Introduction to Cryptology (GBIN8U16)
TP — Birthday attack on CBC

2020-03/04

Grading

This TP is graded as the *contrôle continu* of this course. You must send a written report (in a portable format) **detailing** your answers to the questions, and the corresponding source code, *including all tests*, **with compilation and execution instructions** by the end of April, (2020-04-30T23:59+0200) to:

pierre.karpman@univ-grenoble-alpes.fr.

Part one: preparatory work

Download the tarball at https://www-ljk.imag.fr/membres/Pierre.Karpman/cry_intro2019_tp.tar.bz2. This tarball contains the files `specks.c` and `specks.h` that define some instances of the Speck block cipher family (described in https://eprint.iacr.org/2013/404), namely Speck 32/64, Speck 48/96, Speck 64/128.

Question 1

Which of these three Speck instances offer sufficient security against generic attacks (where we consider a single-user setting and a success probability of 2^{-10}) mounted by an adversary with a budget of:

— 1 000 EUR?

— 100 000 EUR?

— 1 000 000 EUR?

— 100 000 000 EUR?

Question 2

Implement a function

```
size_t cbc_enc_s32_64(uint16_t key[4], uint8_t *pt, uint8_t *ct, size_t ptlen)
```

that encrypts a `ptlen`-long plaintext with Speck 32/64. You may assume that the plaintext length is always a multiple of the block size, and you are not required to implement any padding.

N.B. Be careful in how you generate the IVs.

Write a small test function that verifies that the encryption is non-deterministic.

Question 3

Implement a function

```
size_t cbc_dec_s32_64(uint16_t key[4], uint8_t *ct, uint8_t *pt, size_t ctlen)
```

Write a small test function that verifies that an encrypted plaintext is properly decrypted to itself.

Part two: the attack

The goal is now to implement the birthday attack on CBC mode.

Question 1

Recall the attack and its data complexity.

Question 2

Implement the attack for the above implementation of CBC (using Speck 32/64) by recovering the XOR of two plaintext blocks of a single message from the ciphertext only. That is, implement a function:

```
uint32_t attack(uint8_t *ct, size_t ctlen)
```

What kind of data structure can you use to find colliding ciphertext blocks efficiently? How much data do you need on average for the attack to be successful in practice? Is this consistant with the theoretical predictions?

Question 3

Redo the previous question but with Speck 48/64.

Question 4 (bonus)

Same, but with Speck 64/128.

Question 5

Would it be feasible to implement this attack on a decent desktop computer for a block cipher with 80-bit blocks? And with 128-bit blocks?

Question 6

For each of the three Speck ciphers considered here, specify how much *bits* can be encrypted in CBC mode with a single key before the success probability of the attack becomes larger than 2^{-32}. Could this be increased by using the CTR mode? Or another mode *at all*?