

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

Lecture – 49
Multi – Byte and Key Scheduling Based Fault Analysis of AES

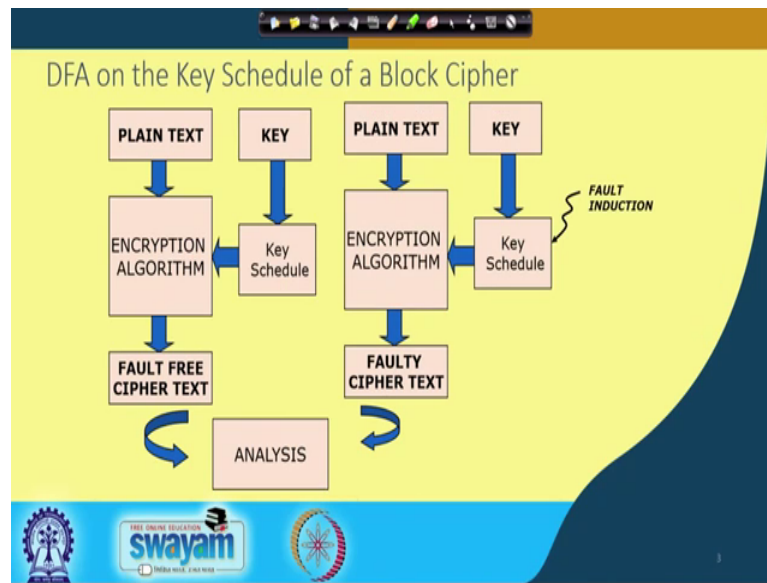
So, welcome this class on Hardware Security. So, today we shall be trying to look into you know when the fault is effecting the key schedule of a block cipher. And, we will be talking with respect to the AES algorithm as we were discussing.

(Refer Slide Time: 00:30)



So, this is what we will be covering today on how to perform fault attack on AES targeting the key schedule of AES.

(Refer Slide Time: 00:36)



So, therefore, this is a kind of recapitulation of what happens in a fault attack scenario in particular in context to differential fault attacks. So, you have got plain text and this is the normal encryption algorithm and you get the fault free cipher text. In another scenario you have got the plain text and you get the faulty cipher text, but remember right that we have got the key schedule also which is in play; that means, the input key is getting transformed into the round keys, by using this key scheduling algorithm ok.

So, now what we are trying to do is that we are trying to affect the fault, but we are trying to affect the key schedule of AES ok. And, we know that when the fault is affecting the key schedule then that would imply intuitively that the round keys will also get affected. And, if the round keys are getting affected when the round key is kind of mixed with the data path the data path also gets affected ok. So, it becomes a more complex scenario, where both data path as well as the keys are getting corrupted, but how do you kind of analyze a block cipher in such a scenario ok because, this can be a very practical fault model ok.

In fact, right this can be a very serious scenario, where you; suppose you know like you have got the keys kind of you know like calculated at the beginning of a particular encryption. And suppose right you basically corrupt the key scheduling at that point; that means, the round keys are kind of permanently wrongly counted, you know like calculated.

So; that means, right as we will see later on if you doing something like a comparison based check right, where you basically compare with the reference. Here every time you repeat the encryption you will always get the wrong result ok. And therefore, right that there can be quite serious issue. So, therefore, right faults targeting the key scheduling right or key scheduling of any block cipher are the topic which should be carefully investigated.

(Refer Slide Time: 02:13)

The slide contains the following text:

- Kim: 2011, 2 faults to recover 96 bits of the AES-128 key.
- Ali, Mukhopadhyay (FDTC) 2012: 1 fault with 2^8 remaining keys.

The diagram shows the AES-128 key schedule with annotations:

- Red circles around key bytes in the 8th, 9th, and 10th rounds.
- Red arrows labeled K_8 , K_9 , and K_{10} pointing to the respective round keys.

Logos for 'swayam' and 'FREE ONLINE EDUCATION' are visible at the bottom of the slide.

So, let us try to kind of take a look in to how the DFA or how DFA has been reported on key scheduling in AES. So, this the recapitulation of how the key schedule works in AES-128, again without loss of generality. So, this the 8th round and as you can see that you know like this is the 8th round key, followed by the 9th round key, followed by the 10th round key ok.

So, how does this like round transformation work. So, remember that this is suppose you know like the last column of the 8th round key. So, what you basically do is that you do a sub word; that means, every byte here is transformed by an S box ok. And, then you do something which is called as a root word, which means you do a circular left shift ok. So, circular left shift would mean that you basically end up in doing a circular shift like this ok; that means, like this byte gets moved one position, this byte gets moved one position, this byte gets moved one position and this byte comes to this byte location ok.

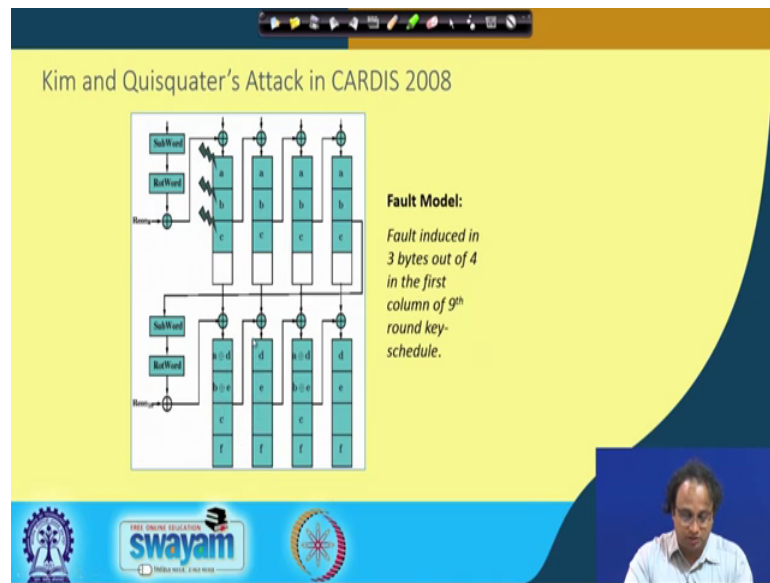
Again remember after the S box has been operated ok. And, then you basically XOR with this column; that means, you XOR with this column to give the first column of the 9th round key. The remaining 3 columns are obtained in a much more simple way what we do is that we kind of XOR this, as you can see right we XOR this with this column; that means, with this column to get essentially this column and likewise, you XOR this column with this column to get this column and again you XOR this column with this column to give you this column ok.

