Project - Advanced Topics in Cybersecurity (650.050)

KLEIN's attack with a reduced number of rounds
(at most 6)

Baptiste CALLARD

28 June 2022

Table des matières

I Introduction

The goal of this project will be to use the knowledge we have learned in class on a topic from a research paper. This will allow us to confront a real problem of an advanced level. We will try to implement an attack thanks to this paper to see that it doesn't work only theoretically but that it is also effective in practice.

My subject for this project is to find a differential attack on KLEIN64 on the maximum of round. For that we will first introduce and describe KLEIN. Then I will describe the properties that were useful to me and the weaknesses that were found on KLEIN. Finally, I will explain how I have done to attack KLEIN64 between 2 and 6 rounds.

Without going into details, KLEIN64 is a 64 bits block cipher. The attack consists in first finding the lower nibbles. Thus, it is possible to find 32 bits with a differential attack. Then, the higher nibbles can be recovered by brute force or by using a trick again, so we can also reduce the complexity of this part. Thus, we will show that it is possible to have an $O(2^{24})$ attack for 5 rounds and $O(2^{30})$ for 6 rounds to fully recover the key.

II Description of KLEIN

KLEIN is a family of lightweight block ciphers presented by Gong, Nikova and Law at RFIDSec2011. The goal was to find an implementation that did not use a lot of memory resources. The version we will study is KLEIN64 which corresponds to 64-bit blocks. The structure is very close to the one of AES. On the other hand, some simplifications are made to reduce the complexity. Some of these simplifications lead to biases that allow today to fully attack KLEIN64. Each round is composed of 4 layers : AddRoundKey, SubNibbles, RotateNibbles and MixNibbles.

To go into more details. The first step AddRoundKey is to Xored the RoundKey to the current state. This is the same step seen in most blockciphers.

Then, the SubNibbles step aims at introducing nonlinearity in the cipher. This method uses a SBOX which is a bijective permutation of the nibbles. This operation operates on 16 parts of 4 bits :
$c^i = c^i_0||c^i_1||c^i_2||c^i_3||c^i_4||...||c^i_{14}||c^i_{15}$.Ainsi $S(c^i) = S(c^i_0)||S(c^i_1)||S(c^i_2)||S(c^i_3)||S(c^i_4)||...||S(c^i_{14})||S(c^i_{15})$
We always use the same SBOX no matter what round.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S[x]	7	4	a	9	1	f	b	0	c	3	2	6	8	e	d	5

Figure 1 – KLEIN SBOX

Then there is the part that deals with the diffusion of the differences and the mixing of the nibbles. The RotateNibbles rotates the state of two bytes on the left and the inverse RotateNibbles of two bytes on the right.

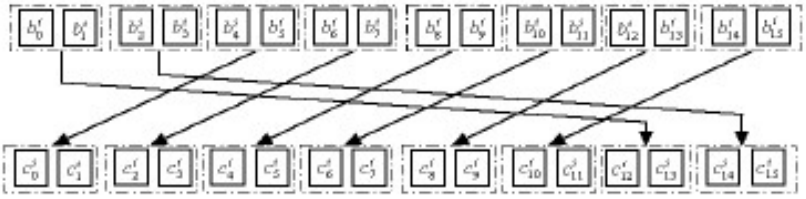


FIGURE 2 – RotateNibbles

Finally, for the MixNibbles, we divide the state into 8 parts of 8 bits :

$$c^i = c_0^i||c_1^i, c_2^i||c_3^i, c_4^i||c_5^i, c_6^i||c_7^i, c_8^i||c_9^i||c_{10}^i||c_{11}^i, c_{12}^i||c_{13}^i, c_{14}^i||c_{15}^i$$

Then, we apply the AES Mixcolumn matrix to the vector : $[c_0^i||c_1^i, c_2^i||c_3^i, c_4^i||c_5^i, c_6^i||c_7^i]^t$ puis $[c_8^i||c_9^i, c_{10}^i||c_{11}^i, c_{12}^i||c_{13}^i, c_{14}^i||c_{15}^i]^t$

As we operate separately on the two parts of texts we will note as MixNibbles the step with both matrices. When we refer to only one part then we will also use MixColumn.

$$\begin{bmatrix} s_0^{i+1}||s_1^{i+1} \\ s_2^{i+1}||s_3^{i+1} \\ s_4^{i+1}||s_5^{i+1} \\ s_6^{i+1}||s_7^{i+1} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} c_0^i||c_1^i \\ c_2^i||c_3^i \\ c_4^i||c_5^i \\ c_6^i||c_7^i \end{bmatrix}, \begin{bmatrix} s_8^{i+1}||s_9^{i+1} \\ s_{10}^{i+1}||s_{11}^{i+1} \\ s_{12}^{i+1}||s_{13}^{i+1} \\ s_{14}^{i+1}||s_{15}^{i+1} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} c_8^i||c_9^i \\ c_{10}^i||c_{11}^i \\ c_{12}^i||c_{13}^i \\ c_{14}^i||c_{15}^i \end{bmatrix}$$

FIGURE 3 – MixNibbles

Finally, we will explain the key scheduling to finish this quick description of KLEIN. The process is described with the diagram below. Non-linearity is added thanks to two Sboxes. Moreover, the round number is added to each round so that the key scheduling depends on the round.

