_0)$$

$$F_1 = S^{-1}(x_{13} \oplus k_{13}) \oplus S^{-1}(x'_{13} \oplus k_{13})$$

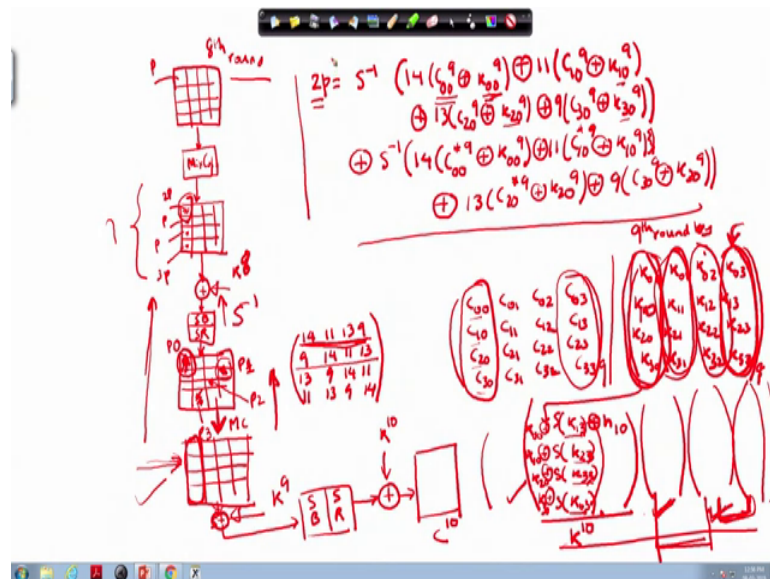
$$F_1 = S^{-1}(x_{10} \oplus k_{10}) \oplus S^{-1}(x'_{10} \oplus k_{10})$$

$$3F_1 = S^{-1}(x_7 \oplus k_7) \oplus S^{-1}(x'_7 \oplus k_7)$$

swamyam

So, therefore, right in order to understand that, we have to basically look into the equations. So, therefore, here you know like we have been exploiting these equations and therefore, we have reduced 2 power of 128 key size to 2 power of 32 values, what we now want is we want to work behind and go to these positions ok. We want by exploit the relationships here among these key wats and reduce them further. So, in order to do that let us try to you know like derive or you know we just redraw this picture and try to see how the equations work.

(Refer Slide Time: 05:08)



So, in particular right what I am trying to say is that, imagine that you have this is the fault that you had already you know like induced and remember that this fault essentially suppose has a value of p. So, I do not really bother about what is the value of the fault, the fault can take any random value and then you have the Mix Columns. So, let me write this as the mix columns just ok. So, mix columns and then because of the mix columns you have the differential essentially as follows. So, therefore, let us just write the differentials here. So, differentials are 2 p p p and 3 p. And then you have the XOR of the 9th round key. So, this is your k 9 which comes into play and then you have got the sub byte.

So, I am just writing them as SB and shift row. Remember the 9th row 9th column 9th round has shift rows ok. So, you have the shift rows here and then you have because of that right these particular bytes, get spread out because of the shift row. So, therefore,

how does it get spread out? We know that the shift rows does you know like the left shift for example, in the first row there is no shift in the second row this gets shifted like this, then it gets shifted here and then this gets shifted here. So, you have got the values like p_1 p_2 and this value is you know like suppose or maybe just write it as p_0 and then I write it as p_2 and this is p_3 ok.

So, these are the 4 bytes that you have or 4 differential bytes that we have and then you follow this up by a mix columns because of the mix columns we have seen that you get those nice relationships that we have just previously seen. So, I am just not elaborating them, but just the fact that after this you have got again you know like you have got again you know like ok. So, here this was actually the 8th round key because this is in the 8th round.

So, this is this fault has been induced at the input of the 8th round and then what you do is, you because of here you get these relationships and then you XOR this with the 9th round key which is a k_9 and after this you have got again a layer of 2 operations which is the sub bytes and the shift rows and finally, you do an XORed with the 10th round key which is k_{10} to give your corresponding cipher. So, this is your cipher text. So, let us denote it as C_{10} this is how your differential works.