As you can easily understand that because of this right the key scheduling is also invertible, because now if I want to obtain this column and if I am being provided with the 9th round key then I just need to take an XOR of this column and this column to obtain this column ok. Likewise, I will take the XOR of this column and this column to obtain this column and I will obtain in like similarly I can also obtain this column ok. Again for the first column I will do a little bit of trick, because I have to also accommodate this operation, but that is also not very difficult to do.

So, therefore, right we can essentially you know like this is the reversible process then therefore from K 9, I can obtain K 8 and likewise from K 10 I can obtain K 9 ok. And, that is something that implicitly we are been assuming although right.

So, now we will be talking about the, what we can do right. And in fact, right there was a paper which was published in 2011, which shows how to use 2 faults on the AES key schedule to remove to recover 96 bits of the AES-128th key, for the remaining key positions right, it basically had to do a brute force search ok.

(Refer Slide Time: 04:53)



And this attack was essentially due to Kim and Quisquater and has been essentially kind of shown here. So, this is again actually reported in CARDIS in 2008 apparently where you take you know like when you consider the AES key. And, imagine that you know like what you do is you kind of corrupt in the first column of the 9th round key schedule; so, as you can see that these 3 bytes are corrupted by corrupted in a random way ok.

Now, because of the key scheduling of AES what happens is that this fault propagates right and therefore, right these 3 bytes gets corrupted. And likewise right this is how the fault kind of propagates ok. Now, if you consider the next key, again you know like you will see that the all the columns or all the entire round key is now getting corrupted in the 10th round key.

(Refer Slide Time: 05:38)

Propagation of the fault pattern

Summary of the Attack:
Requires two faulty ciphertexts to retrieve 12 bytes of the AES 10th round key.
Thus brute force search of 2^{32} is still needed!

And, these keys are now basically mixed with the original data path ok. So, here you can see that this is the normal data path, but remember that the key has now been corrupted. So now, when we are talking about the differential picture, the differential is not only involving the data path, but also involves the differential in the key ok. So, therefore, right if you want to observe remember that here when you are, because of fault is not in the data path the differential here in the state matrix is 0, because there is no difference actually.

But, there is a difference in the key and that essentially gets manifested in the data path now and that propagates you know like in a normal way. And finally, right the 10th round key comes in and therefore, you get you know like distribution which is kind of looks pretty much random in the entire cipher text.

So, now in this particular attack right just to summarize it requires 2 faulty cipher text to retrieve 12 bytes of the AES key or the AES 10th round key, but there is a brute force search which is required of 2 power of 32, which is still required ok. And therefore, right in this particular attack you still need to kind of you know like recover, you know like 4 bytes and 4 bytes would mean 4 into 8 which is 32. So, therefore, you have to still do brute force which is of you know like something like 2 power of 32.

And, therefore, right this not a very I would say an efficient fault attack and there seems to be a scope of improvement ok. More importantly right you see that you need for faults

of this nature and this is also kind of a restricted fault model, because you need fault of a specific type and it is not a very generic fault model in that sense.

(Refer Slide Time: 07:17)

Can we perform the attack with one fault?

- The present attack:
 - Relies on a single-byte fault model
 - Performs the attack with a single faulty ciphertext.

The slide features a yellow background with a blue footer containing the Swayam logo and a small video inset of the presenter.

So, now what we will see or what we will investigate is how we can do this attack using again you know like our favorite single byte fault model. And also right you will perform the attack with a single faulty cipher text only with one shot ok.

(Refer Slide Time: 07:27)

Propagation of the Fault required in the single fault attack

The fault is induced at the 8th round key. Faulty Byte spreads to the first row of the 8th round key, the 1st and 4th rows of 9th round key.

$$\begin{aligned}
 q &= S[K_{0,3}^8] \oplus S[K_{0,3}^8 \oplus p] \\
 &= S[K_{0,3}^8 \oplus K_{0,2}^8] \oplus S[K_{0,3}^8 \oplus K_{0,2}^8 \oplus p] \\
 &= S[K_{0,3}^{10} \oplus K_{0,1}^{10}] \oplus S[K_{0,3}^{10} \oplus K_{0,1}^{10} \oplus p]
 \end{aligned}$$

Likewise, the fault spreads to the 1st, 3rd, and 4th rows of the 10th round key. Thus,

$$\begin{aligned}
 r &= S[K_{3,3}^9] \oplus S[K_{3,3}^9 \oplus q] \\
 &= S[K_{3,3}^{10} \oplus K_{3,2}^{10}] \oplus S[K_{3,3}^{10} \oplus K_{3,2}^{10} \oplus q]
 \end{aligned}$$

The slide includes a diagram of a cipher's round function with red annotations showing fault propagation. It also features a yellow background, a blue footer with the Swayam logo, and a small video inset of the presenter.

So, for that let us take a look at how the; what is the type of fault. So, what we will do is that we will again consider the 8th round key rather than 9th round key and you know

like we will basically kind of inject the fault in this way. So, we will basically say you know like take the first column here of the 8th round and that this shaded region shows how the fault is affected.

Because of the diffusion that I just now said right this kind of gets xor with the next round remember that now the key shielding takes place the fault kind of propagates here, it again propagates here and it propagates here. So, now if you consider further diffusions right you remember that what will happen over here is that this particular you know like, this particular differential line will get transformed by the S box ok.

And, then you will be doing a circular shift, because of the circular shift this byte will come to this location. And therefore, right when you now XOR this with this particular column, then the differential will look like this because you know; like because of this right what will happen is that this differential will be 0, but here you will have some non 0 difference. And, now when you XOR this with this you will get distribution where this is p and this is q whereas, these 2 differentials are 0 ok.

