SOEN 321 Assignment Deadline: December 12th, 10am
Weight: 33% Submissions are through Moodle

Assignment: Differential Cryptanalysis of WES

Your task is to apply differential cryptanalysis to a toy cipher, WES (Weak Encryption Scheme), and recover the encryption key.

Motivation. The assignment emulates the following scenario. Consider an authentication system that uses a smart card and a reader. In order to authenticate, the card and the reader share the same encryption key and implement a simple authentication protocol: when the card is close to the reader, the reader generates a random number and sends it to the card. The card encrypts this number with the shared key and sends the result back. The reader then decrypts this ciphertext, and if the result equals the previously generated number, this proves that the card knows the key and thus the reader authenticates the card.

If you want to clone the smart card, you will need to extract the encryption key. Doing this directly might require special equipment or might destroy the key in the process. As an alternative, you can apply a cryptanalytic attack to obtain the key. Since the card encrypts whatever message it receives, you can mount a chosen plaintext attack, specifically, you can use differential cryptanalysis.

Tasks

To simulate the smart card, you are given a compiled binary program, `wes-key-##`, that implements WES cipher. The encryption key is hardcoded and is not trivial to recover[1]. But WES specification is public (see below) and you can use `wes-key-##` binary to generate unlimited number of plaintext/ciphertext pairs. **Your ultimate goal is to recover the hardcoded encryption key using differential cryptanalysis**. You specific tasks are:

1. By looking at the WES specification, analyse the input and output differences of the S-boxes and find high-probability differentials. For each S-box, create a difference distribution table in which the rows represent ΔX values (in hexadecimal) and the columns represent ΔY values (in hexadecimal). Each element of the table represents the number of occurrences of the corresponding output difference ΔY given the input difference ΔX. An example for S-box 8 is shown in Table 1.

2. Using the results from the previous task find input/output differences that have probability of 1.

3. Proceed with determining a useful differential characteristic of the overall cipher. Combine S-box difference pairs from round to round so that the nonzero output difference bits from one round correspond to the non-zero input difference bits of

[1]Extracting the key is not terribly difficult either if you have experience in reverse engineering, but that's not the point of this assignment.

		Output Difference															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
I	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	1	0	0	0	2	0	2	0	0	0	2	2	2	2	2	0	2
p	2	0	0	0	0	0	2	2	4	0	4	0	0	0	2	2	0
u	3	0	0	2	0	4	0	0	2	0	2	4	0	2	0	0	0
t	4	0	0	0	2	0	2	0	0	0	4	0	2	0	2	0	4
	5	0	0	4	0	4	0	0	0	0	2	2	0	2	0	0	2
D	6	0	0	0	2	0	0	6	0	0	0	0	2	0	0	6	0
i	7	0	0	2	2	0	2	0	2	0	2	0	2	2	2	0	0
f	8	0	0	0	2	0	0	2	4	0	0	2	4	0	0	0	2
f	9	0	0	0	0	2	0	4	2	2	0	2	0	2	2	0	0
e	A	0	2	2	0	0	0	0	0	2	0	2	0	2	2	4	0
r	B	0	6	0	4	2	0	0	0	0	0	0	2	0	0	2	0
e	C	0	0	2	2	2	0	2	0	4	0	0	0	2	0	0	2
n	D	0	0	2	0	0	4	0	2	2	0	0	0	0	2	0	4
c	E	0	6	0	0	2	4	0	0	2	0	0	0	0	2	0	0
e	F	0	2	2	0	0	0	0	0	4	0	2	2	2	0	2	0

Table 1: Difference Distribution Table for S-box 8

the next round. This will allow you to find a high probability differential consisting of the plaintext difference and the difference of the input to the last round. For this task, you will need to draw a diagram similar to the one from Lecture 4, slide 32.

4. Use the identified differentials (i.e differentials that propagate to the final round) to recover the last round key with differential cryptanalysis. Use `wes-key-##` to generate plaintext-ciphertext pairs. About 6-12 pairs should be enough.

5. Similarly to the previous task, use differential cryptanalysis to recover round 3 subkey.

6. Use plaintext-ciphertext pairs and properties of the WES's key schedule algorithm to break the remaining round keys and recover the master key.

Deliverables.

1. Report (format: PDF, A4, single column; no limit on the number of pages). The report should contain:

   - Difference distribution tables for the remaining 7 S-boxes.
   - A diagram showing propagation of useful (high probability) differentials for the whole cipher that you can use to break the last round.
   - A diagram showing propagation of useful (high probability) differentials for the whole cipher that you can use to break round 3.

- A short description of your attack on WES: differentials used (and why), number of plaintext/ciphertext pairs used for the attack, number of wrong candidate keys (false positives), a short explanation how the master key was recovered, etc.

