Birthday Attack on CBC
Introduction to Cryptology TP

Charles Marshall, Bruno Lehouque

April 2019

Part one: Preparatory work

Encryption/Decryption

We implemented both `cbc_enc`, and `cbc_dec` in the file cbc.c. In the early stages of testing, we manually wrote **ptlen**-long plaintexts, but switched to using online generated lorem-ipsum files at the following website.

To simplify the use of texts with different block sizes, we added a simple padding. When the text size isn't a multiple of the block size, we append some 0's at the end of the text to fill the last block. For additional simplicity, our code is only able to handle block sizes that are multiples of 16.

We have written two test functions, entitled `verify_non_deterministic`, and `verify_proper_decryption` in order to verify that our encryption is non-deterministic, and that `cbc_dec` does its job properly. These functions can be found in the file verif.c. Additionally, we generate random IV's with secure random numbers using `getrandom(prev_ct, BYTES_PER_BLOCK, GRND_NONBLOCK)`, where `BYTES_PER_BLOCK` is defined as `(HALF_BLOCK_SIZE/8)*2)`. `HALF_BLOCK_SIZE` can be changed in tczero.h.

All the necessary information on testing these functions can be found in the README.md file included with our source code.

Part two: The attack

Question 1

1.1 The principle

The principle of the birthday attack is as follows: given a ciphertext consisting of $q + 1$ blocks, c_0 to c_q (where c_0 is the IV), we wish to find two identical encrypted blocks $c_i = c_j$ with $i \neq j$. This is called a collision.

By definition, $c_i = \mathcal{E}(k, m_i \oplus c_{i-1})$ and $c_j = \mathcal{E}(k, m_j \oplus c_{j-1})$. Thus, $m_i \oplus c_{i-1}$ must be equal to $m_i \oplus c_{j-1}$. From that, we can deduce

$$m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$$

To express it in words, when we find a collision between two ciphertext blocks, we can recover the `xor` of the associated plaintext blocks by `xor`-ing the two preceeding ciphertext blocks.

In general, to find such an instance, one must evaluate the function using various different inputs, chosen at random, until the same output is found for two distinct inputs. This method is made efficient due to the birthday bound, and is studied for the remainder of this report.

1.2 Complexity

For a Block Cipher with n-bit blocks, there exist 2^n different possible blocks. The *birthday paradox* states that there is a "high" probability of collision after encrypting $2^{n/2}$ blocks due to a "pigeonhole" in the hash table where two generated ciphertexts yield the same hash value. This is called a collision and will be the goal of our attack on CBC.

1.3 Approach

To find colliding ciphertext blocks efficiently, we use a hash table, where searching, inserting, and deleting all take $O(1)$ in the best case.

To execute the attack, we encrypt a long plaintext retrieved from a file (details in readme file), and store the resulting blocks of ciphertext in our hash table. Then, if we don't find a collision within the given ciphertext, we encrypt a small plaintext repeatedly, each time checking if the resulting ciphertext already exists in our hash table, and if not, inserting it into our hash table. We do this until a collision is found, and we take note of the amount of data used up until

the collision. In theory, tests of this attack will average $2^{n/2}$ blocks to find a collision.

All the necessary information on testing this attack can be found in the README.md file, included with our source code.

Question 2

For 32-bit blocks, the average number of block encryptions needed to find a collision should be around $2^{32/2} = 2^{16} = 65536$.

Experimentally, when running the attack 100 times, with an initial plaintext size of 1Mb, we found that the average number of encryptions needed for finding a collision was around 80000, with an average time of $0.2s$ per attack. This is consistent with the theoritical results.

Question 3

As previously stated, our code operates with blocks whose sizes are multiples of 16. So, instead of blocks of size 50, we used blocks of size 48.

For 48-bit blocks, the average number of encryptions should be around $2^{48/2} = 2^{24} \approx 16$ million.

In practice, we found that we needed between 18 to 20 million block encryptions on average to find a collision, which is consistent with expectations. The average time for an attack was around 2 minutes. This is 600 times slower than for 32-bit blocks.

If the time only depended on the number of blocks encrypted, we could have expected a slowdown factor of $256 = 2^8 = 2^{24}/2^{16}$. The 600 times slower factor is likely due to the length of the ciphertext that makes computations slower.

Question 4

For 64-bit blocks, we expect to need 2^{32} encrypted blocks to find a collision. Since a block is 8 bytes, plus another 16 bytes to keep a pointer to the block and its predecessor, it would require $24 \times 4 = 96$Gb of memory.

The time would be multiplied by another $256 = 2^8 = 2^{32}/2^{24}$. If we add a factor of 2 or 4 to account for the extra time due to the block length, it would take around 1000 times longer.

1000×2 minutes is 33 hours. This is doable in theory if you have a very good laptop with 128Gb of RAM and a lot of patience.

Question 5

Blocks of size 80 are certainly seem out of reach for a personal computer. The computation would require Terabytes of memory and take several years to terminate. Thus, 128-bit blocks are clearly unthinkable and not feasible to implement on a "decent desktop computer".