So, now, if you observe that we basically have been have exploited, you know we have targeted these differentials this has already been targeted and therefore, you have got reduction now we want to target these differentials. So, the question is how do I exploit these differentials? So, in order to work out that right we will basically right we see that we if you want to work out that, then we basically need to work behind ok. So, remember that in this particular step right I mean when you are working behind, you have basically exploited these differentials here, but you work behind and you want to go to these values p_0 p_1 p_2 p_3 ok.

For this you have to work behind this mix column. So, there is a mix column we just taken place here so; that means, if you want to work behind the mix column, you have to con you have to see then inverse mix column matrix ok. So, what is the inverse mix column matrix? So, the inverse mix column matrix is shown here as 14 11 13 and 9. So, you just can go back and check. So, it is 14 11 13 and 9 is a circular shifts 14 11 13 and 9 and likewise 14 11 13 and 9. So, this is your inverse mix column matrix ok.

So, now, you observe that if I want to you know like get this particular differential, if I want to get these position or these differential then I have to you know like take this row of the mixed converse mixed column matrix and I have to multiply with this differential here or these this column here ok. So, therefore, right we just write down the columns in this way. So, let us, so, this is your the output of the you know like this is the point where you are getting the output of the 9th round cipher ok.

So, the 9th round cipher you can write as so, let us just write it down C 0001 02 C 03 then you have got C 10 C 11 C 12 C 13 C 20 C 21 C 22 C 23 C 30 C 31 C 32 and C 33 this is your 9th round cipher. So, if you want to come to this location right you are basically multiplying this with this column ok. So, you are multiplying for both the correct value and for the faulty ciphertext faulty operation and then you are taking XORed to get the value of this p 0 differential ok. Likewise right if you want for this p 1 you have to multiply you know like suitably this row with this column ok. So, likewise right you have to basically kind of multiply for the respective columns.

So, therefore, right I mean. So, therefore, right if you want to do that, then you need to you know like and if you do that right then you these are the equations that you get and let us you know like quickly write down those equations. So, in particular right I mean you will get the equations as say. So, let me just first of all. So, with this right we can work out these equations and so, the equations right which you will get is say 2 p. So, because you know like these 2 p this p p these p and this is 3 p, but you get that when you work behind ok.

So, remember when you get to this differential, after that you can work behind because you know like you have to just you know like already right I have taken care of the shift rows, but when you are working behind you have to work behind the a inverse S box because there is an S box here ok. So, you work behind this and that is denoted as S inverse and the k does not create a problem because it is a differential.

So, therefore, if I take the differential then the effect of k it gets cancelled out ok. So, the kth round does not create a problem. So, therefore, right if you work behind then you get S inverse and since you are you know like working behind and I am saying that you have to take the effect of these 14 11 13 and 9. So, you can write this pretty much as 14 multiplied with C 009 XORed with k 009.

So, again remember that when I am writing k_{009} means I am considering the 9th round key and the 9th round key of the 9th round key is essentially nothing, but k_{00} k_{01} k_{02} k_{03} and exactly in the same way k_{10} k_{11} k_{12} k_{13} k_{20} k_{21} k_{22} k_{23} k_{30} k_{31} k_{32} and k_{33} this is your 9th round key ok. And this essentially right we will see how this line down propagates to become the 10th round key ok, but right now I am assuming that I am just considering the 9th down cipher and the corresponding 9th round key and therefore, I can XOR them and come to this position. So, therefore, right I just take and I take 14 11 and I apply on this. So, therefore, these like if you just continue like this. So, you get 11 C 109 and that is XORed with k_{109} and that is XORed with 13 C 20 9 XORed with k_{209} XORed with 9 times C 30 9 XORed with k_{309} .

So, here you see that we have considered C 00 C 10 C 20 and C 30. So, its pretty much you know like this column we have multiplied with this row all the inverse mix color matrix and this is for the correct operation right this is all for these are all for the correct operation if I want to get the differential, then I have to XOR this with the faulty case ok. And in the faulty case essentially can be denoted as exactly same the only thing that we will do is we will write C 00 star 9 to indicate that this is the faulty cipher ok. So, you get k_{009} XORed with 11 multiplied with C 10 9, but again there is a star XORed with k_{109} XORed with sorry XORed with 13 C 20 star 9 XORed with k_{209} XORed with 9 times C 30 9 XORed with k_{309} ok.

