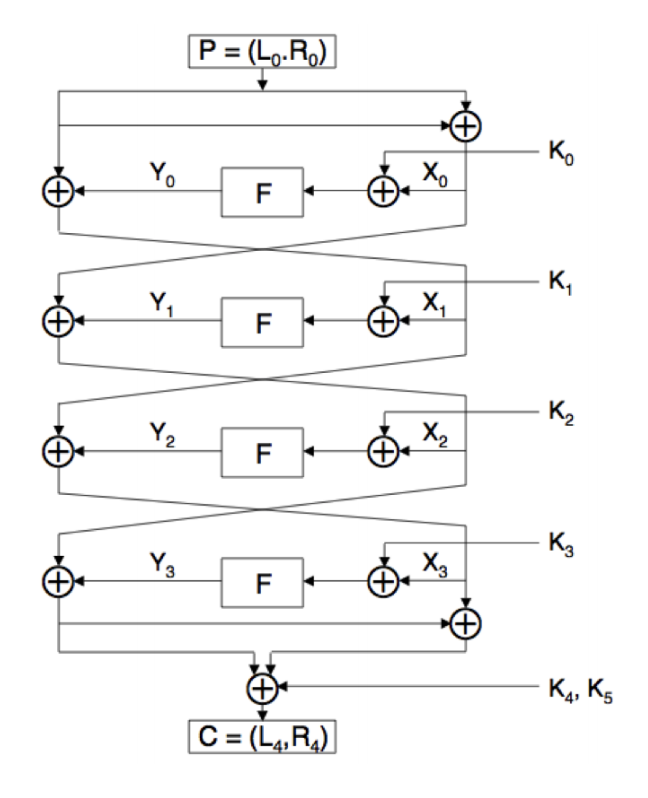
**Name:** Ruben Vasconcelos

**Student Number:** 16212630

**Module Code:** CA642

**Assignment Title:** Linear Crypto Analysis of FEAL 4



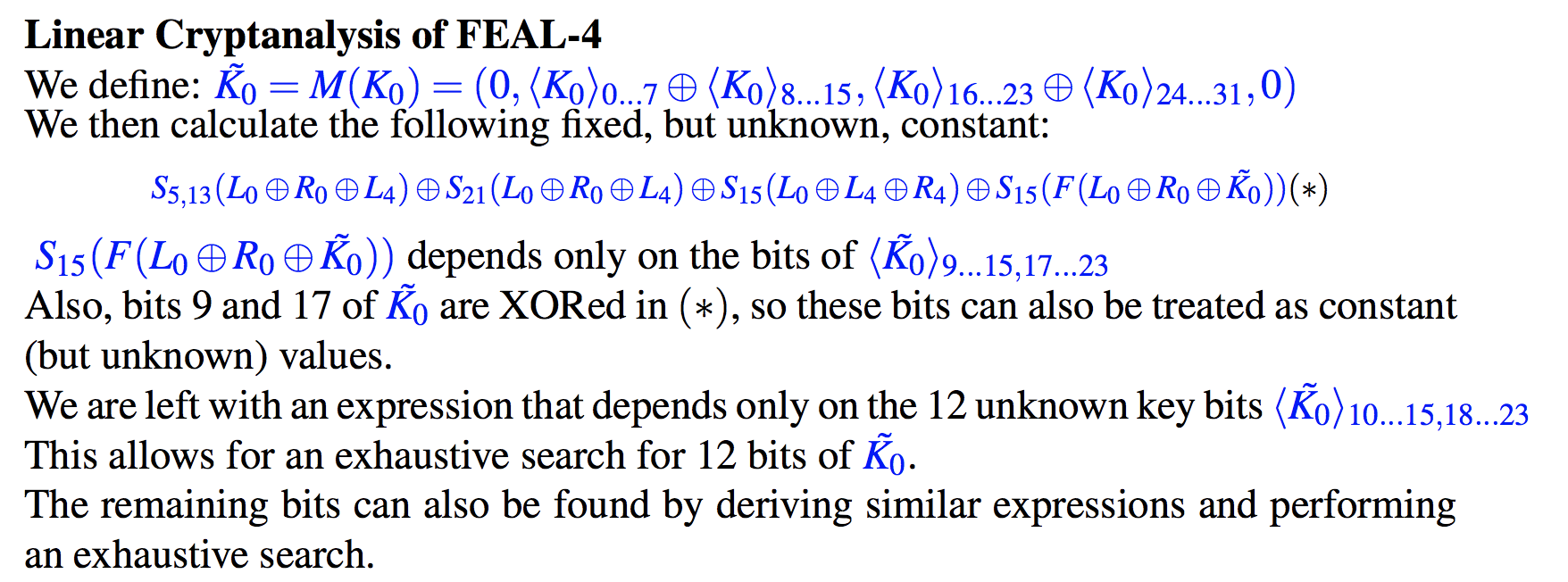
Figure

From figure 1 one we can spot to obvious relations, which we can use to derive a number of constants for the linear attack of FEAL 4:

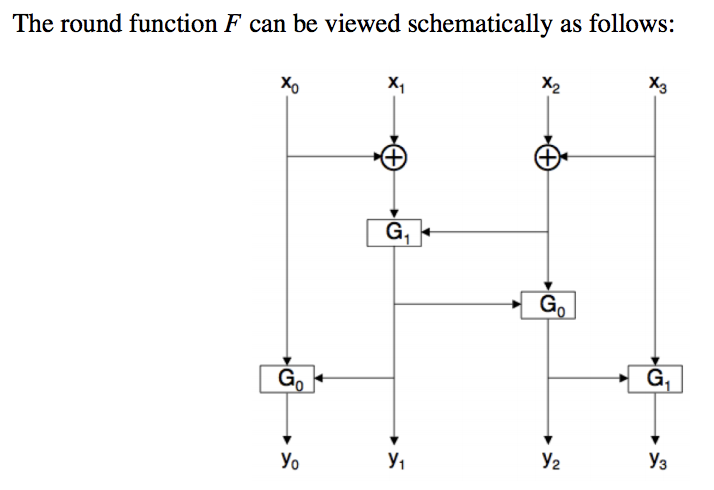
1: X0 ⊕ Y1 ⊕ Y3 = K4 ⊕ L4

2: L0 ⊕ Y0 ⊕ Y2 ⊕ L4 ⊕ K4 = K5 ⊕ R4

And from the lecture notes we know that we can search 12 bit first rather than searching all 32 which takes a lot longer.



Figure



Figure

By mapping through the inputs out outputs of the function F, we can prove the following:

|  |  |
| --- | --- |
| Figure | Move Y terms to the left and group X’s:  S13(Y) = S7,15,23,31(X) ⊕ 1  S5,15(Y) = S7(X)  S15,21(Y) = S23,31(X)  S23,29(Y) = S31(X) ⊕ 1 |

We can use this in addition to the relations from earlier on to the derive the required constants.

**Attack K0:**

The information on the notes including the one in figure 2 would be enough to perform the attack on k0, so if you’re not interested in reading the same thing twice skip to attack on K1.

Using the relation L4 = X0 ⊕ Y1 ⊕ Y3 ⊕ K4 we derive the following:

**S23,29(L4) = S23,29(X0) ⊕ S23,29(Y1) ⊕ S23,29(Y3) ⊕ S23,29(K4)**

S23,29(X0) = S23,29(L0 ⊕ R0)

S23,29(Y1) = S23,29 **F**(X1 ⊕ K1)

= S23,29 **F**(L0 ⊕ Y0 ⊕ K1)

= S31(K1) ⊕ S31(Y0) ⊕ S31(L0) ⊕ 1

S31(Y0) = S31 **F**(X0 ⊕ K0)

= S31 **F**(L0 ⊕ R0 ⊕ K0)

S23,29(Y3) = S31(X3 ⊕ K3) ⊕ 1

= S31(L4 ⊕ K4 ⊕ R4 ⊕ K5 ⊕ K3) ⊕ 1

= S31(L4 ⊕ R4) ⊕ S31(K4 ⊕ K5 ⊕ K3) ⊕ 1

S23,29(L4) = S23,29(L0 ⊕ R0) ⊕ S31(K1) ⊕ S31 **F**(L0 ⊕ R0 ⊕ K0) ⊕ S31(L0) ⊕ 1 ⊕ S31(L4 ⊕ R4) ⊕ S31(K4 ⊕ K5 ⊕ K3) ⊕ 1 ⊕ S23,29(K4)