So, likewise when you kind of XOR this right with this one then this p and this p will get cancelled out. So, this will be 0 here and 0 and 0 and then when you when you kind of XOR this with this you will again get p and q. So, this kind of will repeat. And therefore, you will get again 0 0 q here ok.

So now, when you basically you know like again carry this in similar fashion. So, again you will see that this q will now move here move here because of the circular shift. And therefore, right you will have something like an r here and whereas, you know like you will have 0 which will come down here and so on right, we will see this will move here, this will move here, and this will move here. So, we will have something like this ok.

So now, if you take an XOR of this and this so, you will get p, which comes over here because p XOR with 0 will be p, likewise right this r will propagate here and you will have q XOR with 0 so, it will be q ok. So, likewise when you take an XOR with this then q and q will get cancel. So, you will have 0 here this will 0 and you will have p and r. And finally, right when you XOR this with this again this will be 0, this is 0 and this is r and q. And again when you XOR this then this q and this q will cancel so, you will have essentially a difference pattern of this type.

So, now that would imply that, I can now form you know like some kind of some equations as shown here. For example, if you observe the propagation here ok. So, you can see that this q can be related with this p . For example, if you consider right the 8th round key and you can observe that in one of the cases right we have got the actual key remember right, we are we are suppose you are considering the key right of the remember that my key is written in this way this is my $K_{0,0}$, this my $K_{0,1}$, this is $K_{0,2}$, this is $K_{0,3}$ ok.

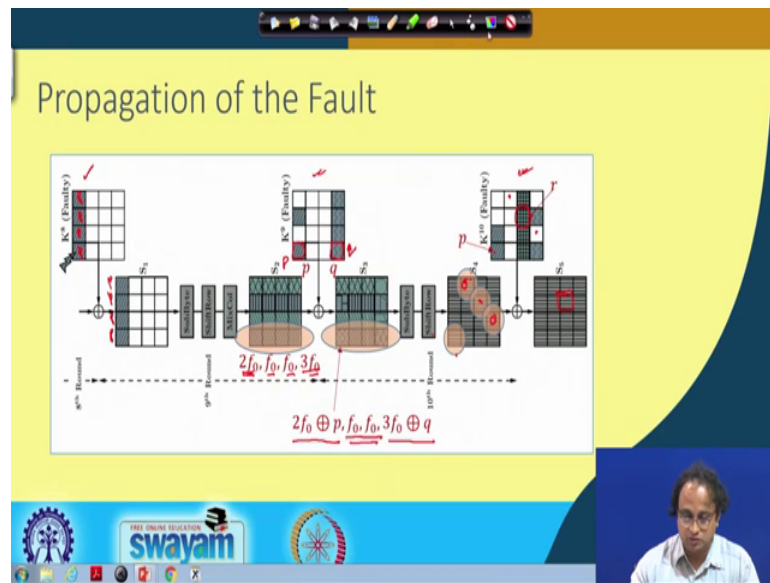
This is my $K_{1,0}$, $K_{1,1}$, $K_{1,2}$, $K_{1,3}$ right this is how we have been writing here. So, this particular byte right is referring to $K_{0,3}$ of the $K_{0,3}$ of 8 ok. So, that is essentially shown here and likewise if you consider the differential here then this is $K_{0,3,8}$ XOR with P ok, because the P is essentially differential in key.

So, now, if I take the s box here then that essentially would give me my this q differential And therefore, I can write this equation ok. Remember because of the invertibility that I showed this $K_{0,3,8}$ can be obtained by the XORing of the 9th round keys, which are adjacently placed ok. And likewise right for example, you can observe that. So, for example, like I know as I said that this particular column right can be obtained right from this column and from this column by XORing them ok. And, therefore, right I do an XOR between $K_{0,3,9}$ and $K_{0,2,9}$ to get $K_{0,3,8}$ and likewise continuing this fashion you can express this in terms of the 10th round keys.

So, likewise you can also observe the fault propagation from the 9th round to the 10th round And, then you can essentially form similar equations as shown here. In fact, you know like I would request you to look into for example, this column here, like you know like how the fault propagates from q and gets converted into r to obtain this relationship ok. So, for example, remember that, when you are considering this particular byte then this is your you know like $K_{3,3}$ ok.

So, in one case you have got $K_{3,3,9}$, in the other case you got $K_{3,3,9}$ XOR with q because q is the differential and if you XOR them then you have got this r ok. So, that is the simplest ways where you can get the equation likewise this 9th; now 9th round keys you can write in terms of the 10th round keys ok. So, that explains you know like the how you can model the propagation of the fault in the key schedule of a AES 128.

(Refer Slide Time: 12:29)



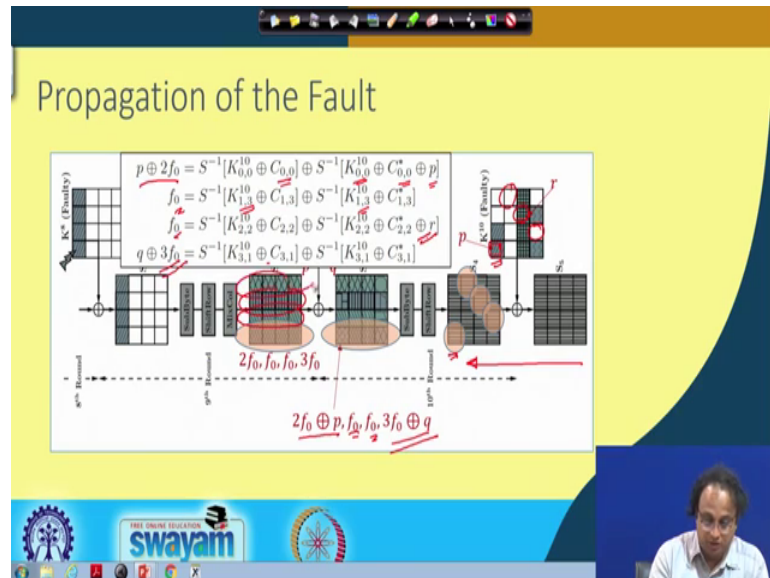
So, now let us you know like see right what is the effect of this fault on the propagation of the AES algorithm. So, here we take the input 8th round and remember that I am considering the effect of the fault. So, this is my fault propagation I have seen p p p p. So, this is the written in this way in a horizontal fashion, this is exactly the 9th round differential and this is the 10th round differential as we have seen ok.

