Cryptographic Analysis Task Report

Author: Nam Sy Hoang Phan

December 8, 2020

1 Heys Cipher Implementation and Cryptanalysis

The Heys Cipher program is located in the assignment folder, in `HeysCipher.java` file. This folder also includes 2 cryptanalysis program `Linear.java` and `Differential.java` which will be used later to perform Linear and Differential Cryptanalysis attack on the Heys Cipher, respectively. To run `Linear.java`, enter in command line:

```
~$ cd com6014
~$ javac assignment1/Linear.java
~$ java assignment1/Linear file/input.txt
```

With `file/input.txt` is the path to a file that contains all plaintext/ciphertext pairs. To generate 10000 pairs of plaintext/ciphertext with random keys and break it, run the program without any parameter: `java assigment1/Linear`. Similarly, to run `Differential.java`:

```
~$ cd com6014
~$ javac assignment1/Differential.java
~$ java assignment1/Differential file/input.txt
```

The code structure and functionality is described in the comments in the code. The program follows Heys' procedure, exhausting 256 candidate keys and increment count whenever the approximation equation holds true. After obtaining all biases from all candidate keys, all sub keys with the best value will be returned since there are no clear difference between them. After deducing all 16 bits of the key all sub keys will be mixed to create a list of potential result.

1.1 Results

Heys' approximation gives the value hex 9 (1001) and hex D (1101) as the results for key bits K_5 to K_8 and K_{13} to K_{15} respectively. This is the only result, there are no other candidate with identical value.

1.2 Breaking the last 8 bits

An approximation with a reasonable bias is first presented. Later the report will introduce a higher bias approximation but requires a more exhaustive search.

An approximation to recover the remaining key bits

An estimation was constructed with the overall bias of 0.5/32 or 0.015625. The approximations are chosen for the following S-boxes:

$$\begin{array}{rl} S_{11}: X_1 \oplus X_4 = Y_1 & \text{with probability } 4/16 \text{ and bias -1/4} \\ S_{14}: X_1 \oplus X_4 = Y_1 & \text{with probability } 4/16 \text{ and bias -1/4} \\ S_{24}: X_1 \oplus X_4 = Y_1 & \text{with probability } 4/16 \text{ and bias -1/4} \\ S_{31}: X_1 = Y_1 \oplus Y_3 & \text{with probability } 10/16 \text{ and bias +1/8} \end{array}$$

The final estimation can be express in the equation below:

$$P_1 \oplus P_4 \oplus P_{13} \oplus P_{16} \oplus U_1 \oplus U_9 = 0 \tag{1}$$

Linear characteristic is presented in the form of a diagram shown in figure 1. The probability of the 3-round approximation is 15.5/32 or 16.5/32 (depends on the key) with the bias of 0.5/32 (0.015625).

Finding a single approximation with a high bias (equals 1/32 similar to Heys' example) was difficult due to many reason. Heys approximation's success was due to a particular approximation: Hex input 4 - Hex output 5. This is one of the most effective approximation as it has a good bias of -4, taking 1 active S-box as input and only 2 S-boxes for output (in fact, this is the only approximation with this 2-1 input/output correlation property).

A more exhaustive search

Including linear approximations with the absolute bias equals to 6 is a risky choice because they all have output connection to at least 3 other S-boxes, which may tremendously reduces the overall bias of the approximation by increasing the number of active S-boxes. Furthermore, applying these approximation to round 3 can avoid this but instead will force the attack to exhaust 12 key bits instead of 8. This lead to the increase of total key exhausted from 256 to 4096. The fact that these approximations have a high bias can potentially compensate for its high combinations of key to search. An approximation was constructed utilizing this characteristic:

$$\begin{array}{rl} S_{11}: X_1 \oplus X_2 = Y_3 & \text{with probability } 12/16 \text{ and bias +1/4} \\ S_{12}: X_1 \oplus X_2 = Y_3 & \text{with probability } 12/16 \text{ and bias +1/4} \\ S_{23}: X_1 \oplus X_2 = Y_3 & \text{with probability } 12/16 \text{ and bias +1/4} \\ S_{33}: X_3 = Y_1 \oplus Y_2 \oplus Y_3 & \text{with probability } 2/16 \text{ and bias -6/16} \end{array}$$

Active S-boxes are presented in figure 2 and the equation of the approximation is:

$$P_1 \oplus P_2 \oplus P_5 \oplus P_6 \oplus U_3 \oplus U_7 \oplus U_{11} = 0 \tag{2}$$