= S23,29(L4) ⊕ S23,29(L0 ⊕ R0) ⊕ S31 **F**(L0 ⊕ R0 ⊕ K0) ⊕ S31(L0) ⊕ 1 ⊕ S31(L4 ⊕ R4) ⊕ 1

**= S23,29(L0 ⊕ R0 ⊕ L4) ⊕ S31(L4 ⊕ R4 ⊕ L0) ⊕ S31 F(L0 ⊕ R0 ⊕ K0)**

**S13(L4) = S13(X0) ⊕ S13(Y1) ⊕ S13(Y3) ⊕ S13(K4)**

S13(X0) = S13(L0 ⊕ R0)

S13(Y3) = S7,15,23,31(X) ⊕ 1

= S7,15,23,31(L4 ⊕ K4 ⊕ R4 ⊕ K5 ⊕ K3) ⊕ 1

= S7,15,23,31(L4 ⊕ R4) ⊕ S7,15,23,31(K4 ⊕ K5 ⊕ K3) ⊕ 1

S13(Y1) = S13 **F**(X1 ⊕ K1)

= S13 **F**(L0 ⊕ Y0 ⊕ K1)

= S7,15,23,31(K1) ⊕ S7,15,23,31(Y0) ⊕ S7,15,23,31(L0) ⊕ 1

S7,15,23,31(Y0) = S7,15,23,31 **F**(X0 ⊕ K0)

= S7,15,23,31 **F**(L0 ⊕ R0 ⊕ K0)

**= S13(L0 ⊕R0 ⊕L4) ⊕ S7,15,23,31(L0 ⊕L4 ⊕R4) ⊕ S7,15,23,31 F(L0 ⊕R0 ⊕K0)**

**S5,15(L4) = S5,15 (X0) ⊕ S5,15(Y1) ⊕ S5,15(Y3) ⊕ S5,15(K4)**

S5,15(X0) = S5,15(L0 ⊕ R0)

S5,15(Y3) = S7(X) ⊕ 1

= S7(L4 ⊕ K4 ⊕ R4 ⊕ K5 ⊕ K3) ⊕ 1

= S7(L4 ⊕ R4) ⊕ S7(K4 ⊕ K5 ⊕ K3) ⊕ 1

S5,15(Y1) = S5,15 **F**(X1 ⊕ K1)

= S5,15 **F**(L0 ⊕ Y0 ⊕ K1)

= S7(K1) ⊕ S7(Y0) ⊕ S7(L0) ⊕ 1

S7(Y0) = S7 **F**(X0 ⊕ K0)

= S7 **F**(L0 ⊕ R0 ⊕ K0)

**= S5,15(L0 ⊕R0 ⊕L4) ⊕ S7(L0 ⊕L4 ⊕R4) ⊕ S7 F(L0 ⊕R0 ⊕K0)**

**S15,21(L4) = S15,21(X0) ⊕ S15,21(Y1) ⊕ S15,21(Y3) ⊕ S15,21(K4)**

S15,21(X0) = S15,21(L0 ⊕ R0)

S15,21(Y3) = S23,31(X) ⊕ 1

= S23,31(L4 ⊕ K4 ⊕ R4 ⊕ K5 ⊕ K3) ⊕ 1

= S23,31(L4 ⊕ R4) ⊕ S23,31(K4 ⊕ K5 ⊕ K3) ⊕ 1

S15,21(Y1) = S15,21 **F**(X1 ⊕ K1)

= S15,21 **F**(L0 ⊕ Y0 ⊕ K1)

= S23,31(K1) ⊕ S23,31(Y0) ⊕ S23,31(L0) ⊕ 1

S23,31(Y0) = S23,31 **F**(X0 ⊕ K0)

= S23,31 **F**(L0 ⊕ R0 ⊕ K0)

**= S15,21(L0 ⊕R0 ⊕L4) ⊕ S23,31(L0 ⊕L4 ⊕R4) ⊕ S23,31 F(L0 ⊕R0 ⊕K0)**

**const\_1 = S23,29(L0 ⊕ R0 ⊕ L4) ⊕ S31(L4 ⊕ R4 ⊕ L0) ⊕ S31 F(L0 ⊕ R0 ⊕ K0)**

**const\_2 = S13(L0 ⊕R0 ⊕L4) ⊕ S7,15,23,31(L0 ⊕L4 ⊕R4) ⊕ S7,15,23,31 F(L0 ⊕R0 ⊕K0)**

**const\_3 = S5,15(L0 ⊕R0 ⊕L4) ⊕ S7(L0 ⊕L4 ⊕R4) ⊕ S7 F(L0 ⊕R0 ⊕K0)**

**const\_4 = S15,21(L0 ⊕R0 ⊕L4) ⊕ S23,31(L0 ⊕L4 ⊕R4) ⊕ S23,31 F(L0 ⊕R0 ⊕K0)**

By combining some of these we get:

**S5,13,21(L0 ⊕R0 ⊕L4) ⊕ S15(L0 ⊕L4 ⊕R4) ⊕ S15 F(L0 ⊕R0 ⊕K0)**

We can use the above constant equation to calculate candidates for K~0

First we generate all possible 12 bit keys with bits at 10..15 && 18…23, 2^12 = 4096

Then we calculate constant value for every plaintext/cypher text pair, if it stays the same every time then save key as a candidate k~.