So, this is your you know at this point you have come to 2 p. So, you have come to basically you have got this differential and there from this differential you. So, basically you have to come this location taking the S inverse and you have to you have got this differential now you have come to 2 p here. But remember all these key equations have now involved with k_{009} and these are the 9th round key which I have to kind of express using the 10th round key.

So, for this right we will just take a look into how the key shielding works and if you see the key scheduling, the key scheduling of AES is very simple. So, what you do is you just consider them column by column for example, I consider this column and I consider this column and then I basically I know I do an operation on this column. So, this column is essentially you know like rotate the operation which is called as a road toward and then that is followed by an S box operation.

So that means, what happens is this k_{13} goes at the top this k_{13} comes at the top then we have got k_{23} then we have got k_{33} and then this k_{03} comes in and then we apply an S box operation on them. So, this is essentially called often a sub word and then there is a round constant. So, in this case the round constant you say the 10th round constant you say denoted as h_{10} and you get the final you know you get this particular operation ok. So, this is your you know like this is your first round of the 10th round key.

And then you basically you know like what you do is once you have got that you XORed this with the first round. So, you basically the first column. So, you basically XOR this with k_{00} and you XORed this with the second value we XORed with k_{10} then you XORed this with k_{20} and then you XORed this with k_{30} and that gives you the first column of your 10th round key ok.

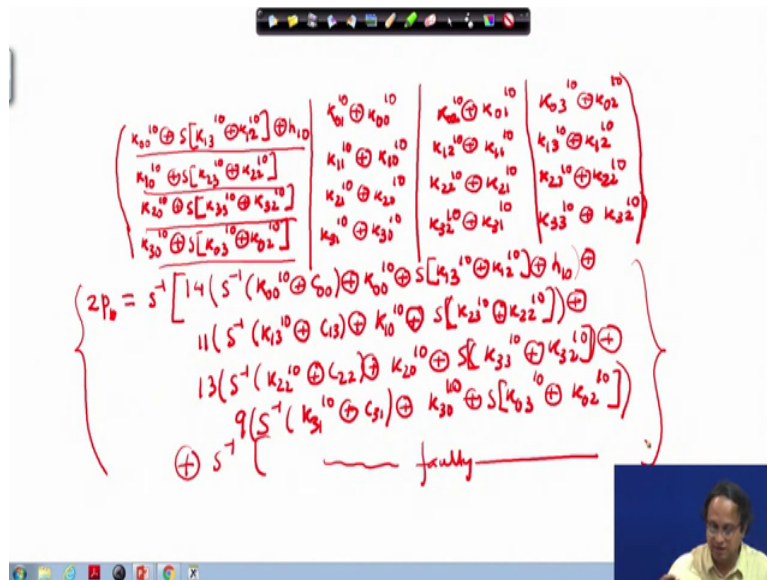
So, this is your first column of 10th round key, for the second column what do you do is you know like again you take this column and now. So, you basically take this column and you XORed it with you know like the with something 4 round 4 steps behind and. So, XORed it with this column and then you get this column ok. And then for the third column of the 10th round key you basically XORed with the previous column with this column and once the final column you basically XORed it with this column you XORed the last column of the 9th round key.

So that means, if you now want suppose I give you the 10th round key, because remember that you have reduced the key size of a key space AES to some possible values. So, suppose you know the 10th round key. So, if you want to express the 9th round key; that means, you have to just take an XORed of these 2 successive columns. So, if you take an XORed of these 2 successive columns you get this color ok.

Likewise, if you take an XOR or exclusively or these 2 columns I mean you take an XORed of these 2 columns you get the previous column. If you take an XORed of these 2 columns you can get this column. So, you basically can express that 3 columns of 9th round key just by XORing successive columns of the 10th round key. Only for the final like that the first column of the 9th round key you have to just do a little bit of management into see that there is a cyclic rotation that we have done ok. And that for that you have to see you know like the values of k_{13} k_{23} and k_{33} and k_{03} of the 9 round key.

