Advanced Cryptography

Spring Semester 2021

Homework 4

- This homework contains one question: the differential cryptanalysis of a toy blockcipher.
- You will submit a **report** that will contain all your answers and explanations. The report should be a PDF document. You can use any editor to prepare the report, but Latex is usually the best choice for typesetting math and pseudocode. You will also **submit** your source code. Any programming language is fine, but **python** is recommended. Your code should be **clear and commented**.
- We ask you to **work alone or in groups of 2**. No collaborations are allowed outside of the group you registered for HW1. Please contact the T.A. if you have a good reason to change group. Feel free to ask questions to the T.A.
- We might announce some typos for this homework on Moodle in the news forum. Everybody is subscribed to it and does receive an email as well. If you decided to ignore Moodle emails we recommend that you check the forum regularly.
- The homework is due on Moodle on Friday, 4^{th} of June at 23h59. Please submit 1 report/source code per group.

$\mathsf{B}_{\mathsf{S}_1,\mathsf{S}_2,\mathsf{S}_3,\mathsf{S}_4,\mathsf{S}_{fin},K_S}(M,K)$

$K_0,\ldots,K_5 \leftarrow \mathsf{KeySchedule}_{K_S}(K)$
$X \leftarrow M$
for $r \in \{0,1,2,3,4\}$:
  $X \leftarrow X \oplus K_r$
  $X \leftarrow \mathsf{Sub}(X)$
  $X \leftarrow \mathsf{Per}(X)$
for $i \in \{0,\ldots,5\}$:
  $X_i \leftarrow \mathsf{S}_{fin}(X_i)$
$X \leftarrow X \oplus K_5$
return X

$\mathsf{Sub}(X)$

for $i \in \{0,1,2,3\}$:
  $X_i \leftarrow \mathsf{S}_{i+1}(X_i)$
$X_4 \leftarrow \mathsf{S}_{fin}(X_4)$
$X_5 \leftarrow \mathsf{S}_{fin}(X_5)$

$\mathsf{Per}(X)$

for $i \in \{0,\ldots,11\}$:
  $\hat{X}_i \leftarrow \pi(\hat{X}_i)$
for $i \in \{1,2,3\}$:
  $\hat{Z}_{2i} \leftarrow \hat{X}_{2(i-1)}$
  $\hat{Z}_{2i+1} \leftarrow \hat{X}_{2(i-1)+1}$
for $i \in \{4,5\}$:
  $\hat{Z}_{2i} \leftarrow \hat{X}_{2i-1}$
  $\hat{Z}_{2i+1} \leftarrow \hat{X}_{2i}$
$\hat{Z}_0 \leftarrow \hat{X}_6$
$\hat{Z}_1 \leftarrow \hat{X}_{11}$
return Z

$\pi(\hat{X} \in \{0,1\}^4)$

// bit permutation $[2,0,1,3]$
$\hat{Z}_0 \leftarrow \hat{X}_2; \hat{Z}_1 \leftarrow \hat{X}_0$
$\hat{Z}_2 \leftarrow \hat{X}_1; \hat{Z}_3 \leftarrow \hat{X}_3$
return $\hat{Z}$

$\mathsf{KeySchedule}_{K_S}(K)$

$K_0 \leftarrow \tilde{\mathsf{B}}_{K_S}(K)$
for $i \in \{1,\ldots,5\}$:
  $K_i \leftarrow \tilde{\mathsf{B}}_{K_S}(K_{i-1})$
return $(K_0,\ldots,K_5)$

Figure 1: B blockcipher. π is a permutation of the 4 bits of a nibble. $\tilde{\mathsf{B}}_{K_S}(K)$ is the encryption of K using B with round keys $K_0 = K_1 = \ldots = K_5 = K_S$.

1 Differential Cryptanalysis of a Toy Blockcipher

In this exercise you will perform differential cryptanalyis on a blockcipher $\mathsf{B} : \{0,1\}^{48} \times \{0,1\}^{48} \mapsto \{0,1\}^{48}$. The blockcipher is a substitution-permutation network with 6 rounds (5 rounds + 1 special final round). That is, B iteratively applies: (1) xor with round key, (2) substitution, (3) permutation, except in the final round where it applies (1) substitution, (2) xor with round key. The substitution is performed using 5 different Sboxes $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4, \mathsf{S}_{fin} : \{0,1\}^8 \mapsto \{0,1\}^8$ (each Sbox works on a byte).