Rather than iterating over and keeping track of the constant for all the text pairs for each key, it’s obvious we only need to keep track of the first result. Then move on to the next pair if the result is not equal to the saved one then we break out of that loop and move on to the next key.

We’re going to use **const\_2** for performing the attack on K0, we could have also used any or all of the other constants but **const\_2** depends on all the same key bits as the other ones put together so there’s no point. Depending on the input using all constants may reduce the final key set, but for the pairs the lecture provided, it made no difference as I tried all of them at the same time for every key and still got the same final results. Would be trivial updating the code to use the other constantans but for code simplicity reasons I’m just going to use **const\_2.**

The next step is to iterate thought each candidate K~0, reconstruct the 4 bytes’ key using each K~0 and calculate the constant value of the equation.

Just as we did for K~0, we’re only going to keep track of the result of the constant for the first text pair for each key, if the result of the next key pair is not equal to the saved one we break out of the loop and move on to the next K~0.

**Attack K1:**

Using the relation L0 ⊕ Y0 ⊕ Y2 ⊕ L4 ⊕ K4 = K5 ⊕ R4 we derive the following constant equations for the attack on K1:

**S23,29(R4) = S23,29(L0) ⊕ S23,29(Y0) ⊕ S23,29(Y2) ⊕ S23,29(L4) ⊕ S23,29(K4) ⊕ S23,29(K5)**

S23,29(Y2) = S23,29 **F**(X2 ⊕ K2)

= S23,29 **F**(X0 ⊕ Y1 ⊕ K2)

= S31(X0) ⊕ S31(Y1) ⊕ S31(K2) ⊕ 1

= S31(L0 ⊕ R0) ⊕ S31(Y1) ⊕ S31(K2) ⊕ 1

S31(Y1) = S31 **F**(X1 ⊕ K1)

= S31 **F**(L0 ⊕ Y0 ⊕ K1)

= S31 **F**(L0 ⊕ **F**(L0 ⊕ R0 ⊕ K0) ⊕ K1)

S23,29(Y0) = S23,29 **F**(X0 ⊕ K0)

= S23,29 **F**(L0 ⊕ R0 ⊕ K0)

= S31(L0) ⊕ S31(R0) ⊕ S31(K0) ⊕ 1

Sub in the results into the equation to get:

**S23,29(L0 ⊕ L4 ⊕ R4) ⊕ S31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1)**

Following the same process for the other we get the following 4 constant equations:

**const\_1 = S23,29(L0 ⊕ L4 ⊕ R4) ⊕ S31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1)**

**const\_2 = S13(L0 ⊕ L4 ⊕ R4) ⊕ S7,15,23,31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1)**

**const\_3 = S5,15(L0 ⊕ L4 ⊕ R4) ⊕ S7 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1)**

**const\_4 = S15,21(L0 ⊕ L4 ⊕ R4) ⊕ S23,31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1)**

Once again we can combine some of these to get:

**S5,13,21(L0 ⊕ L4 ⊕ R4) ⊕ S15 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1)**

Once again we iterate through 12 bits first, calculate the constant for all keys and text pairs, if it stays the same across all text pairs for a given key we store it as a K~1

After that we reconstruct the possible K1’s from 20 bits using K~1 and attack K1 by calculating “**S13(L0 ⊕ L4 ⊕ R4) ⊕ S7,15,23,31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1)**”

**Attack K2:**

For the attack on K2 we use the same relation that we used for K0 “L4 = X0 ⊕ Y1 ⊕ Y3 ⊕ K4”, we just have to work our way back.

**S23,29(L4) = S23,29(X0) ⊕ S23,29(Y1) ⊕ S23,29(Y3) ⊕ S23,29(K4)**

The following are calculated the same way S23,29(X0), S23,29(Y1), S31(Y0).