But remember that you know like k_{13} , k_{23} and k_{33} and k_{03} are something that you know like you have already observed you have already you know like got in the you know you have already obtained them, you have basically already obtained them and express them using the 10th round key.

(Refer Slide Time: 18:43)



So, therefore, you summarized this entire story then you know like you can write the you know like corresponding 9th round the 9th round key in this way. So, you can write this as $k_{00} \oplus 10$ XORed with S times $k_{13} \oplus 10$ XORed with $k_{12} \oplus 10$ XOR with h_{10} and then the second one is $k_{10} \oplus 10$ XORed with S times $k_{23} \oplus 10$ XORed with $k_{22} \oplus 10$, $k_{20} \oplus 10$ XORed with S times as $k_{33} \oplus 10$ XORed with $k_{32} \oplus 10$ and then you have got $k_{30} \oplus 10$ XORed with S times $k_{03} \oplus 10$ XORed with $k_{02} \oplus 10$.

So, this is the first column of the 9th round key, but for the others right it is very straightforward because we just do this $k_{01} \oplus 10$ XORed $k_{01} \oplus 10$ XORed with $k_{00} \oplus 10$ just specifically XORing the successive columns $k_{11} \oplus 10$ XORed with $k_{10} \oplus 10$ and then $k_{21} \oplus 10$ XORed with a $k_{20} \oplus 10$ $k_{31} \oplus 10$ XORed with $k_{30} \oplus 10$ and then $k_{02} \oplus 10$ XORed with $k_{01} \oplus 10$ then $k_{12} \oplus 10$ XORed with $k_{11} \oplus 10$ XORed with $k_{21} \oplus 10$ and then $k_{32} \oplus 10$ XORed with $k_{31} \oplus 10$ and then the final column is again in a similar fashion $k_{03} \oplus 10$ XORed with $k_{02} \oplus 10$ and then you have got for the second row (Refer Time: 20:53) we are just doing is $k_{13} \oplus 10$ XORed with $k_{12} \oplus 10$ and then $k_{23} \oplus 10$ XORed with $k_{22} \oplus 10$ and then you have got $k_{33} \oplus 10$ XORed with $k_{32} \oplus 10$.

So, this is your final 9th round key expressed using the 10th round key bytes. So, now, let us you know just write I just write one equation similarly you can derive or we can write the other equations. For example, we have already seen that $2 p_0$ or you know like we have seen that $2 p$ rather let us write $2 p$ is equal to S inverse of. So, you know like if we just go back for example and just see the previous thing. So, in the previous case right we have already expressed $2 p$.

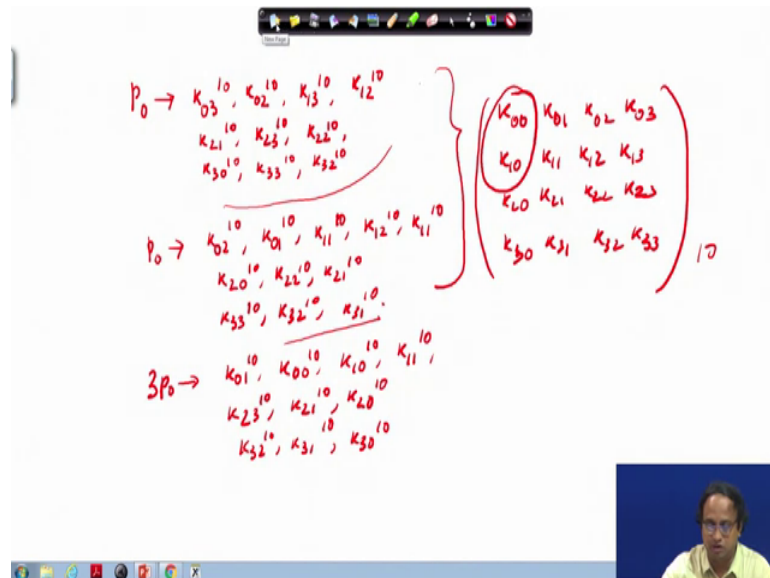
But now what I will do means I will you know like express the C_{09} in terms of the final round cipher and I will also express the k_{09} in terms of the final round key that is the 10 round key ok. So, I will kind of write both of these things in and similarly for the other terms also and complete this equation. So, if you do this right then let us see how we can do that. So, therefore, right you will basically get this that is S inverse of remember like we are 14 multiplied with S inverse of $k_{00,10}$ because this is the final round key XORed with C_{00} XORed with the 9th round key ok. So, the 9th round key as you can see is here ok. So, is written here as is written here or expressed using the 10th round values.