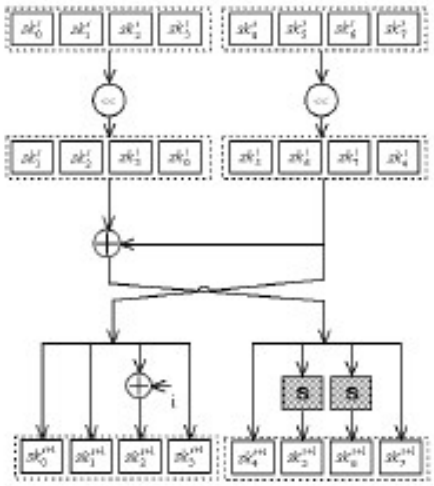


FIGURE 4 – keyScheduling

III Attaque de KLEIN64

III.1 Observation and weakness

The attack takes advantage of the quasi-independence between the higher and lower nibbles. This is used to find and build truncated differential paths. It can also be used to attack independently subparts of the ciper. The attack relies mainly on MixNibbles. Indeed, it has been shown that all encryption steps except MixNibbles do not mix higher and lower nibbles. The following proposals are fundamental to the attack I have implemented :

Proposition 1 if the eight nibbles entering **MixColumn** are of the form **0X0X0X0X**, where the wild-card X represents any 4-bit value, then the output is of the same form if and only if the MSB of the 4 lower nibbles all have the same value. This case occurs with probability 2^{-3}. This is 2^{-6} if we want this event to occur for both MixColumns at the same time.

This first proposal will be used to attack 1 and 2 rounds. However, two characteristics with a higher probability can be used to attack 3 to 6 rounds.

Proposition 2 if the eight nibbles entering **MixColumn** are of the form **0X0X0000**, where the wild-card X represents any 4-bit value, then the output is of the same form if and only if the MSB of the 4 lower nibbles all have the same value. This case occurs with probability slightly better than 2^{-2}.

Proposition 3 if the eight nibbles entering **MixColumn** are of the form **00000X0X**, where the wild-card X represents any 4-bit value, then the output is of the same form if and only if the MSB of the 4 lower nibbles all have the same value. This case occurs with probability slightly better than 2^{-2}.

Observation 1 In the first round, the nibbles (1,2,3,4,13,14,15,16) and (5,6,7,8,9,10,11,12) do not interact. Indeed, the first set (1,2,3,4,13,14,15,16) goes after the ShiftNibbles in the second MixColumn and the second set in the first MixColumns.

Finally, thanks to the report we also have that :

Observation 2 If the difference entering MixColumn is of the form 0X0X0X0X where the wildcard X represents a difference in (0,...,7) or respectively (8,...,f) then the output difference is of the form 0Y0Y0Y0Y, where Y represents a possibly null difference.Furthermore, the average number of non-zero Y's is 3.75, as one can experi- mentally verify.

III.2 Attacking KLEIN

Thus, thanks to these propositions we can construct the following truncated differential paths. Using the first proposition we obtain :

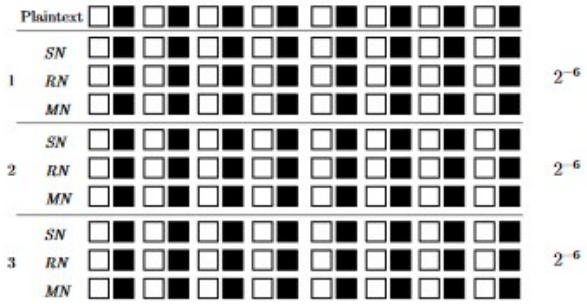
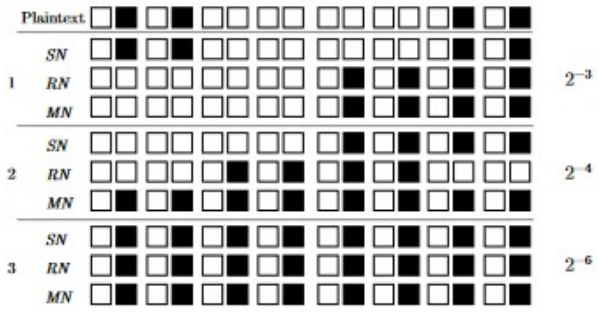
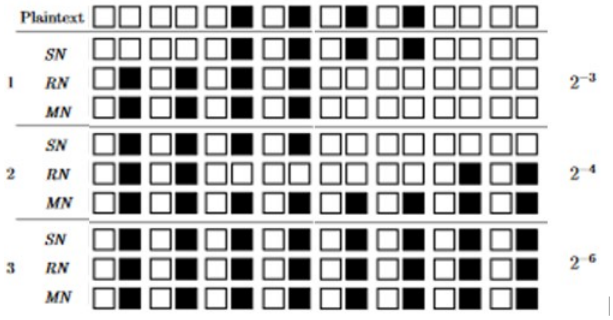


FIGURE 5 – truncated differential paths 1

Then, using propositions 2 and 3 :



(a) truncated differential paths 2



(b) truncated differential paths 3

FIGURE 6

Here we have only the beginning of the truncated differential paths. The first truncated differential continues with the same schema for the number of rounds we want. Then path 2 and 3 we continue the truncated differential with the scheme of the first one to have more than 3 rounds.

