Crypto Engineering TP — Square attack on 3½ rounds of AES

Calister NNONA
TALAKI Ezinam Bertrand

November 2018

Exercise 1 : Warming up

Question 1

The function xtime in the file *aes-128.enc.c* multiplies the input by the polynomial X over Galois field (GF(2^8)) with $P = X^8 + X^4 + X^3 + X + 1$.

Example : We'll try to compute $(X \times Q(X))$ mod P with $Q(X) = X^7 + X^3 + X + 1$
$X \times Q(X) = X^8 + X^4 + X^2 + X$
$(X^8 + X^4 + X^2 + X) mod P = X^3 + X^2 + 1$

On the other hand, when using the function xtime on the input 0x8B (10001011) we get the output 0x0D (00001101) (Figure 1) which corresponds to the polynomial $X^3 + X^2 + 1$.

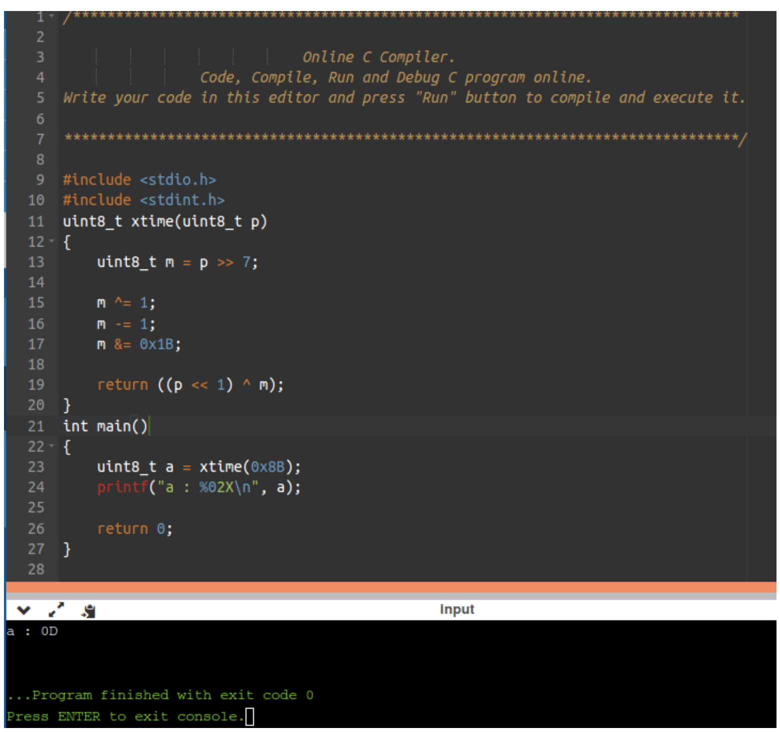

FIGURE 1 – Execution of xtime() with the input 0x8B

To write a variant of xtime for a different representation of $\mathbb{F}_{2^8}$ (with the polynomial $X^8 + X^6 + X^5 + X^4 + X^3 + X + 1$) we can just change the line *m &= 0x1B;* to *m &= 0x7B;*.

Question 2

To implement the function prev_aes128_round_key we can just implement the reverse of the next_aes128_round_key.

Question 3

To implement the described construction, we can use the AES encryption function twice with the same plaintext and k1 and k2. Then we wil XOR the result to get the output of the keyed function.

If k1 = k2 the output will be the all zero value (0) and we could loose the information (message x).

Exercise 2 : Key-recovery attack

Question 1

The partial decryption function takes a state byte of the last round ($1/2$ round) and a key byte and computes the previous state byte by decrypting the last round. In our case, we took as input the whole state byte and the whole key and we return the previous state. This will help us implement the entire attack at the end.

In order to test it we use the AES standard : we take the output of the $3 1/2$ given the initial input as defined in the standard and retrieve the output of the third round. Then we compare our output to the one provided in the standard.

Question 2

The entire attack is implemented like this :

- We make queries to the aes $3 1/2$ encryption oracle with our 256 plaintexts
- We decrypt the $1/2$ round
- Then we use the distinguisher to find out for any guessed byte of the key, if the oracle acts like a permutation or not.