So, in particular we are basically seeing you know like the fault because of this p, p, p, p nature, gets propagated here as p, p, p and p right. And, then you have got the sub by shift row and the mix columns, because of the mix columns, now if you observe each of these columns here. So, I am showing here one column, this column here has got 2 f 0, f 0, f 0 and 3 f 0 ok. So, now you see that the; now see that this key is also corrupted and the key right here this particular differential is p whereas, this differential is q ok.

So, therefore, right I mean when you basically kind of XOR them, then what happens is that this byte get corrupted as 2 f 0 XOR with p ok. Whereas, this byte gets corrupted with 3 f 0 XOR with q whereas, these 2 remains as f 0, where f 0 is some random byte value ok. So, now if you have observe the 10th round, the last round right you see that because of the shift rows this column get distributed like this right. So, it basically kind of gets spread like this. And, now when you are basically bringing in the 10th round key then you have got here for example p and what matters here is r because here by this differential is 0 and this differential is 0.

So, these two does not matter, you can see that this differential here is 0 and this differential here is 0, but only here you have got r ok. So, therefore, this r gets and gets or; gets into the mixed in the XOR and therefore, is also effects this particular byte. So, now, what you can do is that with this background right you can basically what you can do is you can just see you know like what happens here.

(Refer Slide Time: 14:25)



So, for example, like this is how the, you know like let me just clear this. So, this is how your equation looks like. So, what we have done is we have basically concentrated on this column and remember that I am targeting you know like say p XOR with 2 f 0. So, this is your P XOR with 2 f 0.

So, I am now you know like again like working back; that means, I am you know like going from the cipher text and from the faulty cipher text. Remember in the current case I have got K 0, 0 10 so, this is my K 0 0 10 whereas, in the other case it is K 0, 0 10 XOR with p, because there is a p which is a differential ok, that is why I have got K 0, 0 10 XOR with p.

Likewise in this case right there is no differential, because the other I mean in this case as you can see here there is no differential. So, therefore, in this key both of them are K 1, 3 and you know like it is K 1, 3 10 and K 1, 3 10 over here. Whereas, if you consider this right this byte then there is a differential of r and that is accommodate here in this equation and this gives me my f 0 and f 0. So, this stands out for this and this in the in

differential. And here you have got finally, you know like again no differential because you are observing this, but then you are basically getting 3 f 0 XOR with q ok.

So, this your 3 f 0 XOR with q. So, therefore, you can form an equation of this type. Remember this is for one key quartet you can actually form 3 equations for the remaining 3 quartets, because all of them you can similarly form.

(Refer Slide Time: 15:58)

4-sets of Equations

$f_1 = S^{-1}[K_{0,1}^{10} \oplus C_{0,1}] \oplus S^{-1}[K_{0,1}^{10} \oplus C_{0,1}^* \oplus p]$
 $f_1 = S^{-1}[K_{1,0}^{10} \oplus C_{1,0}] \oplus S^{-1}[K_{1,0}^{10} \oplus C_{1,0}^*]$
 $f_1 = S^{-1}[K_{2,3}^{10} \oplus C_{2,3}] \oplus S^{-1}[K_{2,3}^{10} \oplus C_{2,3}^* \oplus r]$
 $q \oplus 3f_1 = S^{-1}[K_{3,2}^{10} \oplus C_{3,2}] \oplus S^{-1}[K_{3,2}^{10} \oplus C_{3,2}^* \oplus q]$

$p \oplus 2f_2 = S^{-1}[K_{0,2}^{10} \oplus C_{0,2}] \oplus S^{-1}[K_{0,2}^{10} \oplus C_{0,2}^*]$
 $f_2 = S^{-1}[K_{1,1}^{10} \oplus C_{1,1}] \oplus S^{-1}[K_{1,1}^{10} \oplus C_{1,1}^*]$
 $f_2 = S^{-1}[K_{2,0}^{10} \oplus C_{2,0}] \oplus S^{-1}[K_{2,0}^{10} \oplus C_{2,0}^* \oplus r]$
 $q \oplus 3f_2 = S^{-1}[K_{3,3}^{10} \oplus C_{3,3}] \oplus S^{-1}[K_{3,3}^{10} \oplus C_{3,3}^*]$

$2f_3 = S^{-1}[K_{0,3}^{10} \oplus C_{0,3}] \oplus S^{-1}[K_{0,3}^{10} \oplus C_{0,3}^*]$
 $f_3 = S^{-1}[K_{1,2}^{10} \oplus C_{1,2}] \oplus S^{-1}[K_{1,2}^{10} \oplus C_{1,2}^*]$
 $f_3 = S^{-1}[K_{2,1}^{10} \oplus C_{2,1}] \oplus S^{-1}[K_{2,1}^{10} \oplus C_{2,1}^* \oplus p]$
 $q \oplus 3f_3 = S^{-1}[K_{3,0}^{10} \oplus C_{3,0}] \oplus S^{-1}[K_{3,0}^{10} \oplus C_{3,0}^* \oplus q]$

Solving each of these equations separately, as in the DFA on AES data-path seems to be infeasible!

We need to properly apply a divide-and-conquer strategy.

For a given value of p and q , the 1st, 2nd, and 4th equations, give us 2^8 values of: $(K_{0,1}^{10}, K_{1,0}^{10}, K_{3,2}^{10}, r)$

Likewise, the 1st, 2nd, and 4th equations of this set give us 2^8 values of: $(K_{0,3}^{10}, K_{1,2}^{10}, K_{3,0}^{10})$.

Combining, we have 2^{16} values for the 6 key bytes $(K_{0,1}^{10}, K_{1,0}^{10}, K_{3,2}^{10}, K_{0,3}^{10}, K_{1,2}^{10}, K_{3,0}^{10})$

$2^4 \times \frac{1}{2^6} = 2^8$

So, if you do that right, then you will get equations of this nature and this is how the four, these are the remaining three sets of equations ok. As you can see here these are the three sets of equations, again I leave it to you as an exercise to verify them, but you can similarly derive them ok.