Probability analysis

Our attack uses truncated differential defined as a collection of characteristics. The goal is to use the independence between higher and lower nibbles. Thus, our collection of characteristics wants to keep the higher nibbles inactive. A sufficient condition is that the first 4 and the next 4 lower nibbles are all a difference in {0,...,7} or all in {8,...,f}. This comes from the observation showed previously.

- truncated differential paths 1

In all rounds, we want all higer nibbles to remain inactive. This happens with a probability of p $\approx 2^{-6}$

Indeed, there are on average 3.75 active nibbles coming from each MixColumn of the previous round. It is a property given in the paper. The first case is that all four lower nibbles are active with probability $\frac{15}{16}$. This is necessary to obtain four active nibbles with differences in (8,...f) after SubNibbles, but not to obtain four active nibbles with differences in (0...7). The probability to obtain one the desired sets of differences in one half of the state is :

$$(\frac{7}{15})^{3.75} + (\frac{15}{16}).(\frac{8}{15})^4 \approx 2^{-3}$$

Recall that a difference in (0,...,7) is reached with probability $\frac{7}{15}$, and one in (8,...,f) with probability $\frac{8}{15}$. As the halves of MixNibbles behave similarly :

$$p \approx ((\frac{7}{15})^{3.75} + (\frac{15}{16}).(\frac{8}{15})^4)^2 = 2^{-5.82} \approx 2^{-6}$$

To explain in detail this probability :
o $\frac{7}{15}$ is the probability to have a difference between 0 and 7. If it is the case then we will have on average 3.75 Y non zero after the mixcolumn in this difference 0Y0Y0Y0Y. If Y is zero it is fine also.

o $\frac{15}{16}\cdot \frac{8}{15}^4$: $\frac{15}{16}$ the difference is not zero and we have a difference in (8,...,f) for the 4 nibbles so $\frac{8}{15}^4$. We must have both thus we have a product.

We can append this two probability as they represent the event we want.

Finally, as we want this to happen at the same time for both mixcolumns, the probability is to the power of 2.

- truncated differential paths 2 and 3
In this case, for the first the probability is slightly better than 2^{-3}. Indeed, we impose this condition only for a part of the message so $(\frac{7}{15})^{3.75} + \frac{15}{16}.(\frac{8}{15})^4$

In the case of the 2nd round, we have both MixColumn active but with only half the message, the probability of this round is also slightly p = 2^{-4}. We have a probability of $2^{-3}.2 = 2^{-2}$ for each Mixclomun because the probability is twice as high as having all 4 for a Mixcolumns and so $(2^{-2})^2 = 2^{-4}$ for having both conditions at the same time. So for these two rounds, we have a probability of about $2^{-3}.2^{-4} = 2^{-7}$.

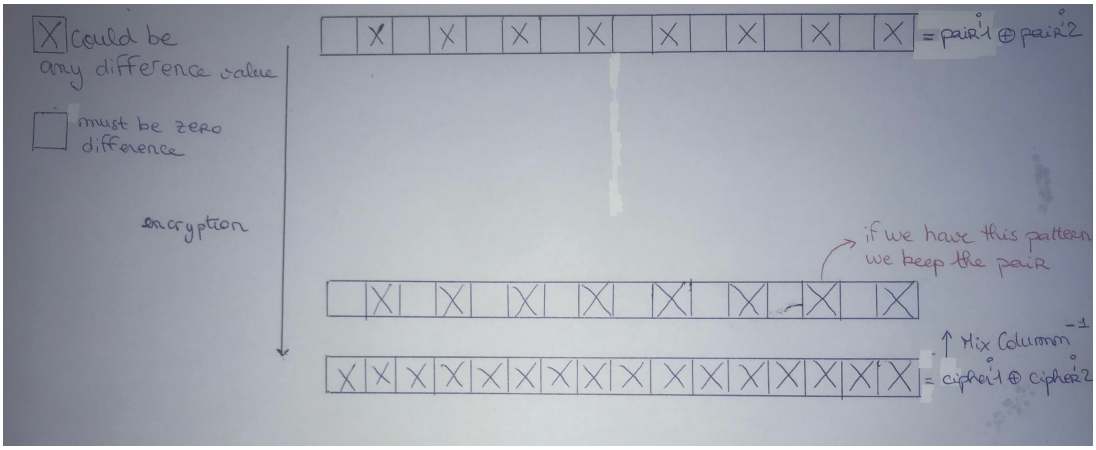
III.3 Attack over 2 rounds

We will use the truncated differential paths 1 (Figure 5). Then, we will try to find pairs that are conform to this truncated differential paths. Then, we will be able to guess the master key by making our guess on the first RoundKey. By using several pairs satisfying this truncated differential paths we can further reduce the space of possible keys. When we have more than ten pairs satisfying the truncated differential then we reduce almost systematically the space to a

single key.