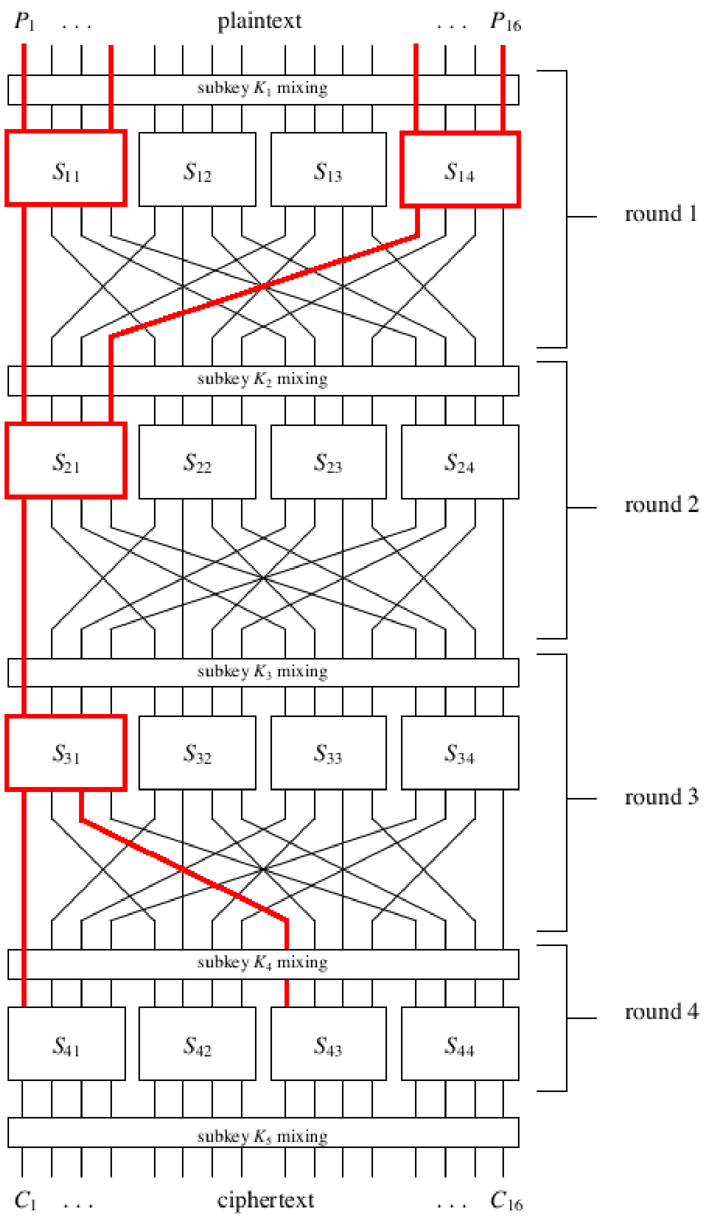

Figure 1: Linear Approximation for K_1 to K_4 and K_9 to K_{12}

The approximation is calculated with the probability of either 14.5/32 or 17.5/32 and bias of 1.5/32 (0.046875). This is significantly higher than the approximation (1) constructed above.

Final Results

Approximation (1) suggests the results for key bits K_1 to K_4 and K_9 to K_{12} can be either hex 0 (0000) and hex 0, hex 0 and hex 8 (1000), hex 8 and hex 0 or hex 8 and hex 8. Combine this with the partial key collected from using Heys' approximation, there are 4 potential results for key K_5 (in hexadecimal form): 090D, 098D, 890D, 898D

Using approximation (2), the number of identical results for bit K_1 to K_12 collected increased slightly, they are (in hexadecimal form): 090, 092, 0B0, 0B2, 290, 292, 2B0, 2B2. Since there is an intersection of bits K_5 to K_8 between these results and Heys' approximation result, the number of possible correct keys can be easily reduced to satisfy both approximations. The final key deduced from using approximation (2) with Heys' approximation is 090D.

Experimenting the approximation with 10000 ciphertext/plaintext pairs generated with 1000 random keys shows that the accuracy of approximation (2) is over 75%. This is slightly higher than Heys' example at 70% since the approximation constructed yield a more distinctive value. Approximation (1) despite not having a significant bias, was still able to give a 65% accuracy. With a more exhaustive search over the keyspace, the accuracy was slightly enhanced. Therefore, there is a trade-off between time complexity and accuracy.