So, now, what we would like is we would like to solve each of these equations separately as you know, but you want to do that then you see just like if you try to solve them exactly as we have seen in the DFA context, it is not practically possible. Why know because if you see each of the equations, there are four key quartets such you can observe here ok, there is an f 1 byte, there is also q and there is also r and there is also p.

So, there are many variables and therefore right, it is difficult or at least does not look very very evident, how the you know like the possible or the possible values will gets reduced because reduction is very important right otherwise you will not get very

efficient key recovery mechanism. So, therefore, we need to develop a proper divide and conquer strategy by using various equations that we have.

So, what we try to do is first we basically try to take the first, second and the fourth equation in each of them. So, what we do is we basically observe that in all these equation sets ok, in the third equation there is a variable r ok. For example, you see that what we observe here is you know like for each of these equations for example, if you observe right then so, each of this equation in the third equation right you see that there is an r which is involved.

So, therefore, what we do is basically keep aside these third equations. And we basically for a given value of p and q . Now p and q can take 2^4 values, imagine at this point you have been given a value of p and q ok. So, later on we will multiply with all possible values of p 's and q 's. So, what we do is that we consider the first, the second and the fourth equations of each of these sets ok. See for example, if I consider of this set ok then you see that the key right or the then the keys bytes were involved are $K_{0,1,10}$, $K_{2,3,10}$ and sorry not this one $K_{0,1,10}$ $K_{1,0,10}$ and $K_{3,2,10}$ this is essentially shown here.

And, we know that there are 2^{24} possible values here, but then you have got 3 equations and therefore, this has got a probability of $1/2^{16}$, which means there are 2^8 possible keys which will survive ok. So, likewise you again try this for this set you get another 3 key bytes you try this for another if you combine this two things. Therefore, you will have 2^{16} values for these 6 key bytes ok so, this 6 key byte nothing, but a union of this list and this list.

(Refer Slide Time: 18:51)

Further Optimizations

$$\begin{aligned}
 q &= S[K_{0,3}^8] \oplus S[K_{0,3}^8 \oplus p] \\
 &= S[K_{0,3}^9 \oplus K_{0,2}^9] \oplus S[K_{0,3}^9 \oplus K_{0,2}^9 \oplus p] \\
 &= S[K_{0,3}^{10} \oplus K_{0,1}^{10}] \oplus S[K_{0,3}^{10} \oplus K_{0,1}^{10} \oplus p]
 \end{aligned}$$

$$\begin{aligned}
 p \oplus 2f_2 &= S^{-1}[K_{0,2}^{10} \oplus C_{0,2}] \oplus S^{-1}[K_{0,2}^{10} \oplus C_{0,2}^*] \\
 f_2 &= S^{-1}[K_{1,1}^{10} \oplus C_{1,1}] \oplus S^{-1}[K_{1,1}^{10} \oplus C_{1,1}^*] \\
 f_2 &= S^{-1}[K_{2,0}^{10} \oplus C_{2,0}] \oplus S^{-1}[K_{2,0}^{10} \oplus C_{2,0}^* \oplus r] \\
 q \oplus 3f_2 &= S^{-1}[K_{3,3}^{10} \oplus C_{3,3}] \oplus S^{-1}[K_{3,3}^{10} \oplus C_{3,3}^*]
 \end{aligned}$$

$$\begin{aligned}
 r &= S[K_{3,3}^9] \oplus S[K_{3,3}^9 \oplus q] \\
 &= S[K_{3,3}^{10} \oplus K_{3,2}^{10}] \oplus S[K_{3,3}^{10} \oplus K_{3,2}^{10} \oplus q]
 \end{aligned}$$

Using, the equation based on the fault propagation in the keyschedule one can reduce the possible choices for these 6 key bytes to $2^{16} \times 2^{-8} = 2^8$ values.

Now, we use the 1st, 2nd, and 4th equations from this set and obtain finally 2^{16} values for the 9 keybytes,
 $(K_{0,1}^{10}, K_{1,0}^{10}, K_{3,2}^{10}, K_{0,3}^{10}, K_{1,2}^{10}, K_{3,0}^{10}, K_{0,2}^{10}, K_{1,1}^{10}, K_{3,3}^{10})$

As now we have the values for the keybytes $K_{3,3}^{10}$ and $K_{3,2}^{10}$, using this equation based on the faulty keyschedule, we can compute r .

So, now, what we do next is basically right, we basically try to kind of combine this and see you know like how we can conjure the key further ok. So, what we do here is that in order to develop a further optimization, we basically you know like consider the fault propagation that we have seen ok. Remember that we are in equation on q using the 8th round keys and subsequently express in the according with the 10th round keys. So, what we do now is that, we using this equation based on the fault propagation in the key schedule one can reduce the possible choices for this 6 key bytes, to because what is the probability that a random you already got a value of q go back right, you see that these keys like $K_{0,3}^{10}$ $K_{0,1}^{10}$ are already in your set $K_{0,3}$ and you know like $K_{0,1}$ are already in your set ok.

So, we have already are there in my set and therefore, right what I can do is I will just plug into this equation and see whether this satisfies this equation. And the probability of that being happening is 2 to the power of minus 8 so; that means, like 2 to the power of 16 will be multiplied with 2 to the power of minus 8. And therefore, you will have 2 power of 8 values for this 6 key bytes ok.

Now, what you can do is you can again go back to the remaining key s set that we have; so, we have still you know like one set that we have not looked into for example, this set. And considered again the first second and the fourth equations from this set in the similar fashion and again you know like obtain finally, 2 to the power of 16 values for the 9 key byte. So, remember that here there are 3 additional 3 key bytes.

So, if we combine with the 6 key bytes that we have obtained; that means, you have got in total 9 key bytes ok. And, the probability that this will basically satisfy this equation right is 1 by 2 to the power of 24. And remember right you have got 2 to the power 24, 3 key byte values. So, totally right you will have 2 power of 16 values still survive ok.