The attack proceeds as follows :

- find a set of pairs satisfying the characteristic :



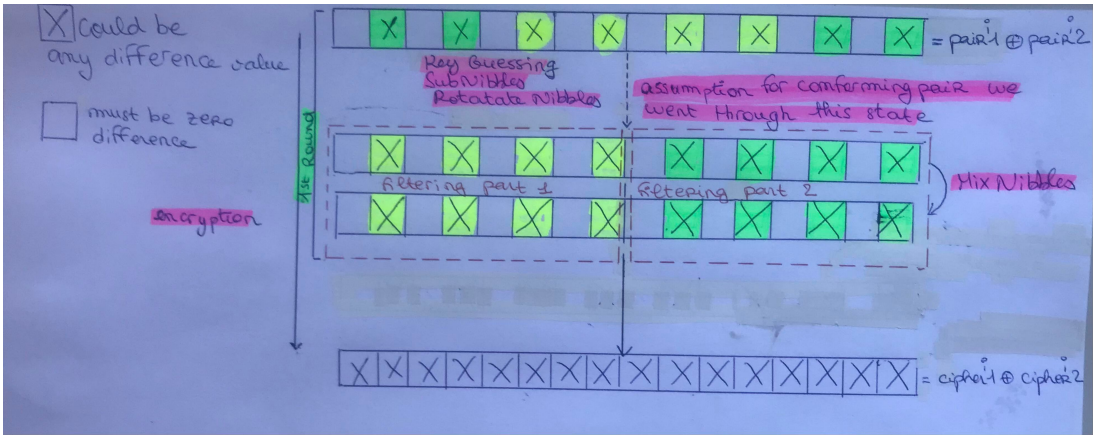
(a) find conforming pairs

FIGURE 7

For this we take two pairs with the following difference 0X0X0X0X 0X0X0X0X. In a second time, we calculate the corresponding cipher texts and we don't care about the difference at this step. Then, as the AddRoundKey and MixNibbles are linear then we can easily have the difference before the MixNibbles. If the difference has also the difference 0X0X0X0X 0X0X0X0X then we keep the pair as a pair satisfying the characteristic. Once we have collected the number of pairs satisfying the characteristic then we can start guessing the key. It is also important to note that we can't use the neutral bits with this method. Indeed, we want to have different values at the beginning for the plaintexts with the same differences.

The probability of obtaining a conforming pair is $p = 2^{-6}$ for 2 rounds. Indeed, we must consider a differential for n-1 round to attack n rounds. As we want about 2^4 conforming pairs then we need to test with $n = 2^{10}$ for 2 round.

- Guess of the first Round Key :



(a) key filtering

FIGURE 8

We will repeat this part for all conforming pairs. Knowing that we had a difference 0X0X0X0X 0X0X0X0X before the last MixNibble for these pairs then we have a great chance to have exactly followed the differential (figure 5) and that the difference was also 0X0X0X0X 0X0X0X0X after the first MixNibble. Thus, using Proposition 1 we can reduce the key space by checking after adding a candidate key, SN, RN and MN if we still have the difference 0X0X0X0X 0X0X0X0X. As the part with nibbles in yellow (higher and lower so 32 bits) and the next 32 bits are independent from the MixNibbles, we can make our guess on 16 bits independently. We can still optimize, for that we guess at the same time on the two parts of 16 bits to avoid making the double of encryption. In the first round there is no interaction between the two parts where we guess before the MixNibbles. This allows in some way to parallelize the calculation. If one actually had the difference 0X0X0X0X with the real key then one must reduce the space by $2^{16}/2^3 = 2^{13}$. This comes from Proposition 1 given earlier ($p = \frac{1}{2^3}$). We can filter on the two MixNibbles we can finally reduce from 2^{32} to $2^{32}/2^6 = 2^{26}$. We repeat this with all conforming pairs and it takes about ten conforming pairs to reduce the space to one key by taking the key that appears the most times. Indeed, it is possible that a conforming pair did not have this difference after the first mixNibble, so we take the max and not the intersection.

Study of complexity

For both attacks the complexity in time is memory is as follows if we don't consider the trick for the higher nibbles [1] :

Time : $O(2^{32})$ because we have to make the brute force of higher nibbles

Memory : $O(2^{16})$ because we need to store the number of times the keys pass the filtering. We can attack the two MixColumns separately so we have to store at most 2^{16}. If we attack in the same time then we have to store $O(2^{17})$ but it won't make much difference. For brute force the complexity is constant because we do not store the keys.

As we focused our attack on the lower nibbles we can also look at the time complexity of this part. The memory complexity is the same.

1. It is possible to reduce the higher nibbles attack to about $O(2^{25})$. I didn't have time to implement it but I described in detail how to do it at the end of the report because it was beyond project expectations

We recall that, to attack n rounds, we have to consider a characteristic on n-1 round.

For 2 rounds, lower nibbles part :

Here we have $p = 2^{-6}$ to follow the characteristic over 2 rounds so as we want about 2^4 then we will consider $n = 2^{10}$ pairs. So, for the time complexity we try the 2^{16} keys with the 2^4 conforming pairs. So the time complexity is 2^{21} because we have to do this twice but as we do it only for first round and not the fully cipher we divide it by the number of round (2) so it remains 2^{20}.