1.3 Breaking of the remaining 4 keys

For an n-round cipher, to break the remaining 4 keys, a simple approach is to decrypt the ciphertext backward using the recovered key K_5 and perform a similar exhaustion again with a n-1 round approximation. First, an approximation for the first 3 round is formed. The aim of constructing a high bias approximation to recover partial sub keys still remains. Next, using round 5 key recovered from the last step, XOR it with the ciphertext, the result is decrypted using the 4 S-boxes. With the obtained partially decrypted result, another exhaustion can be performed on the corresponding key bits to look for the most distinctive result. This process can then be repeated to break all remaining keys.

Reason for unsuccessful break

A clear weakness of this method is the fact that it relies fully on the correctness of the previously recovered keys. In addition, the correctness of encryption keys that are nearer to the end of the cipher is more important because deriving these key wrongly can cause a chain of incorrect results.

A suggestion for improvement is to accept the time-accuracy trade-off and purposely choosing a more exhaustive search. This can ensure the correctness

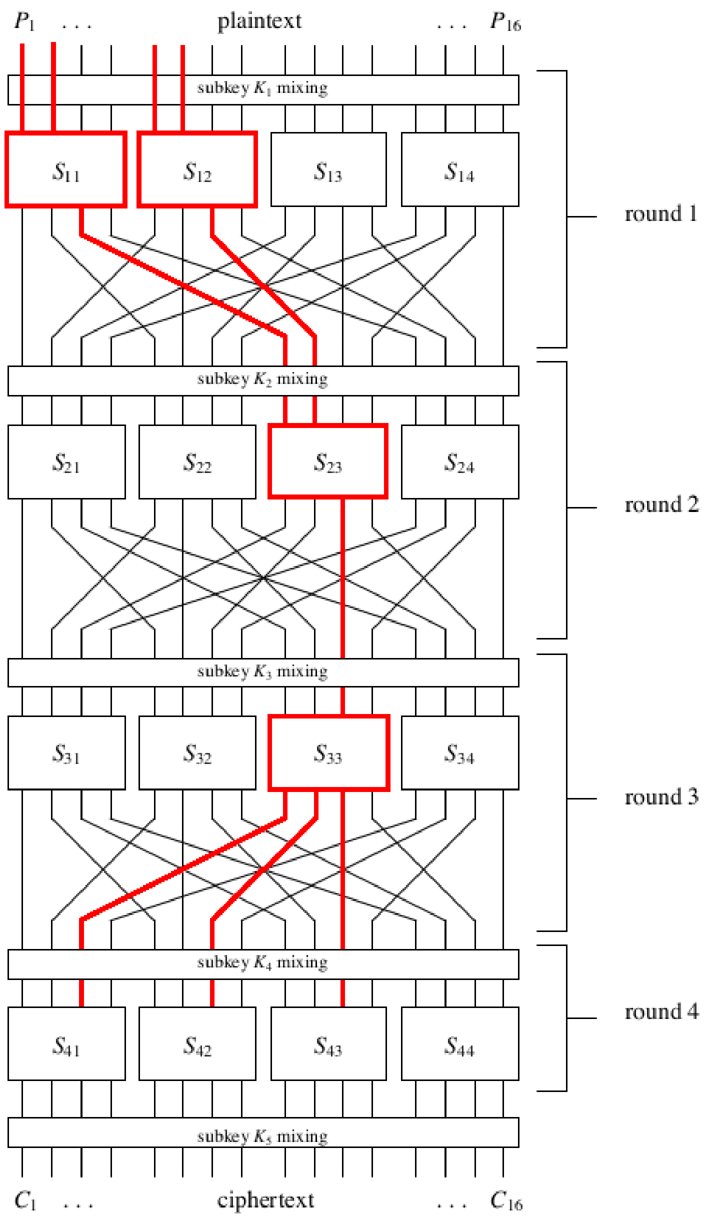

Figure 2: Linear Approximation for K_1 to K_{12}

of the first few keys by having a more distinctive bias, hence reduce the risk of incorrect decryption in later stages.

1.4 Improving resilience of the Cipher

One reason why the attack was possible is the diffusion characteristic of the Cipher is not effective enough, changing 1 bit to the input will not affect all of the output bits. Moreover, the number related bits of the input and output to the round key is trivial, which resulting in cryptanalysts extracting a small enough portion of the round key (Partial Sub Key) to perform exhaustive searchs and recover it.