So, therefore, you will have 2 power of 16 values for the 9 key bytes ok. And so, now, what we do now is that we again look into the other equation that we had with related to r. And we would like to now go back to the equations that we have left out that is the third equations, but in order to do that we need the value of r. So, how do we get the value of r, we will use this equation for r and we already have as you can see you know like is $K_{3,3}^{10}$ and $K_{3,2}^{10}$ and therefore, right we can plug in to this equation we have the value of q we obtain the value of r ok.

(Refer Slide Time: 21:15)

End of Phase 1

$$f_1 = S^{-1}[K_{2,3}^{10} \oplus C_{2,3}] \oplus S^{-1}[K_{2,3}^{10} \oplus C_{2,3}^* \oplus r]$$

$$f_2 = S^{-1}[K_{2,0}^{10} \oplus C_{2,0}] \oplus S^{-1}[K_{2,0}^{10} \oplus C_{2,0}^* \oplus r]$$

$$f_3 = S^{-1}[K_{2,1}^{10} \oplus C_{2,1}] \oplus S^{-1}[K_{2,1}^{10} \oplus C_{2,1}^* \oplus r]$$

From the left out equations of each set, keeping the values of f_1, f_2, f_3 from the previous iterations, we obtain one value for the keybytes $(K_{2,3}^{10}, K_{2,0}^{10}, K_{2,1}^{10})$. Thus, combining with the remaining we have 2^{16} values for:

$$(K_{0,1}^{10}, K_{1,0}^{10}, K_{3,2}^{10}, K_{0,3}^{10}, K_{1,2}^{10}, K_{3,0}^{10}, K_{0,2}^{10}, K_{1,1}^{10}, K_{3,3}^{10}, K_{2,3}^{10}, K_{2,0}^{10}, K_{2,1}^{10})$$

Now using the remaining set we have 2^8 values for the keyquartet: $(K_{0,0}^{10}, K_{1,3}^{10}, K_{2,2}^{10}, K_{3,1}^{10})$. Thus, combining with the rest and considering that there are 2^{16} values for p, q we have in total: $2^{16} \times 2^8 \times 2^{16} = 2^{40}$ candidates for the 10th round AES-key!

Logos: swayam, INDIAN INSTITUTE OF TECHNOLOGY

And, once we obtain the; we can once we compute the value of r, what we do is now we look into these three equations which we have left out ok. Remember that you need the equation set that we have left out one equation, the reason we left out because we did not know the value of r but, now we know the value of r. So, we plug in to that and therefore, right from the left out equations of each set keeping the values of f_1, f_2 and f_3 from the previous iterations, because we had a value of f_1, f_2 and f_3 . We obtain 1 value for the key bytes, because you know like here we will get on an average 1 value. So, therefore, you will get one value of $K_{2,3}^{10}, K_{2,0}^{10}$ and $K_{2,1}^{10}$ ok. So, therefore, what we do

now is that with the remaining 2 to the power of 16 values if you combine them. Therefore, you will have 2 the power of 16 values for now you know like you already had you know like this thing, but now you will also taking up these 3 additional key bytes.

So, now using the remaining set ok so, remember that we had one equation set which we did not consider, this is the first equation set that we have. So, I will now take this equation set and just put into this whether it satisfies ok. So, therefore, we know right that from our previous discussions that if you combine with the rest and considering that there are 2 to the power of 16 values. So, therefore, now if you combine this we have got 2 to the power of 8 values for the key quartet; that means, you know like this particular equation right. There will be 2 to the power of eighth values which will satisfy this equation as we have seen traditionally like and we can easily follow that.

So, now what will happen is that if you combine therefore, this 2 to the power of 16 values that we have already obtained will combine with this 2 to the power of 8 values, but again for p and q there are 2 to the power of 16 values. We assume that I know the value of p and q, but the p and q can also take 2 power of 16 values. So, if you multiply all of them you will have 2 power of 40 candidates for the 10th round AES key ok.

So, definitely like it is a reduction, but you know like at the same time right I mean it shows that, you can actually reduce the key size quite significantly only with one single fault at the input of the or in the 8th round key ok. So, this is the final equation that we basically observed and we obtain this 2 to the power of 40 candidates for the 10th round AES key.

(Refer Slide Time: 23:26)

So, in phase 2 we basically would like to reduce it further as we have done for you know like for the DFA on the AES data path. So, for that right we basically you know like kind of look into this particular you know like column r S 1 and we see that here we can essentially form similar four equations ok. So, this equation is based upon the fact that all the key bytes that we have here they are all p, this differential all p ok. So, therefore, I can again form an equation where p equal to s inverse again remember that I have to go back the inverse mixed column.

So, this how it looks like ok. I am not you may verify this off line, but you can see that pretty much this means I am verifying this equation which is p and likewise this is also p and this is also p and this is also p ok. So, this is just you know like based upon the fact that you are going back the 10th round and also the 9th round. So, we derive them in a similar fashion. Now, there are 4 differential equations remember you have the value of p. So, therefore, right the probability of a random you know like K satisfying this equation is 2 to the power of minus 8 whole power of 4 that is 2 power of minus 32 and here 2 power of 40 keys therefore, 2 power of 8 keys will survive.

So, again you know like just as the DFA on AES data path, you see like single fault you are still able to reduce the AES key size to only 2 power of 8 values ok. So, this essentially you know like; essentially explains how you can do the attack, but remember that if you do the attack in this way, then you have to try all the 2 power of 40 keys and

you have to do the attack, which means that your time complexity is proportional to 2⁴⁰ power of 40 values ok.

(Refer Slide Time: 25:02)

Time Complexity (contd.)