($O(2^{17})$) . In this case (Time is $O(2^{19})$).

Time is $O(2^{20})$
Data is $O(2^{10})$

Note : If we attack in parallel the lower nibbles then we divide by two the time complexity but we increase by two the memory complexity

Results

```
#####  RESULTS FOR LOWER NIBBLES #############
Time for finding the set of conforming pairs is : 0.000000 ms
Time for recovering candidate keys part 1 is : 330.000000 ms
Time for recovering candidate keys part 2 is : 314.000000 ms

**************************************
    Start brute force higher nibbles
**************************************

**************************************
We try with the candidate 1 for lower nibble
**************************************
2 f     f f     9 3     f f
**************************************
        True Key
**************************************
2 f     f f     c9 a3     7f df
**************************************
     recovered Key
**************************************
2 f     f f     c9 a3     7f df
```

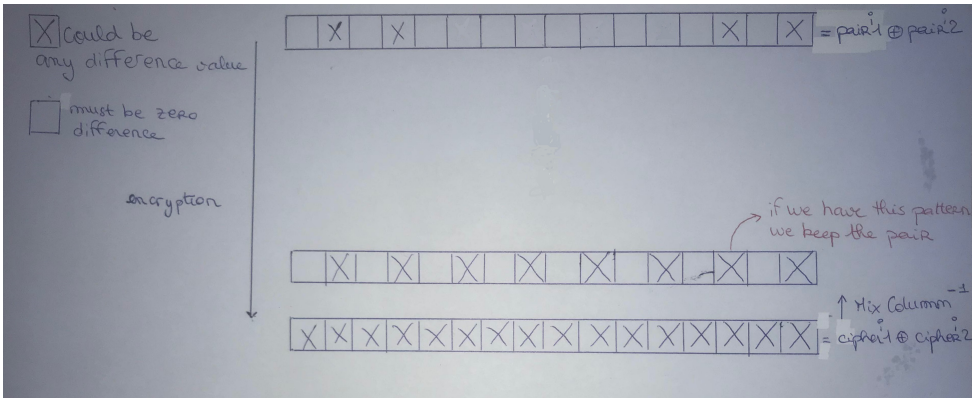
(a) Results for 2 rounds

FIGURE 9

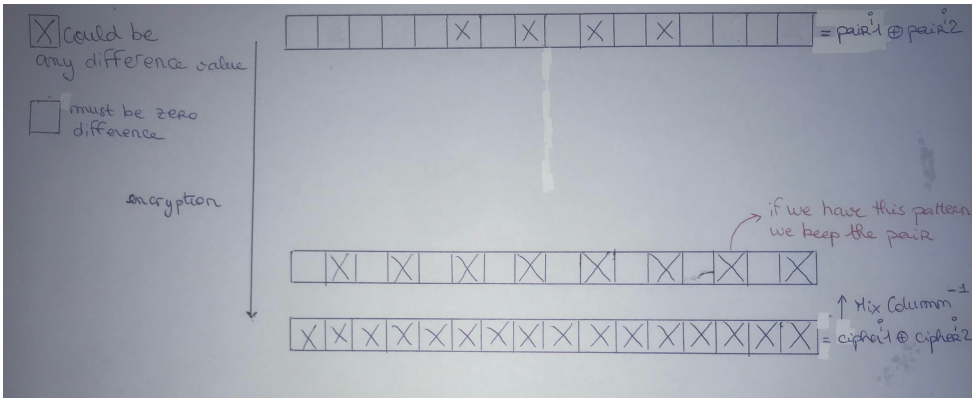
We can verify this complexity experimentally. Here, we have only one iteration, so we should do an average to estimate the complexity correctly. This is consistent with our explanation that what takes the most time is the recovery part where we test the 2^{16} key possibilities.

III.4 Attack over 3 to 6 rounds

We can see that by using this characteristic then the probability of following it becomes very high when we increase the number of rounds. With this method I could find up to 4 rounds and rarely 5 rounds but it took too much time. To reduce the complexity in time and memory, I tried in a second time to find a new attack. I thought that I was not obliged to use the same characteristic to find the first 16 bits (1st MixNibbles) and the 16 other bits. So, I found two other characteristics that allow to find independently these 16 bits with a better probability. So to find the nibbles 2, 4, 14 and 16 I will use the truncated differential paths 2 and to find the nibbles 6, 8, 10 and 12 I will use the truncated differential paths 3. With these two differentials, on the first two rounds we have p = 2^{-7} compared to p = 2^{-12} previously.



(a) find conforming pair path 2



(b) find conforming pair path 3

FIGURE 10

The attack is almost the same as the one explained before. However, based on figure 8, we are again interested in a separate way in the yellow nibbles and in the green nibbles but we have to consider them with all the conforming pairs which have active nibbles where we do our filtering. Thus, we start with a difference 0X0X0000 00000X0X in the case of path 2 and 00000X0X 0X0X0000 with path 3. In the same way, we see if before the last MixNibble we have a difference 0X0X0X 0X0X0X. We keep all the pairs that check this for both paths so we have two sets of conforming pairs. We try to have about ten for each. Then, we make our key guess on 0X0X0000 00000X0X then we check that we had the difference 0X0X0X0X before the MixNibble. We can reduce with a set of a dozen conforming pairs to 1 key. If we still have several values then we have the choice to search again for conforming pairs or we can do brute force with all key candidates. The same procedure is done on 00000X0X 0X0X0000 and we also look if we have the difference

0X0X0X0X before the last MixNibble. For this difference to exist, we must attack at least 3 rounds. That's why I differentiated 2 rounds. To some extent by checking on specific nibbles one can use these differentials for 2 rounds. But the most interesting thing is to attack as many rounds as possible.