S23,29(Y3) = S23,29 **F**(X3 ⊕ K3)

= S23,29 **F**(Y2 ⊕ X1 ⊕ K3)

= S31(Y2) ⊕ S31(X1) ⊕ S31(K3) ⊕ 1

= S31 **F**(X2 ⊕ K2) ⊕ S31(X1) ⊕ S31(K3) ⊕ 1

= S31 **F**(X0 ⊕ Y1 ⊕ K2) ⊕ S31(L0 ⊕ Y0) ⊕ S31(K3) ⊕ 1

= S31 **F**(X0 ⊕ Y1 ⊕ K2) ⊕ S31(L0 ⊕ **F**(X0 ⊕ K0)) ⊕ S31(K3) ⊕ 1

= S31 **F**(L0 ⊕ R0 ⊕ Y1 ⊕ K2) ⊕ S31(L0 ⊕ **F**(L0 ⊕ R0 ⊕ K0)) ⊕ S31(K3) ⊕ 1

= S31 **F**(L0 ⊕ R0 ⊕ **F**(X1 ⊕ K1) ⊕ K2) ⊕ S31(L0 ⊕ **F**(L0 ⊕ R0 ⊕ K0)) ⊕ S31(K3) ⊕ 1

= S31 **F**(L0 ⊕ R0 ⊕ **F**(L0 ⊕ Y0 ⊕ K1) ⊕ K2) ⊕ S31(L0 ⊕ **F**(L0 ⊕ R0 ⊕ K0)) ⊕ S31(K3) ⊕ 1

Just as before we sub in the results and get the following:

S23,29(L0 ⊕ R0⊕ L4) **⊕** S31 **F**(L0 ⊕ R0 ⊕ **F**(L0 ⊕ **F**(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2)

If we do the same for the rest, we end up with the following constants:

**const\_1 = S23,29(L0 ⊕ R0⊕ L4) ⊕ S31 F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2)**

**const\_2 = S13(L0 ⊕ R0⊕ L4) ⊕ S7,15,23,31 F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2)**

**const\_3 = S5,15(L0 ⊕ R0⊕ L4) ⊕ S7 F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2)**

**const\_4 = S15,21(L0 ⊕ R0⊕ L4) ⊕ S23,31 F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2)**

By combining some of these we get the following constant that can be used for finding K~2:

**S5,13,21(L0 ⊕ R0⊕ L4) ⊕ S15 F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2)**

**Attack K3:**

For the attack on K3 we use the same relation that we used for K1 “L0 ⊕ Y0 ⊕ Y2 ⊕ L4 ⊕ K4 = K5 ⊕ R4”, with one slight change. Note that **Y2 = F(L4** ⊕ **K4** ⊕ **Y3** ⊕ **K2)**

**S23,29(L0 ⊕** L4 **⊕ R4) ⊕ S23,29(Y0) ⊕ S23,29(Y2) ⊕ S23,29(K4) ⊕ S23,29(**K5**)**

**Apply Y2 = F(L4** ⊕ **K4** ⊕ **Y3** ⊕ **K2)**

= S23,29(L0 ⊕ L4 ⊕ R4) ⊕ S23,29(Y0) ⊕ S23,29 F(L4 ⊕ K4 ⊕ Y3 ⊕ K2) ⊕ S23,29(K4) ⊕ S23,29(K5)

= S23,29(L0 ⊕ L4 ⊕ R4) ⊕ S23,29(Y0) ⊕ S31(L4) ⊕ S31(K4) ⊕ S31(Y3) ⊕ S31(K2) ⊕ 1

= S23,29(L0 ⊕ L4 ⊕ R4) ⊕ S23,29(Y0) ⊕ S31(L4) ⊕ S31(Y3) ⊕ 1

S23,29(Y0) = S23,29 **F**(X0 ⊕ K0)

= S23,29 **F**(L0 ⊕ R0 ⊕ K0)

= S31(L0) ⊕ S31(R0) ⊕ S31(K0) ⊕ 1

S31(Y3) = S31 **F**(X3 ⊕ K3)

= S31 **F**( **F**(X0 ⊕ Y1 ⊕ K2) ⊕ L0 ⊕ Y0 ⊕ K3)

= S31 **F**(L0 ⊕ **F**(L0 ⊕ R0 ⊕ K0) ⊕ **F**(L0 ⊕ R0 ⊕ **F**(L0 ⊕ **F**(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2) ⊕ K3)

Put it all together and we get:

**= S23,29(L0 ⊕ L4 ⊕ R4) ⊕ S31(L0 ⊕ R0 ⊕ L4) ⊕ S31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2) ⊕ K3)**

The same process can be used to calculate 3 more constants:

**const\_1 = S23,29(L0 ⊕ L4 ⊕ R4) ⊕ S31(L0 ⊕ R0 ⊕ L4) ⊕ S31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2) ⊕ K3)**

**const\_2 = S13(L0 ⊕ L4 ⊕ R4) ⊕ S7,15,23,31(L0 ⊕ R0 ⊕ L4) ⊕ S7,15,23,31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2) ⊕ K3)**

**const\_3 = S5,15(L0 ⊕ L4 ⊕ R4) ⊕ S7(L0 ⊕ R0 ⊕ L4) ⊕ S7 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2) ⊕ K3)**

**const\_4 = S15,21(L0 ⊕ L4 ⊕ R4) ⊕ S23,31(L0 ⊕ R0 ⊕ L4) ⊕ S23,31 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2) ⊕ K3)**

By combining some of these we get:

**S5,13,21(L0 ⊕ L4 ⊕ R4) ⊕ S15(L0 ⊕ R0 ⊕ L4) ⊕ S15 F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ F(L0 ⊕ R0 ⊕ F(L0 ⊕ F(L0 ⊕ R0 ⊕ K0) ⊕ K1) ⊕ K2) ⊕ K3)**

Just do the same as before use the above to find K~3 then use const\_2 to find candidates for K3

**Attack K4:**

From figure 1 is clear that K4 can be found from:

L0**⊕**R0**⊕**Y1**⊕**Y3**⊕**L4

**Attack K5:**

Also from figure 1 we can see that to get K5 we just have to calculate:

L0**⊕**R0**⊕**Y1**⊕**Y3**⊕**L0**⊕**Y0**⊕**Y2**⊕**R4

Which can be simplified to: R0**⊕**Y1**⊕**Y3**⊕**Y0**⊕**Y2**⊕**R4

**Checking Keys:**

All there is left to do it to go through all the cypher texts, decrypt them using the keys and verify that we get the same plain text.

**Results:**

java MyFealLinear 1.81s user 0.10s system 199% cpu 0.954 total

java MyFealLinear 50.82s user 0.38s system 103% cpu 49.365 total

With 200 key pairs and using only one thread/core, it took less than two seconds to find the first correct key combination, close enough to the time required to run a simple Hello world java program. It took around 50 seconds to iterate through the full search space and find 256 correct key combinations. it would be pretty straight forward to extend the solution to use multiple threads speeding up the attack in situation were the number of text pairs is significantly smaller.

256 valid key combinations:

K0 0x2f27cc40 K1 0x7fe36249 K2 0x11bee34a K3 0x4b6fb561 K4 0x8954d914 K5 0x6682b060

K0 0x2f27cc40 K1 0x7fe36249 K2 0x11bee34a K3 0xcbefb561 K4 0x8b54d914 K5 0x6482b060

And so on, 256 times. I’m not going to copy paste it all in here because it takes up too many pages

After examining the bits of all keys, we get the following patterns:

k0: **[0-1]**0101111**[0-1]**0100111**[0-1]**1001100**[0-1]**1000000

k1: **[0-1]**11111**[0-1]**1**[0-1]**1100011**[0-1]**1100010**[0-1]**10010**[0-1]**1

k2: **[0-1]**00100**[0-1]**1**[0-1]**0111110**[0-1]**1100011**[0-1]**10010**[0-1]**0

k3: **[0-1]**10010**[0-1]**1**[0-1]**1101111**[0-1]**0110101**[0-1]**11000**[0-1]**1

k4: 100010**[0-1]**10101010011011001000101**[0-1]**0

k5: 011001**[0-1]**01000001010110000011000**[0-1]**0

[0-1] = either 0 or 1

|  |  |
| --- | --- |
| k0 unknown 0, 8, 16, 24  k1 unknown 0, 6, 8, 16, 24, 30  k2 unknown 0, 6, 8, 16, 24, 30  k3 unknown 0, 6, 8, 16, 24, 30  k4 unknown 6, 30  k5 unknown 6, 30 |  |

If you recall all the constant equations, none depended on any of these unknown bits. Because they were all generated using the equations above, that also do not contain any of these bits. It might be possible to reduce the result set even more If we had a bigger set of text pairs or defined new equations like the ones above by following the bits through the F function.