- In the naïve approach, time complexity would be proportional to 2⁴⁰ encryptions. But we can reduce it!
- How?
- Observe, that the following equation depends on 10 bytes of K_{10} .

$$p = S^{-1} [14(S^{-1} [K_{10}^{10} @ C_{0,2}] @ K_{12}^9) @ 11(S^{-1} [K_{11}^{10} @ C_{1,1}] @ K_{12}^9) @ 13(S^{-1} [K_{20}^{10} @ C_{2,0}] @ K_{12}^9) @ 9(S^{-1} [K_{21}^{10} @ C_{3,3}] @ K_{12}^9) @ S^{-1} [14(S^{-1} [K_{02}^{10} @ C_{0,2}] @ (K_{12}^9 @ p)) @ 11(S^{-1} [K_{03}^{10} @ C_{1,1}] @ K_{12}^9) @ 13(S^{-1} [K_{22}^{10} @ C_{2,0}] @ r) @ K_{12}^9) @ 9(S^{-1} [K_{32}^{10} @ C_{3,3}] @ (K_{12}^9 @ q))]$$

K00	K01	K02	K03
K10	K11	K12	K13
K20	K21	K22	K23
K30	K31	K32	K33

K00	K01	K02	K03
K10	K11	K12	K13
K20	K21	K22	K23
K30	K31	K32	K33

Also, considering the dependence of p and q , we have a further key byte involved: $K_{0,3}^{10}$

So, you can you know like reduce the key time complexity even further and therefore, you can still reduce it the question is how ok. So, for that you have to again take a little bit of look into the equations that we have and if you carefully observe it you will find that the following equations. For example, I just show in one of the examples, it actually depends only on 10 key bytes of K 10 ok. So; that means, I have kind of shown here in the red lines where this means the this is the 9th round key and this is suppose the 10th round key ok.

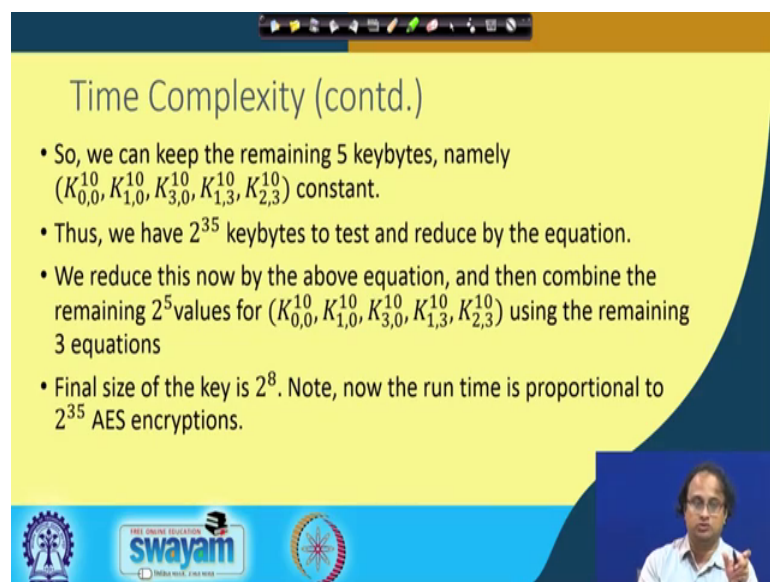
So, what we have shown here is that suppose if we observe that this K 0, 2 10, K 1, 1 10 and K 2, 0 10 and K 3, 3 10 right, this essentially shows you the you know like and other cases K 0, 2 9, K 1, 2 9, K 2, 2 9 and K 3, 2 9. So, I have marked here K02, K12, K22 and K32 of the 9th round key. Remember this column depends upon these two columns of the 10th round as I said.

And, also right you also have you know like K02. So, K 0 2 is already covered in the 10th round key we have got K11. So, it is K11 p again which is covered, then you have got K20 which is this key byte and then you have got K33 for example, which is this K33 ok.

So, if you count right there are 10 key bytes which are affected 4 plus 4 into 2 is 8 plus 2 which is 10 ok. And, in fact right if you also account for the fact that we have a p over here and remember that p and q are independent; are not independent there is a dependence, but that dependence depends upon the $K_{0,3}$ 10 ok. If, you go back right to the equation between p and q then this is how it looks like. So, this is your dependence upon p and q and then the key byte which is involved is $K_{0,3}$ 10 ok. So, therefore, right remember $K_{0,1}$ 10 is already there in my list ok.

So, therefore, right here I have got $K_{0,3}$ 10 therefore, I have got totally there are 11 key bytes which are essentially crucial here.

(Refer Slide Time: 27:00)



Time Complexity (contd.)

- So, we can keep the remaining 5 keybytes, namely $(K_{0,0}^{10}, K_{1,0}^{10}, K_{3,0}^{10}, K_{1,3}^{10}, K_{2,3}^{10})$ constant.
- Thus, we have 2^{35} keybytes to test and reduce by the equation.
- We reduce this now by the above equation, and then combine the remaining 2^5 values for $(K_{0,0}^{10}, K_{1,0}^{10}, K_{3,0}^{10}, K_{1,3}^{10}, K_{2,3}^{10})$ using the remaining 3 equations
- Final size of the key is 2^8 . Note, now the run time is proportional to 2^{35} AES encryptions.

So, therefore, right we can keep the remaining 5 key bytes which are shown here as constant ok. And, then you have got 2 power of 35 key bytes it was remember you have reduced the key size to 2 to the power of 40 values. So, you can split that into 2 parts, in 1 part there are 2 power of 5 values, which actually does matter for this equation. So, you can take 2 power of 35 key values and then just try to solve them by this equations, the list get shortened and it never exceeds 2 power of 35.


So, at a time right you are basically never trying anything more than 2 power of 35 and therefore, right that you are restricting the time complexity which is below 2 power of 35 you do not allow it to exceed beyond 2 power of 35. So; that means, right this attack right would be a at least you know like much faster compared to an attack which actually

is proportional to a time complexity of 2 power of 40 AES encryptions. So, therefore, the final size of the AES key size is still 2 power of 8, but now the run time is proportional to 2 power of 35 AES encryptions.

(Refer Slide Time: 27:56)


Comparison with previous Works

Reference	Fault Model	Number of Faults	Exhaustive Search
Chen & Yen	Single Byte	22 to 44	1
Peacham et. al.	Multi Byte	12	1
Takahashi et.al.	Multi Byte	2	2^{48}
Kim et. al.	Multi Byte	2	2^{32}
Our attack in CARDIS 2011	Multi Byte	1	2^{32}
This Attack	Single Byte	1	2^8



So, here is kind of a summary of the attack. So, you see that the attack requires one faults and reduce the AES key size to 2 power of 8 values, but with 2 power of 35 key conjuring's which are required.