So, if we just plug in over there you get $k_{00,10}$ XORed with S times $k_{13,10}$ XORed with $k_{12,10}$ and XORed with h_{10} ok. So, that gives you essentially this part and then again similarly right you have got 11 and 11 multiplied with S inverse of $k_{13,10}$ XORed with C_{13} XORed with the corresponding 9 down value which is nothing, but $k_{10,9}$ round key expressed in 10th round expressed using 10th round key is S time. So, this is essentially this key that is S multiplied with $k_{23,10}$ XORed with $k_{22,10}$ and this XORed with similarly write with 13 times S inverse of $k_{22,10}$ XORed with C_{22} XORed with $k_{20,10}$ XORed with S times $k_{33,10}$ XORed with $k_{32,10}$.

And that is nothing, but you can see here is essentially this part of the key XORed with the 14 11 13 and then finally, now 9. So, you have got S inverse of $k_{31,10}$ XORed with C_{31} XORed with $k_{30,10}$ again this is this part of the key k_3 is $k_{3,0,10}$ XORed with S of $k_{03,10}$ XORed with $k_{0,2,10}$. And so, therefore, right I mean this is essentially you know like one part and you have to essentially do the same stuff with you know like I am not writing this part, but only that this part is for the faulty ciphers. So, therefore, you have to just replace these equations with the star variants of them. So, therefore, right we will if you just. So, therefore, this shows that you can express $2 p$ in terms of the 9th out ok.

So, you can continue this in a similar fashion basically as we have seen the dependencies like we have seen like how to write one of the equations to get to the point of to p 0.

(Refer Slide Time: 25:22)



So, we can basically continue in this way and basically just rather you can elaborate the equations in a similar fashion, but what is important to notice the number of key bits which essentially are responsible for calculating the values of these differentials $2 p_0 p_0$ and $3 p_0$.

So, therefore, right I will just write for the remaining 3 bytes what are the key bytes which are involved. For example, for p_0 we have got k_{03}^{10} k_{02}^{10} k_{13}^{10} and k_{12}^{10} and k_{21}^{10} k_{23}^{10} k_{22}^{10} . So, you have got k_{03} k_{02} k_{13} k_{12} k_{21} k_{23} and k_{22} followed by k_{30}^{10} k_{33}^{10} and k_{32}^{10} . So, like this right you for the other p_0 differential you will have k_{02}^{10} k_{01}^{10} k_{11}^{10} k_{12}^{10} k_{11}^{10} k_{20}^{10} k_{22}^{10} k_{21}^{10} and then k_{33}^{10} and k_{32}^{10} followed by k_{31}^{10} ok. Again for the remaining differential which is $3 p_0$ you have got the $3 p_0$ depending on k_{01}^{10} k_{00}^{10} , k_{10}^{10} k_{11}^{10} k_{10}^{10} has already been taken care of. So, you have got k_{23}^{10} k_{21}^{10} k_{20}^{10} and then finally, k_{32}^{10} and k_{31}^{10} you also have k_{30}^{10} .

So, now if you observe the interestingly. So, one possibility is that you know that there are 4 equations ok, so, 4 differential equations. So, we have already seen one of the differentials in the previous for example, this was one of the differentials. So, therefore, right if I just randomly you know like take my keys and I substitute them then I get a

value of 2 p. So, I can take them and I can you know like pretty much reduce it by the other 3 equations and therefore, right I mean if I do that then there will be like to power of 32 like for every 2 power 32 keys I have to kind of substitute in these equations I have to solve them ok.