Notation. Let $X \in \{0,1\}^{48}$ be the state variable. Then, we let $X_i,\ i \in \{0,\ldots,5\}$ be the byte decomposition of X, i.e. $X = X_0X_1X_2X_3X_4X_5$. Note that we work in Big-Endian here, thus X_0 is the most significant byte and X_5 is the least significant byte. Similarly, $\hat{X}_i \in \{0,1\}^4,\ i \in \{0,\ldots,11\}$ denotes the nibble decomposition of X, i.e. $X = \hat{X}_0 \ldots \hat{X}_{11}$. Again, $\hat{X}_0$ is the most significant nibble.
Example: $X =$ `0xBEEFA0123456`, $X_0 =$ `0xBE`, $\hat{X}_0 =$ `0xB`.

The blockcipher. The complete description of B is given in Figure 1, more information is provided in the caption. In particular, the round keys are derived from the master key K as follows. K_0 (resp. K_i) is the encryption (using B) of K (resp. K_{i-1}), with all round keys equal to the constant K_S. The bit permutation π permutes the 4 bits of a nibble. E.g. $\pi(9) = \pi(\texttt{0b1001}) = \texttt{0b0101} = 5$.

Parameters file. In your parameters file you will find:

- `S_1`, `S_2`, `S_3`, `S_4`, `S_fin`: the Sboxes.
- `K`: the secret key.
- `K_S`: the constant key for the key schedule.
- `pt_1`: a plaintext to encrypt.
- `ct_2`: a ciphertext to decrypt.

Each of these parameters is given as a python bytes object. You can convert them into hexadecimal strings by calling `object.hex()`.

Question 1. Code a function `df_table(S)`, that takes a Sbox as input and outputs a table containing $\mathsf{DP}^S(a, b)$, for all $a, b \in \{0,1\}^8$. Find $(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4)$ s.t. $\mathsf{DP}^{\mathsf{S}_i}(a_i, b_i)$ is high.

We have

- $DP(\texttt{0xa0}, \texttt{0x1b}) = 0.578 = p_1$
- $DP(\texttt{0x1d}, \texttt{0x4f}) = 0.641 = p_2$
- $DP(\texttt{0x2f}, \texttt{0xea}) = 0.594 = p_3$
- $DP(\texttt{0xec}, \texttt{0x66}) = 0.594 = p_4$

Question 2. Code the encryption and decryption functions of B. Provide in your report the encryption $\texttt{ct\_1} = \mathsf{enc}(\texttt{K}, \texttt{pt\_1})$ and decryption $\texttt{pt\_2} = \mathsf{dec}(\texttt{K}, \texttt{ct\_2})$. Your answers needs to be a python bytes object. You can convert any hexadecimal string to a bytes object by calling `bytes.fromhex(hex_string)`.

Question 3. For each byte of the key, provide a differential characteristic that will allow you to recover that byte. **You need to recover the full key using differential cryptanalysis. No bruteforce (even for 1 bit!) or other technique is allowed.** Implement a function that generates corresponding plaintext/ciphertext pairs for each characteristic.

We number the bytes of the last round key K_5 bytes from 0 to 5. The core encryption is the whole encryption except the last substitution/xor.

Recovering byte 4 **and** 5**.** This was actually the easiest, even though bytes 4 and 5 are resulting from the good Sbox S_{fin}. One can notice than an input difference `0xYZ0000000000` (with `YZ` any byte) *always* results in an output difference of `0x00??00000??0`, with $? \in \{0,1\}^4 \setminus \{0^4\}$. I.e. the difference in byte 4 (resp. 5) is `0x0?` (resp. `0x?0`) where ? is any non-zero nibble. The probability of such a characteristic is 1. However, one can replace ? with any non-zero nibble to get a characteristic that happens with large probability ($\approx \frac{1}{16}$ since S_{fin} is a good Sbox).

