Hardware Security EL9423
LAB 2

Nimisha Limaye (nsl278)
Ashik H. Poojari (ap4613)

February 28, 2017

Abstract

In this lab, we have designed an algorithm to scan out the key used for Advanced Encryption Standard (AES) using the register available in the AES code [1]. The algorithm is inspired from [2], where we assume that the register is placed after the round operation, since most of the AES designs have this register after the round operation.

1. AES Encryption

We are using a 128-bit input and 128-bit key AES algorithm. We are considering that the location of round register is placed after the round operation as shown in Fig. 1 [2]. First, we generate keys for each round using the KeyExpansion function. Then we exclusive OR the first generated key with the plaintext input using the AddRoundKey function. This is given to SubBytes function, ShiftRows, MixColumns and then to AddRoundKey again. The output of this AddRoundKeys function is stored in the round register, used for testing. We use this round register to leak the key.

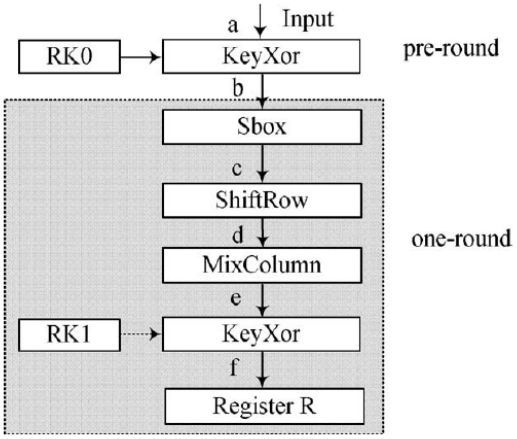

Figure 1: Round Operations of AES encryption

2. Attack on AES

2.1 Leaking the round key

Step 1:
We start with an initial value of 0 given to variable t. We calculate a0 = 2*t and a1 = 2*t + 1, and store it as a byte in the plaintexts p0 and p1 respectively. We perform one round of AES encryption on these plaintexts and obtain two ciphertexts f0 and f1 respectively.
Step 2:
The two ciphertexts obtained from step 1 are exclusive OR-ed and we count the number of ones in this result. We know from [1] that if the number of ones are 9, 12, 23 or 24, we obtain the result of first AddRoundKey function in form of pairs.
Step 3:
We exclusive OR the pair obtained in step 2 with either a0 or a1 to get the stolen key pair. Now we have two options for every byte in the key. We wrote an algorithm to generate all possible key combinations using the stolen key pairs.

2.2 Recovering the secret key

Step 1:
Take the key generated from step 3 and perform AES encryption.
Step 2:
Compare the ciphertext generated from this key with the original ciphertext. If they are same, then we have found the key, else repeat the above two steps.

3. Results

Fig. 2 shows the plot of Number of plaintexts required to leak a particular key. Since our plaintexts increment by 1 starting from 0, we get a negative slope graph. Careful study of the above graph and trying different arrangements of plaintext, may help to reduce the number for each key pattern.

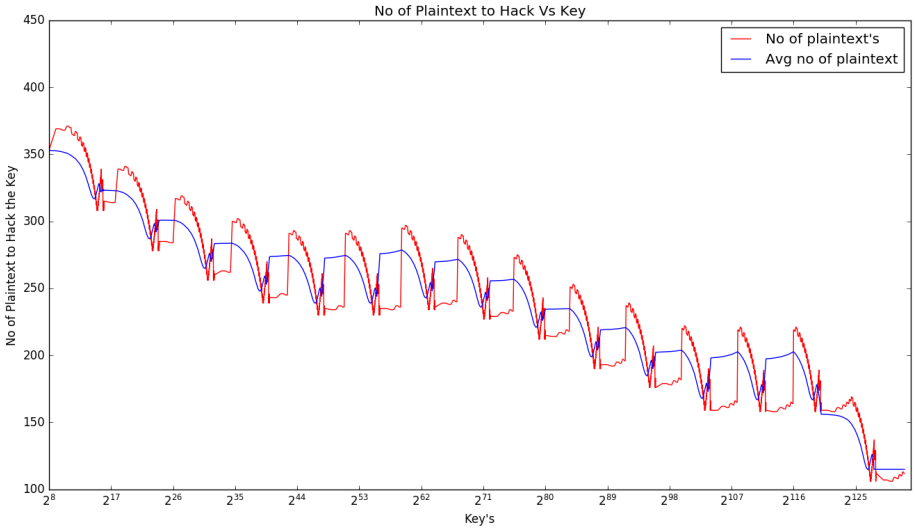

Figure 2: Number of plaintexts used to leak the secret key

4. Conclusion

The round key was leaked using scan out approach where the round register is placed after the round operation. Further, brute-force was applied on 16 bytes to recover the secret key.

References

[1] URL: https://github.com/B-Con/crypto-algorithms.

[2] B. Yang K. Wu and R. Karri. *Secure scan: a design-for-test architecture for crypto chips*. pp. 135–140, DAC 2005. ACM, 2005.