So, I can do slightly better by observing an important property of the key schedule here; that means, if you observe the number of key bytes which are responsible for example, if I just quickly write down this k 0001 k 02 k 0 3 k 10 k 11 k 12 k 13 k 20 k 21 k 22 k 23 and k 30 and k 31 k 32 and k 3. So, this is your 10th round suppose key bytes you will see that these 2 equations there is no key byte which actually depends upon these 2 key bytes it does not depend upon k 00 and it does not depend upon k 10.

So, that implies that you can actually do an interesting you know like you can utilize this in an interesting manner and you know like that can be you know like exploited to reduce the time complexity of the attack. So, with this background let us just you know like get back and see that how you can reduce the we have already reduced from 2 power of 128 to 2 power of 32.

(Refer Slide Time: 29:23)

Reducing the Search space

- Find 2^{32} candidates K^{10}
- Find 2^{32} Candidates of K^9 using keyschedule
- Reduce K^9 to 2^8 candidates

Differential Equations:

$$\begin{aligned} 2f' &= \Delta S_9^{(0,0)} \\ f' &= \Delta S_9^{(0,1)} \\ f' &= \Delta S_9^{(0,2)} \\ 3f' &= \Delta S_9^{(0,3)} \end{aligned}$$

Handwritten calculation: $2^{24} \times 2^{32} = 2^8$

So, now what you can you can use the equation that we are just now seen and you know like there are a for further differential equations and therefore, right you know that for 4 those for the differential equations which you can summarize in this format. So, this is the compact way of writing them. You know that the probability of a random key to

satisfy this is minus 2 power of minus 24 and you have got to power of 32 keys from the first phase. So, in average you will be 2 power of 8 keys will survive.

But now if you take all the 2 power of 32 keys and if you plug into them then your time complexity is still 2 power of 32. So, that what, so, now, we will exploit the just now the fact that we have seen and reduce you know like the time complicity of the attack. So, therefore, if you do this right then definitely you can reduce the key size, but how can we improve the upon the time complexity.

(Refer Slide Time: 30:14)

Results

- Requires 2^8 brute-force search.
- Time complexity 2^{32}

swayam THE ONLINE EDUCATION

So, therefore, right what we do now is we observe as well. So, this is essentially which we can trivially observe here is that you can reduce the key size to 2 power of 8 values, but the time complexity is 2 power of 32.

(Refer Slide Time: 30:26)

Reducing Time Complexity

- Existing DFA required to test 2^{32} candidates of K_{10} by the 8th round differential equation.

$$\begin{aligned} 2f' &= \Delta S_9^{(0,0)} \quad (1) \\ f' &= \Delta S_9^{(0,1)} \quad (2) \\ f' &= \Delta S_9^{(0,2)} \quad (3) \\ 3f' &= \Delta S_9^{(0,3)} \quad (4) \end{aligned}$$

Diagram illustrating key schedule components:

k_0	k_7	k_8	k_9	k_{10}
k_1	k_4	k_5	k_6	k_{11}
k_2	k_3	k_{12}	k_{13}	k_{14}
k_{15}	k_{16}	k_{17}	k_{18}	k_{19}

- Equations (2) and (3) does not contain key byte k_0 and k_1

swayam

So, what you can now do is based upon the previous observation that it does not depend upon 2 key bytes. So, this is another you know like naming convention that we have. So, there are different the way I you know like short the key bytes essentially was by the rows and the columns. So, typically another way of you know like or another naming convention is in this way like k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10} k_{11} and so on till k_{15} ok.

So, this is essentially in a you know like you are basically doing the across the columns ok. So, here we have seen that in you know equations 2 and 3 does not depend upon these 2 key bytes remember this was my k_{00} and k_{10} in the previous nomenclature that is it does not depend upon k_0 and k_1 . So, therefore, what you can do is, you can do this attack in a nicely you know in a divide and conquer manner.

(Refer Slide Time: 31:27)

So, what you can do is elaborated here for example, what you can do now is you can try to accelerate this process by breaking up. So, this is you know the initial key quartets for example, you know like suppose this is your first and fourth key quartet. So, remember that you know like you have got a 1 b 1 c 1 d 1 these are all your key your key quartets and each of them have got size 2 power of 8 ok. Because you have got for every key quarter there are 2 power of 8 values; 2 power of 8, 2 power of 8, 2 power of 8, 2 power of 8 that was that is how we got 2 power of 32.