Then we can make our guess independently and use the same filtering as before (based on figure 8).

Study of complexity

For both attacks the complexity in time is memory is the same for the entire attack :

Time : $O(2^{32})$ because we have to make the brute force of higher nibbles (or $O(2^{24})$ is we optimize this part)

Memory : $O(2^{16})$ because we need to store the number of times the keys pass the filtering. We can attack the two MixColumns separately so we have to store at most 2^{16}

Once again, we can also look at the time complexity of the lower nibbles part. The memory complexity is the same for this part.

To attack n rounds, we have to consider a characteristic on n-1 round.

for 3 rounds :

Here we have $p = 2^{-(3+4)}$ to track the characteristic, as we want about 2^4 then we will consider $n = 2^{11}$ pairs. The time complexity corresponds to testing twice the 2^{16} keys with the 2^4 conforming pairs and we can divide by 3 as we are only encrypte for the first round.

Time is $O(2^{20})$
Data is $O(2^{11})$
Memory : $O(2^{16})$

for 4 rounds :

Here we have $p = 2^{-(3+4+6)}$ to follow the characteristic, so as we want about 2^4 then we will consider $n = 2^{17}$ pairs and we divide by 4 because again we encrypte only the first round.
Time is $O(2^{19})$
Data is $O(2^{17})$
Memory : $O(2^{16})$

for 5 rounds :

Here we have $p = 2^{-(3+4+6+6)}$ to track the characteristic, so as we want about 2^4 then we will consider $n = 2^{23}$ pairs. So the time complexity of this part will be higher than for key recovering and so the time complexity becomes $O(2^{24})$ because we have to do this twice. It is the double of the data complexity.
Time is $O(2^{24})$
Data is $O(2^{23})$

Memory : $O(2^{16})$

Pour 6 rounds :

Here we have $p = 2^{-(3+4+6+6+6)}$ to follow the characteristic, so as we want about 2^4 then we will consider $n = 2^{29}$ pairs because we have to do this twice.

Time est $O(2^{30})$
Data est $O(2^{29})$
Memory : $O(2^{16})$

Results

for 3 rounds

```
#####  RESULTS FOR LOWER NIBBLES #############
Time for finding the first set of conforming pairs is : 0.000000 ms
Time for recovering candidate keys part 1 is : 220.000000 ms
Time for finding the second set of conforming pairs is : 15.000000 ms
Time for recovering candidate keys part 2 is : 252.000000 ms

**************************************
    Start brute force higher nibbles
**************************************

**************************************
We try with the candidate 1 for lower nibble
**************************************
2 f     f f     9 3     f f
**************************************
        True Key
**************************************
2 f     f f     c9 a3     7f df
**************************************
     recovered Key
**************************************
2 f     f f     c9 a3     7f df
```

(a) Results for 3 rounds

FIGURE 11

We see that it is always the attack of the lower nibbles that takes the most time. This is the result we were waiting for.

Pour 4 rounds

```
#####  RESULTS FOR LOWER NIBBLES #############
Time for finding the first set of conforming pairs is : 220.000000 ms
Time for recovering candidate keys part 1 is : 298.000000 ms
Time for finding the second set of conforming pairs is : 408.000000 ms
Time for recovering candidate keys part 2 is : 329.000000 ms

**************************************
    Start brute force higher nibbles
**************************************

**************************************
We try with the candidate 1 for lower nibble
**************************************
2 f     f f     9 3     f f
**************************************
        True Key
**************************************
2 f     f f     c9 a3     7f df
**************************************
     recovered Key
**************************************
2 f     f f     c9 a3     7f df
```

(a) Results for 4 rounds

FIGURE 12

We still expect to take longer to do the key recovering than to find the conforming pairs. However, we can see that it is not as clear as before. This shows that we are using probabilities.

Pour 5 rounds

```
#####  RESULTS FOR LOWER NIBBLES #############
Time for finding the first set of conforming pairs is : 29212.000000 ms
Time for recovering candidate keys part 1 is : 235.000000 ms
Time for finding the second set of conforming pairs is : 24322.000000 ms
Time for recovering candidate keys part 2 is : 350.000000 ms

**************************************
    Start brute force higher nibbles
**************************************

**************************************
We try with the candidate 1 for lower nibble
**************************************
2 f     f f     9 3     f f
**************************************
        True Key
**************************************
2 f     f f     c9 a3     7f df
**************************************
     recovered Key
**************************************
2 f     f f     c9 a3     7f df
```

(a) Results for 5 rounds

FIGURE 13

This time, the part that takes the most time is to find the conforming pairs. This is what we

observe experimentally.