Recovering bytes $1, 2, 3$**.** For a byte B, by an abuse of notation, we write $\pi(B)$ for the π permutation applied on both nibbles of B.
The first observation is that $\pi(b_1) = a_2$, $\pi(b_2) = a_3$, $\pi(b_3) = a_4$. Then, $b_4 =$ `0x66` and $\pi($`0x6`$) =$ `0xa`. It means that when inputting a difference of `0xa00000000000`, in each successive round we will get:

1. A difference `0x001d00000000` after the first round.
2. A difference `0x00002f000000` after the second round.
3. A difference `0x000000ec0000` after the third round.
4. A difference `0xa0000000a000` after the fourth round.
5. A difference `0x001d00000??0` after the fourth round.

Thus, we have the following characteristics:

- $DP($`0xa00000000000`$,$ `0x001d00000??0`$) = p_1^2 \times p_2 \times p_3 \times p_4$.
- $DP($`0x001d00000000`$,$ `0x0?002f000???`$) = p_1 \times p_2^2 \times p_3 \times p_4$.
- $DP($`0x00002f000000`$,$ `0x0???00ec0???`$) = p_1 \times p_2 \times p_3^2 \times p_4$.

The first (resp. second and third) characteristics allows to recover the byte 1 (resp. 2 and 3) of the last round key.

Recovering byte 0**.** Following the same reasoning as above, we get the following characteristic.

$$DP(\texttt{0x000000ec0000}, \texttt{0xa?????00????}) = p_1 \times p_2 \times p_3 \times p_4^2 \ .$$

That is, if we input a difference of `0xec` in byte 3, we have a good chance of getting an output difference of `0xa?` in the first byte.

Question 4. Implement the attack. Explain in details in your report how you conducted the cryptanalysis. Your code must be runnable and output the correct key K (not the final round key K_5) with high probability. In addition, you must follow these guidelines:

- It must be perfectly clear from your code that K is never used (except to encrypt when collecting plaintext/ciphertext pairs). Ideally, pair collection and cryptanalysis should

be implemented in different files.

- Explain (in the report or a README) how to run the cryptanalysis code s.t. it outputs K.
- You can either precompute the plaintext/ciphertext pairs to speedup the computation (and make it deterministic) or generate them on the fly.
- We repeat that no bruteforce is allowed.

We apply differential cryptanalysis as explained in the course, with the differential trails given above. In particular, the equation that is checked when incrementing a candidate byte k_i is

$$S_{fin}^{-1}(\mathsf{ct}_1 \oplus k_i) \oplus S_{fin}^{-1}(\mathsf{ct}_2 \oplus k_i) = b$$

where ct_i is the encryption of pt_i s.t. $\mathsf{pt}_1 = \mathsf{pt}_2 \oplus a$ and a, b are s.t. $DP(a, b)$ is large (they correspond to the characteristics given above).

Question 5. For each byte of the key, compute (an approximation of) the number of pairs needed for the attack to succeed with high probability. How does it compare with the number of pairs you really needed?

Recovering the bytes 4 and 5 takes only a few queries as the corresponding differential trails happens with probability 1. The differential probability for the trails corresponding to byte $1, 2$ and 3 are $p \approx 0.076$, $p \approx 0.084$ and $p = 0.078$, respectively. Now, the probability p_n that the property is satisfied for a random (wrong key) is $\frac{1}{256} \approx 0.004$. As this noise is still smaller than the "signal" p, we can use the heuristic of the course and conclude that with a number pairs much larger than $\frac{1}{p} \approx 13$, we can recover the correct key. In practice, a few hundred pairs suffice to recover the corresponding byte with prob. ≈ 1.

The analysis for byte 0 is a bit more complicated. We approximate the probability p_n that the output difference is `0xa`? for a random key byte as $\frac{1}{16} = 0.0625$. Now, the differential probability for the corresponding trail is $p \approx 0.078$. So, we see that the noise and signal are very close to each other and we cannot discard the noise in the analysis. The signal is $np + np_n \pm \sqrt{n(p + p_n)}$ and the noise is $np_n \pm \sqrt{np_n}$. We want the expected difference of signal and noise larger than the sum of variances (we take only the variance of the signal here for simplicity). In other words, we want $np >> \sqrt{n(p + p_n)} + \sqrt{np_n}$. We get

$$\sqrt{n} >> \frac{\sqrt{p + p_n} + \sqrt{p_n}}{p}$$

and thus we want the number of queries to be much larger than 64. In practice, the number of queries needed to get the correct byte almost surely is a few hundreds.