But now what you can do is that one of these key quartets you can actually break it up into 2 parts ok. So, one part is shown here as you know like L 1 and the other part is L 2 ok. So, note that there are some interesting observations here if you see the you know the contents of this table we have written you know like a 1 b 1 c 1 d 1, a 2 b 1 c 1 d 1; that means, the way the key quartets will occur ok. So, you will find that you will get solutions like a 1. So, suppose you know like yeah for 3 key bytes if you get b 1 c 1 d 1 you will find that there are 2 values of a 1 a 2 which will be along with this ok.

So, I am not explaining why this happens, but I would rather leave it to an exercise with a small hint that is you know like the form of these equations is of this form right $S^{-1}(x) \oplus S^{-1}(x \oplus \alpha) = \beta$. So, therefore, if I fix alpha and beta and suppose you know like you get a solution at x equal to x 1 this implies that you also have another solution at x equal to x 1 XOR of alpha.

So, therefore, there are 2 solutions for every beta that you get and that is why you get the solutions in paired. So, therefore, now is that is why you can split this into 2 parts and you can essentially you know like you can write those 2 key value those 2 values of k_0 for example and write a_1 a_2 and in the other table right you have got the remaining values like b_1 c_1 and d_1 ok.

So, likewise you can split up this key quartet also into 2 parts L_3 and L_4 . So, now, what you can do is remember that we know that equation 2 and 3 does not depend upon k_0 and k_1 . So, you can kind of leave that and the other part of the key right I mean other part of the key quartet that is L_2 the size of these right is 2^7 because totally it was 2^8 which we have broken up into 2 parts. So, here there are 2^2 values and here there are remaining 2^7 values.

So, therefore, the maximum size of this list L_2 becomes 2^7 likewise the maximum size of you know like the of this list L_4 is also 2^7 . So, therefore, what I do is I take 2 these 2^7 these 2^7 and for the other key quartets. So, in the second in the third key quartets where there are 2^8 and 2^8 values. So, totally if I multiply I get you know 7 plus 7 14 plus 16 and I take them and I reduce them by test 1 which is my equation 2 and 3 ok.

And then right I bring this k_0 and k_1 and I reduce them by test 2 remember in test 2, I can actually paralyze this process. So, you can basically pretty much parallel do them do these tests in parallel and you know like you can further reduce it.

(Refer Slide Time: 34:47)

Reducing Time Complexity (Cont.)

- From first phase of the attack we have four quartets $\{k_0, k_{13}, k_{10}, k_7\}$, $\{k_{12}, k_9, k_6, k_3\}$, $\{k_8, k_5, k_2, k_{15}\}$ and $\{k_4, k_1, k_{14}, k_{11}\}$ each of size 2^8
- Split the first and fourth quartets in to two list of size 2^7
 - L_1 contains k_0
 - L_2 contains k_{13}, k_{10}, k_7
 - L_3 contains k_1
 - L_4 contains k_4, k_{14}, k_{11}
- The second and third quartet are represented by two lists L_5 and L_6 each of size 2^8

swayam

So, basically right what I am trying to say here is summarized also in the you know like in the next discussion here, for example, what you can observe here is that L_1 contains k_0 , L_2 contains k_{13}, k_{10} and k_7 there is a remaining 3 bytes L_3 contains k_1 L_4 contains k_4, k_{14} and k_{11} . So, the second and a third key quartets are represented by 2 lists L_5 and L_6 each of dimension to power of 8

(Refer Slide Time: 35:05)

Reducing Time Complexity (Cont.)

- Equation (2) and (3) require list L_2, L_4, L_5 , and L_6 .
- Number of possible keys required $2^7 \times 2^7 \times 2^8 \times 2^8 = 2^{30}$.
- 2^{22} candidates satisfy the equation (2) and (3).
- Combine each of 2^{22} candidates to four values of k_0 and k_1 from lists L_1 and L_2
- Number of candidates increased to 2^{24} .
- Reduce 2^{24} candidates to 2^8 by equation (1) and (4).