Pour 6 rounds

```
#####  RESULTS FOR LOWER NIBBLES #############
Time for finding the first set of conforming pairs is : 919660.000000 ms
Time for recovering candidate keys part 1 is : 307.000000 ms
Time for finding the second set of conforming pairs is : 1241162.000000 ms
Time for recovering candidate keys part 2 is : 330.000000 ms

**************************************
    Start brute force higher nibbles
**************************************

**************************************
We try with the candidate 1 for lower nibble
**************************************
2 f     f f     9 3     f f
**************************************
        True Key
**************************************
2 f     f f     c9 a3     7f df
**************************************
     recovered Key
**************************************
2 f     f f     c9 a3     7f df
```

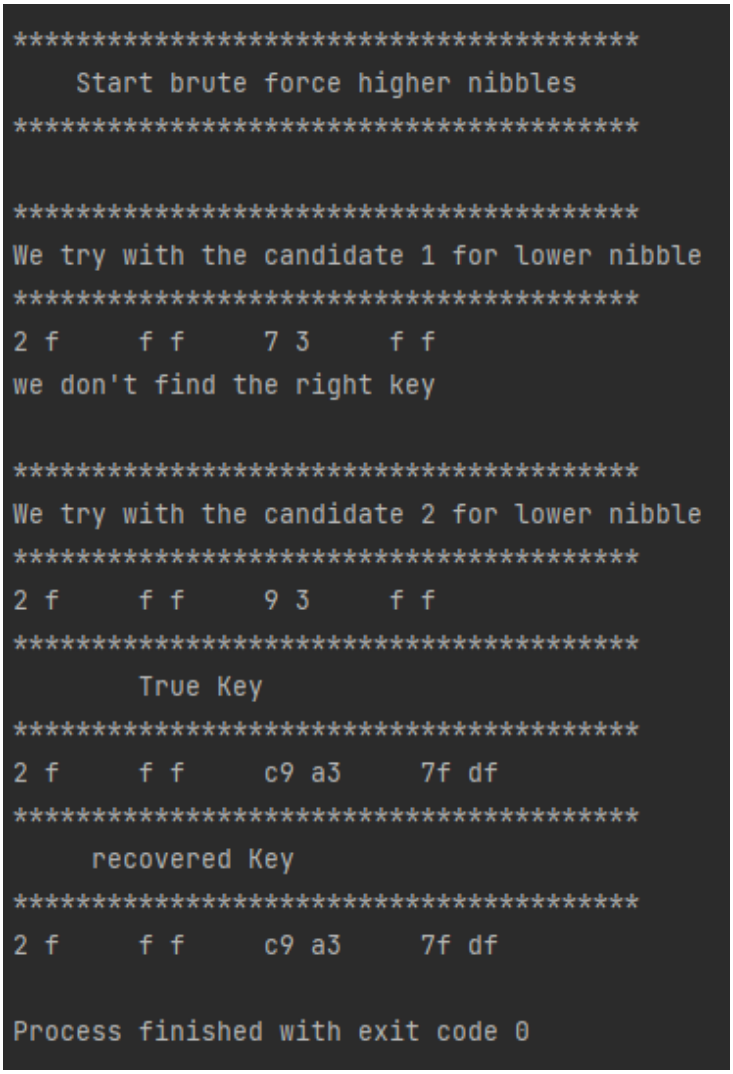
(a) Results for 6 rounds

FIGURE 14

It is possible to recover for 6 rounds with a complexity lower than the brute force. The attack remains feasible with my computer which is not very powerful. With more powerful machines one can imagine that it takes even less time.

Two candidates

If more than two candidates pass the filtering for the lower nibbles part then we do brute force force both. In fact, it appends almost never I reduce intentionally the number od try to see it.



(a) If more than one candidate for lower nibbles

FIGURE 15

IV Improvement regarding higher nibbles attack

To take the project further, I wanted to find a technique to reduce brute force. This was not explained in the paper. For the moment, in the research paper it is suggested to brute force the higher nibbles. The complexity of brute force 32 higher nibbles dominates the complexity for attacks ranging from 2 to 6 rounds because it is $O(2^{32})$. Thus, this is the part that prevents us from speeding up the process. Even if we consider fewer rounds.

I didn't have time to implement this part because of a time constraint because I had a lot of projects at the end of the year and my internship starts on July 1st.

However, I had an idea. For this, I based myself on a filtering condition from a research paper I found on the internet.

We have seen that the higher nibbles and lower were independent in all steps except the Mix-Nibbles. But even in this part if we look carefully then we notice that they are in fact very little

related. Indeed, we can use the decomposition of the following values a,b,c and d in bytes :