(Refer Slide Time: 28:08)

- DFA on AES Key-schedule vs DFA on AES datapath
- Faults are induced in the Key-schedule.
 - Attacks on Key-Schedule show that a single byte fault, in the AES-128 keyschedule, reduces the AES key size to 2^8 values:
 - This result is analogous to the single byte fault induction in the AES-128 datapath, where also the remaining key size is 2^{32}
 - However the time complexity in this present attack is 2^{35} , while for the datapath it was 2^{30}
- 

So, if you compare the AES key schedule to DFA with the AES data path key DFA, then you can see here that it is like a 2 power of 30, you know like there is a you know 2 power of 35 versus 2 power of 30 complexity, where as you know like the remaining key size is 2 power of 32 with you know like, I mean the result is the analogous to the single byte fault induction in the 128 data path. We are also the remaining keys size is 2 power of 32 and actually you can reduce it further 2 to the power of 8 values as we have seen.

(Refer Slide Time: 28:39)

Reduction Proof for Optimality

- Adv_{DFA}^{state} : Adversary against AES performing DFA on state.
- Adv_{Colin}^{AES} : Classical Adversary on AES.
- Classical adversary searches for plaintexts P and P' such that after a particular round r a target difference ΔS is created. Probability: $Pr(\Delta S)$.
- K_S : Key space of AES wrt. classical cryptanalysis.
- K_P : Key space of AES wrt. DFA
- $K_S \leq K_P \leq Pr(\Delta S)$.

Ali, Mukhopadhyay, Tunstall: Differential Fault Analysis of AES: Towards Reaching its Limits, Cryptology eprint 2012 (ICEN)

So, you can actually so, you have this minute comment on the optimality of the attack. So, here what we try to show here is that imagine that I would like to know whether my fault attack is optimal or not ok. So, what we can try to do is basically we can try to kind of pretty much guess any values of P and P' , and imagine that I am trying to create a differential at this point ok.

So, I am trying with you know like different P and P' and I will try the kind of create a differential at this point, which essentially nothing but the fault differential that I am you know like exploiting in a DFA. So, imagine that there is a DFA fault attacker who is essentially denoted as this adversary of DFA. And, what it is does is basically you know like reduce the key size of AES ok.

And, that essentially is denoted as a K_I which is the key size of AES or key space of the AES with respect to a DFA ok. And, imagine that K_S is the key space of AES with respect to classical cryptanalysis ok. So, remember that to understand the if right I am

able to do a DFA attack ok. And, the DFA attack right essentially reduces the key size to only K_1 values; that means, you know like you are like DFA K complexity is K_1 ok. But, how many times you kind of need to do or choose; how many times you need to basically you know like choose pairs of plain text. So, that this difference is created at this point ok.

So, this is essentially nothing but one by the probability of this ΔS , because I am trying to create a ΔS differential here or maybe you know like a ΔS differential at this point. And that line would mean that if the probability of this ΔS you know is probability of ΔS , then I have to try one by probability ΔS to at least have one case where this differential is created ok.

So, therefore, right if I multiply these two things then I know that my total key complexity of AES cannot be more than this ok. Therefore, I can establish a bound where I can write that K_S is less than or equal to K_1 into 1 by probability of ΔS ok, or in other words I can tell that K_1 is greater than or equal to K_S into probability of ΔS .

(Refer Slide Time: 30:54)

Optimal limit for a byte fault DFA

- Assuming, $K_s = 2^{128}$.
- ΔS : Single Byte difference at the input to the eighth round.
 - $\Pr(\Delta S) = 2^{-120}$.
 - Therefore, $K_s = 2^{-120} 2^{128} = 2^8$.
- This analysis has been found to work for single byte fault attacks on AES-192, 256 and also for multiple byte faults.
- Similar analysis can be also performed for DFA on key-schedules.

Handwritten notes on the slide include the equation $\frac{2^8}{2^{128}} = 2^{-120}$ and a small 3x3 grid diagram with a red 'X' in the top-right cell.

So, what is the implication of this we can easily observe here if you plug-in the values of in the context of AES. For example in K_S in AES right we assume that K_S equals 2 to the power of 128 and we know that suppose you know like you want you basically want different distribution suppose, which is of this nature 1 byte difference ok. So, therefore,

right there are totally 2 to the power of 128 possible values out of which you want a difference of 2 to the power of 8 , because this byte can take any possible value.

And this is equal 2 to the power of minus 120 ; that means, if you try 2 to the power of 120 experiments on a random, you would expect that at the input of that round you probably get one such case, where you have this kind of difference ok. And, that gives you know probability of ΔS and therefore, like you can expect that K_1 is equal to 2 to the power of 8 ok.

And you can if you remember now right that is what exactly what we get in our attack, because exactly we reduce it to 2 to the power 8 and this shows that this is an optimal attack. If you can reduce it further it kind of hints you can do a classical cryptanalysis or classical differential analysis of AES ok, which would be really interesting ok.

(Refer Slide Time: 32:04)

Conclusions

DFA can be applied on AES, even more than one bytes, are affected by the fault.

We discussed a Diagonal Fault attack on AES, where even if the fault affects one of the 4 diagonals of the 8th round input of AES.

Final AES keysize is 2^{32} .

DFA can be performed also on AES targeting the AES Key-Schedule

Analysis shows that a single byte fault in the 8th round key can reduce the key size to only 2^8 values with a time complexity of 2^{35} .

So, right I mean so, at this point right we basically done with this discussion and therefore, right to conclude DFA can be applied on a AES even more than you know like 1 bytes or when more than 1 bytes are affected and that we have discussed in the form what is called as a diagonal fault attack on AES. And, so, we have seen that you know even one of the 4 diagonals of the 8th round is effected then the final AES key size can be 2 to the power 32 with the single fault induction. And DFA can be also performed in AES targeting the AES key schedule and analysis shows that a single byte fault in the 8

round key can reduce the key size to only 2 power of 8 values with the time complexity of 2 power of 35.

So, thank you for your attention.