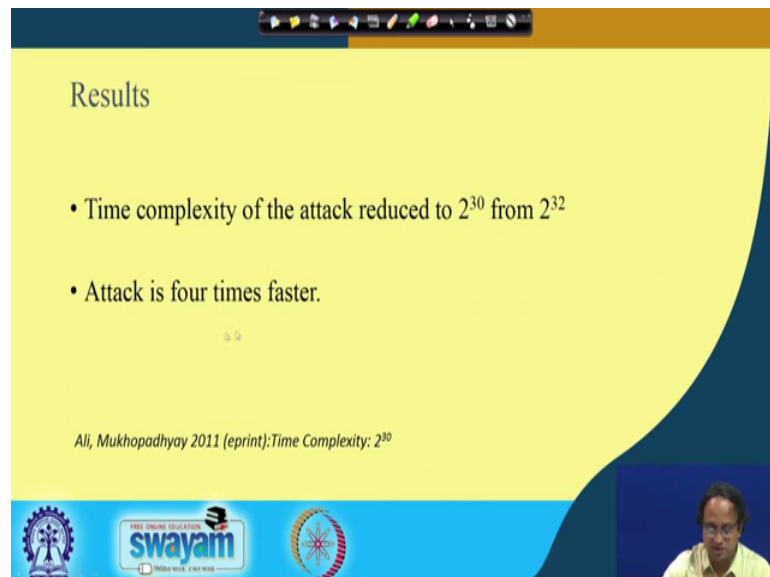
swayam

So, therefore, now if you look at equation 2 and 3 you have got these lists and therefore, the total dimension is 2 power of 30 and therefore, if we now use equations 2 and 3 then

you can reduce it to 2 power 22 candidates ok. Now these 2 power 22 candidates you combine with the other 4 values of k 0 and k 1. So, therefore, there are size of these right will increase to 2 power of 22 multiplied with 2 power of 2 that is 2 power of 24.

And then you use equations 1 and 4 to reduce these 2 power of 24 to the final 2 power of 8 values. So, remember if you do in this way then your size never exceeds 2 power of 30 your size right is always 2 power of 30 and therefore, the time complexity of the attack does not shoot beyond 2 power of 30. So, there is an improvement over the previous 2 over the previous attack where the time complexity was 2 power of 32.

(Refer Slide Time: 35:56)



Results

- Time complexity of the attack reduced to 2^{30} from 2^{32}
- Attack is four times faster.

Ali, Mukhopadhyay 2011 (eprint): Time Complexity: 2^{30}

So, therefore, right you have basically been able to do the attack in a technique which is 4 times faster.

(Refer Slide Time: 36:01)

Conclusions

DFA on AES can be done with a single fault!

A single byte fault at the input of AES-128 can reduce the key size to only around 256 guesses.

In the optimized version one can do the attack with a time complexity of 2^{30}

References:

1. Debdeep Mukhopadhyay: An Improved Fault Based Attack of the Advanced Encryption Standard. AFRICACRYPT 2009: 421-434
2. Michael Tunstall, Debdeep Mukhopadhyay: Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault. IACR Cryptology ePrint Archive 2009: 575 (2009)

The slide features a yellow background with a blue wave-like graphic on the right side. At the bottom, there are logos for Swamyam and other educational institutions, along with a small video inset of the presenter.

So, to summarize DFA on AES can be done with a single fault, a single byte fault or the input of AES 128 can reduce the key size to only around 256 guesses. And in the optimized version one can do the attack with the time complexity of 2 power of 30 which is like a 4 times improvement over straight forward approach using the property of the AES key schedule.

(Refer Slide Time: 36:27)

References

□ Debdeep Mukhopadhyay and Rajat Subhra Chakraborty, Hardware Security: Design, Threats and Safeguards, CRC Press

The slide features a yellow background with a blue wave-like graphic on the left side. The word 'References' is written in a large, stylized font. A book cover for 'Hardware Security: Design, Threats and Safeguards' by Debdeep Mukhopadhyay and Rajat Subhra Chakraborty is shown. At the bottom, there are logos for Swamyam and other educational institutions, along with a small video inset of the presenter.

So, here are some references that we have used in the discussion and of course, like you can refer to the original textbook for more details. So, with this I would like to say.

Thank you.