A solution for this is to have more rounds in the cipher with the purpose of further substituting and scrambling the plaintext, hence producing a more secure ciphertext and have less correlation to the plaintext. Howard Heys stated that the number of active S-boxes is inversely proportional to the overall linear expression bias. If the cipher has more rounds, recovering keys in later rounds will be more difficult as the approximation overall bias cannot be as distinctive because more active S-boxes are required to form it.

Another simple improvement would be to have a larger input size. Doing this would certainly increases the memory and resources required for the cipher, making it less light-weight. On the other hand, it can slow down exhaustive attacks significantly as the attacker is required to exhaust more key bits. If the block size increases, the number of S-boxes increases as well. This lead to another enhancement which is to use more than 1 S-box mapping. With more substitution mapping, more combinations of approximation will be formed which can prolong the finding of a good approximation.

Noted that all suggestions above assume the randomness and independent of the sub keys to each other. The implementation of the Cipher produces the round keys using the java 'Random' library. In practice, this is truly dangerous and round keys should be created using a cryptographically strong random number generator (e.g. SecureRandom).

1.5 A differential attack on Heys Cipher

A differential probability approximation can be constructed for the following S-boxes:

$$\begin{aligned} S_{12} &: X_2 \oplus X_4 = Y_4 && \text{with probability } 4/16 \\ S_{24} &: X_2 = Y_2 \oplus Y_3 && \text{with probability } 6/16 \\ S_{32} &: X_4 = Y_1 \oplus Y_3 && \text{with probability } 4/16 \\ S_{33} &: X_4 = Y_1 \oplus Y_3 && \text{with probability } 4/16 \end{aligned}$$

Figure 3 illustrates active S-boxes and how non-zero bits impact the Cipher. The approximation gives an overall probability of $4/16 \times 6/16 \times (4/16)^2 = 3/512$ or around 0.0058. Experimenting with the approximation over 1000 random keys, an interesting observation was that despite having a considerably lower probability compare to Heys' example of 27/1024 or around 0.026, the accuracy of the attack remains extremely high at 99-100% (for reference, Heys' example approximation produces an absolute 100% accuracy).

2 Attacking a Stream Cipher

2.1 Analyse truth table

With the truth table:

$x_1x_2x_3$	$f(x_1,x_2,x_3)$	$x_1x_2x_3$	$f(x_1,x_2,x_3)$
000	1	100	0
001	0	101	1
010	0	110	0
011	1	111	1

The correlation of the keystream f(x) with x_1 is 4/8 or 50%, with x_2 is also 50% and x_3 is 6/8 or 75%.

2.2 Attack on all three registers

An equation summarize the truth table above can be deduced:

$$f(x_1,x_2,x_3) = (\sim x_1 . \sim x_2 . \sim x_3) + (\sim x_1 . x_2 . x_3) + (x_1 . \sim x_2 . x_3) + (x_1 . x_2 . x_3)$$

First, the stream cipher was used to generate a sufficient amount of keystream bits for the attack. This stream will be used later to deduce the correct LFSR starting position. The longer the length of the LFSRs the more bits are required.

The register with the highest correlation can be exploited. With the correlation of 75% between the output of LFSR 3 and the keystream, an attack can be performed to LFSR 3 by brute-forcing all combination possible for the starting position of LFSR 3. For each of the candidate starting position, equal number of bits to the keystream bits generated above is generated. A score is calculated by taking the hamming distance of the bit stream generated by LFRS 3 and the key stream, divided by their length. With the score for each candidate obtained, a threshold can be used to filter incorrect results. In theory, the correct result (the starting position of LFSR 3) should have the score of around 0.75. Therefore the closest result to 0.75 will be considered correct. After obtaining the inital position of LFSR 3, its starting position can now be set to all 0 and proceed to attack LFSR 1 and 2. The truth table now become:

$x_1x_2x_3$	$f(x_1,x_2,0)$
000	1
010	0
100	0
110	0

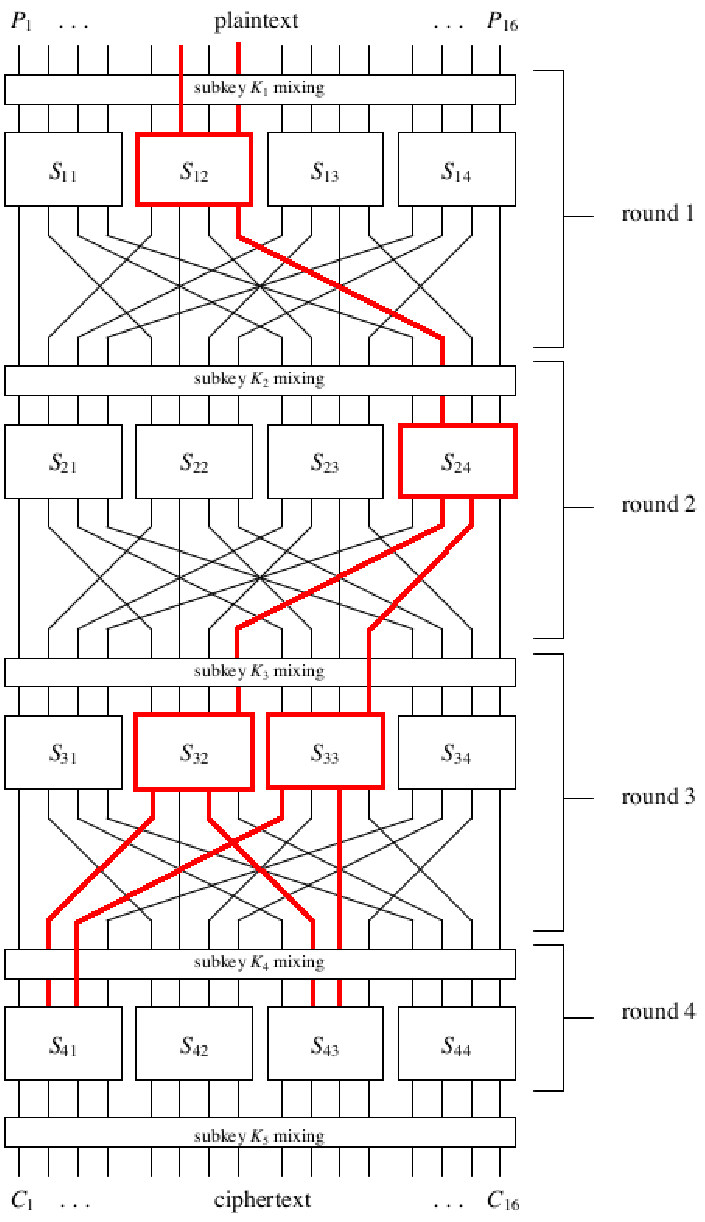

Figure 3: Differential Approximation for K_1 to K_4 and K_9 to K_{12}

and the cipher's new equation is $f(x_1, x_2, 0) = \sim x_1. \sim x_2$. From the truth table above, There is a clear imbalanced correlation between x_1 and $f(x_1, x_2, 0)$ and between x_2 and $f(x_1, x_2, 0)$, both at 1/4 (25%). The same procedure above can be applied consecutively to LFSR 1 and 2 to deduce their starting positions. Exhaust every single possible starting position and calculate their score, filter out higher score candidates and the result is the starting position for LFSR 1 or 2 (depending which one you are exhausting).

2.3 Constraints on direct correlation attack on single register

Using Fast Walsh Hadamard transform, the Walsh spectrum is calculated to be [4,-2,0,2,0,2,0,2]. The maximum Walsh transform or unexpected distance is —2—. The non-linearity of the function is calculated to be 2 which is not the best stat.

With $F(0) = 0$, the cipher is balanced. We observe that $F(001) = -2$, which confirms that this function is not balanced and have some correlation between the input (x_3) and output (keystream). In contrast, both $F(010)$ and $F(100)$ is equal to 0 implies that input x_1 and x_2 have no correlation between them and the output keystream. This is the main reason why correlation attack on LFSR 1 and 2 cannot work.

Ideally there should be no correlation between output and any input stream. For the cipher to resist the correlation attack, $F(001) = F(010) = F(100) = 0$ which makes it

2.4 Further improvement to the resilience of the Cipher

To increase the non-linearity of the Cipher, since the input the the Cipher is only 3, one can simply perform an exhaustive search, altering the cipher truth table to find a better equation. There are 1120 unique functions with non-linearity of 4 and 3 input, and 896 functions with non-linearity of 3.

Another suggestion to improve non-linearity can be based on the non-linearity equation. It can be observed that number of input n is directly proportional to the non-linearity of function f. Therefore increasing the function's number of input can be an option.

To prevent a direct correlation attack on a single register, the most obvious answer is to purposely pick an equation that gives $\hat{F}(001)$ $\hat{F}(010)$ and $\hat{F}(100)$ equal to 0.