2. **(The most important)** A program that implements differential cryptanalysis of WES and recovers the master key (with compile/run instructions). The program should find the key in less than 30 minutes (but less than 5 minutes is better). If the runtime is too long, you can record an asciinema with an execution run of your program.

Notes.

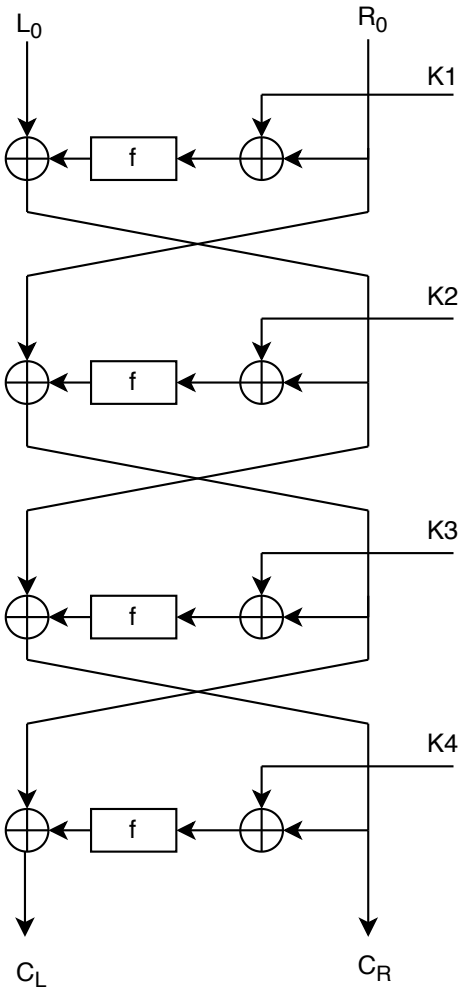
If you cannot uniquely recover the last round key, you can still use differential cryptanalysis to reduce the number of possibilities for the last round key. In this case, provide an estimate of the number of discarded keys for the final round.

Deadline: 12th of December, 9:59am.

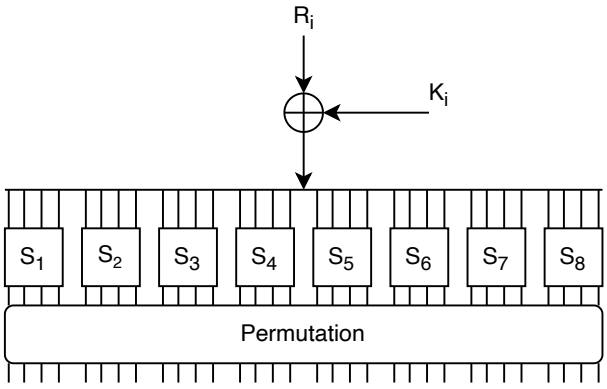
WES Description.

WES is a 4-round Feistel cipher (see Figure 1) with:

- Block size of 64 bits.
- Encryption key of 64 bits.
- Four round keys of 32 bits.



(a) General Structure



(b) Round Function

Figure 1: Weak Encryption Scheme

WES round function (Figure 1b) consists of 3 transformations:

- Key mixing,
- S-box layer,
- Permutation layer.

S-box layer. This layer consists of 8 S-boxes. Each S-box accepts a 4-bit input and substitutes it with a 4-bit output according to Table 2. The table lists the eight S-boxes used in WES. For example, for an input `"0110" (=6)` the corresponding output for S-box S_5 would be `"1011" (=11)`.

	S_1															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	6	12	3	8	14	5	11	1	2	4	13	7	0	10	15	9
	S_2															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	10	14	15	11	6	8	3	13	7	9	2	12	1	0	4	5
	S_3															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	S_4															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	15	9	7	0	10	13	2	4	3	6	12	5	1	8	14	11
	S_5															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	S_6															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	S_7															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	S_8															
in:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out:	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7

Table 2: WES S-boxes

Permutation layer. The permutation shuffles the bits of a 32-bit half-block. Table 3 shows the permutation. For example, 16th bit moves to the first position, 7th bit moves to the second position, etc.

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Clarification: the leftmost bit of th

Table 3: WES Permutation

Round key generation. The size of the main key, M, is 64 bits. It is used to generate four 32-bit round keys using the following procedure. Bits are numbered from 1 to 64 from left to right.

- The first round key, K_1, is generated by taking odd bits in M (1,3,...,63).
- The second round key, K_2, is generated by taking the left half of M (i.e. bits 1-32).
- The third round key, K_3, is generated by taking even bits in M (i.e. 2,4,...64).
- The fourth round key, K_4, is generated by taking the left half of M (i.e. bits 33-64).

Reference implementation. See file `wes.c`.