$$a = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$$

with a_0 the MSB and a_7 the LSB.
Proposition

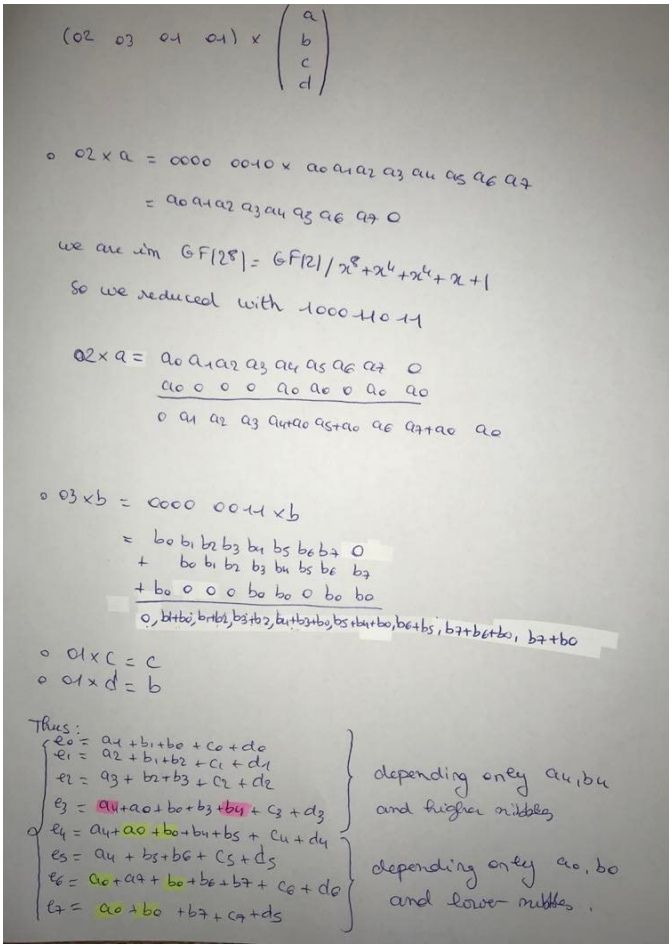
The value of the highers nibbles after the MixColumns depends only on the highers nibbles before the MixColumns and three quantities from the lower nibbles which are :

- $a_4 + b_4$
- $b_4 + c_4$
- $c_4 + d_4$
- $a_4 + d_4$ mais comme $a_4 + d_4 = a_4 + b_4 + b_4 + c_4 + c_4 + d_4$ this fourth value is known with the other three.

This can be shown quickly :

$$\begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

I have detailed the calculations on the following sheet.



(a) Example

FIGURE 16

Thus, we can use the equations explained above which imply the values :

- $a_4 + b_4$
- $b_4 + c_4$
- $c_4 + d_4$

This will reduce the search space by 2^6 for each of the candidates left over from the previous step where lower nibbles were guessed.

We can assume that we have only one candidate left for the lower nibbles step. Indeed, in most cases there is only one candidate left for the lower nibbles. So, here is how we can proceed to reduce the space.

The biggest gain is in the first round. As before, we can attack the two parts independently (in parallel). Thus, we start by adding the 16 higher nibbles to the 32 middle bits in the first round. The values then go into the first MixColumn. We can check if the values are conform on the 3 corresponding information bits. We can do the same on the 16 other higher nibbles at the same time as there is no mixing. Thus, we need 2^{16} encryption. This allows us to reduce the key space by 6 bits in $O(2^{16})$.

Then we test among the remaining 2^{26} candidate keys which ones verify the condition on the 3 information in the 2nd round. We win by 6 bits. By iterating the process, we can reduce the number of candidates for the higher nibble of 6 bits in each round. Thus, the time complexity $(2^{16} + 2^{26} + 2^{20} + 2^{14} + 2^8 + 2^2).\frac{1}{6} \approx 2^{24}$ for 6 rounds ($\frac{1}{6}$ because we don't fully encrypte the data but only one round). For example, for 4 rounds we also have $(2^{16} + 2^{26} + 2^{20} + 2^{14}).\frac{1}{4} \approx 2^{24}$ but 2^{25} for 2 rounds.

Thus, this method allows to decrease the brute force and thus to decrease the time complexity.

This method is not complicated to use. However, it requires a lot of thought. It is necessary to make a preliminary study by hand to detect this type of weakness.

Complexity

Ainsi en utilisant cette nouvelle attaque la complexité totale en incluant les higher nibbles :

for 5 rounds :

Time is $O(2^{24})$ this corresponds to the brute force of higher nibbles with the trick.
Data is $O(2^{23})$ same as before
Memory : $O(2^{16})$ considering that in the first round we store the keys that don't pass $O(2^6)$ instead of $O(2^{26})$. We use the complementary space. Similarly in the second round so we store $O(2^{12})$ after the third round we can store the $O(2^{18})$ that don't pass and after the 4th we store the keys that do pass so $O(2^{14})$. So we are still bellow 2^{16}

for 6 rounds :

Time is $O(2^{30})$ it's not the complexity of brute force but the complexity of finding conforming pairs
Data is $O(2^{29})$
Memory : $O(2^{16})$ using the same technique as above

V Conclusion

Thus, I found a variant of the attack proposed in the research paper. On the other hand, the properties that have been used are from this paper. With the method that was proposed in this project we can find the lowers nibbles up to 6 rounds with a complexity that is below the brute force of higher nibbles. If I had more time I could have tried to reduce the brute force part on the higher nibbles. I really enjoyed this project. As I explained to you, I worked a lot to understand all these concepts since the beginning of the semester. On the other hand, I managed to find an attack that is not explained on the papers (even if it uses results). This shows me that my work has allowed me to understand a lot of things that are concretized in this project.
.