Laboratory 2 – AISC

A simplified version of AES

Agnetta Stefano, Brozzo Doda Umberto, Macario Davide

March 29^{th}, 2023

A.Y. 2021–2022

1 Introduction

This laboratory revolved around the analysis of a simplified version of AES (Advanced Encryption Standard), which was provided as a python script, which can be found at [1], and which also contained all necessary sub–blocks for each round (SBox, ShiftRows, MixColumns and AddKey). These blocks are designed to work on 16–bit inputs, organized as 2–by–2 matrices of 4–bit '*nibbles*'. Also the key is a 16-bit string.

The focus of this analysis is to test the behavior of the simplified AES in terms of avalanche effect, and to cryptanalyze an improper implementation of a block cipher, based on the provided sub-blocks, in order to break a ciphertext.

2 Avalanche effect

3 Improperly implemented block cipher

The second part of this laboratory concerned the cryptanalysis of an improperly-implemented block cipher of which the description was given, and which used on the blocks that were provided for the simplified AES.

The proposed procedure was essentially that of carrying out a KPA having observed a single plaintext-ciphertext pair, which was obtained with the same key as the large ciphertext that needed to be broken. The pair was the following:

- **Known plaintext**: `0b0111001001101110`
- **Known ciphertext**: `0b0001111001100101`

Having the possibility to observe the Python code which implemented the encryption (and decryption) mechanism, it was found that the block cipher only consisted of a single round, composed of "Substitute Bytes", "Shift Row" and "Add Key" (with a 16-bit subkey) blocks only, whose order was reversed for decryption. This structure already highlights one main security issue, as the lack of an initial "Add Key" stage makes it so that the knowledge about the bits before the last "Add Key" automatically reveals the plaintext (the remaining stages do not contain 'secrets' such as the key).

Furthermore, the key expansion procedure proposed for the simple AES can be seen to produce as subkeys k_0 and k_1 the first 8 bits and the last 8 bits respectively. For this reason, the key which is added during the last (and only) "Add Key" stage corresponds to the actual key itself, as provided by the user.

Figure 1 shows the structure of the described block cipher.

Putting all these things together, the proposed attack consists in performing the first two stages on the known plaintext ("Substitute Bytes" and "Shift Rows") and then directly finding the key by evaluating the XOR between the result of the first stages and the known ciphertext. This last step guarantees the discovery of the key (in theory the ensemble of subkeys k_0 and k_1), thanks to the properties of the XOR operator ($A \oplus B = C \implies A \oplus C = B$).

By performing these steps, the key used in the known pleintext-ciphertext pair was found to be `0b1000000111101111`.

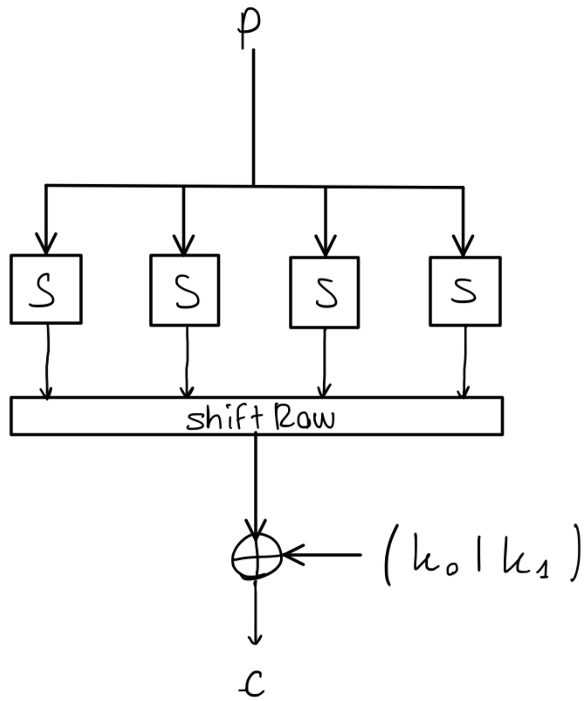

Figure 1: Structure of the improperly implemented block cipher

The ciphertext was then decrypted, knowing the block cipher was used on two characters at a time (16 bits total).

The resulting plaintext contained a quote from 'The Lord of the Rings':

> *When Mr. Bilbo Baggins of Bag End announced that he would shortly be celebrating his eleventy-first birthday with a party of special magnificence, there was much talk and excitement in Hobbiton. Bilbo was very rich and very peculiar, and had been the wonder of the Shire for sixty years, ever since his remarkable disappearance and unexpected return. The riches he had brought back from his travels had now become a local legend, and it was popularly believed, whatever the old folk might say, that the Hill at Bag End was full of tunnels stuffed with treasure. And if that was not enough for fame, there was also his prolonged vigour to marvel at. Time wore on, but it seemed to have little effect on Mr. Baggins. At ninety he was much the same as at fifty. At ninety-nine they began to call him well-preserved, but unchanged would have been nearer the mark. There were some that shook their heads and thought this was too much of a good thing; it seemed unfair that anyone should possess (apparently) perpetual youth as well as (reputedly) inexhaustible wealth*

3.1 Decryption under COA

As seen, breaking the encryption scheme under KPA is simple, as the mechanism lacks the initial XOR operation between plaintext and sub-key. Supposing there is no available known plaintext–ciphertext pairs, the approach taken by an attacker needs to be different.

The overall encryption process for a long plaintext (such as the one in 'ciphertext.txt'), then becomes similar to that of a Vigenère cipher, in which different portions (blocks) of the message are encrypted in the same way. In particular, after being passed through the S-Boxes and the "Shift Rows", the same key is applied on every set of 2 characters (16 bits).

The presence of the two initial sections, however, makes it difficult for the opponent to proceed in the same way as in lab 1, i.e., by performing frequency analysis of the (two) subsequences. Indeed, thanks to the confusion introduced by the 'Substitute Bytes' and 'Shift Rows' blocks, which cause bits from the first of the 2 characters to have an impact on the last four bits, and vice–versa. As a result, each character in the ciphertext essentially depends on two characters from the plaintext. This makes it impossible to cryptanalyze this block cipher in the case of a Ciphertext Only Attack and for this reason the only viable attack is to brute force the 16-bit key. The only approach that may be followed to slightly simplify the search could be that of realizing that the same couple of (ordered) characters are associated to the same couples of characters in the ciphertext, since, even if improperly implemented, this block cipher has been defined to be invertible.

4 Conclusions

References

[1] https://jhafranco.com/2012/02/11/simplified-aes-implementation-in-python