University of Birmingham - Cryptography AES Exercise 2

Haotian Xiang

November 9, 2019

1 Exercise 1

1.1 Questions

1.1.1 Question 1

Consider AES with 128-bit keys. Assume that the initial subkey (k0) and the round 1 subkey (k1) are both all-zero, and that the plaintext block is also all-zero. What is the output of the first round?

1.1.2 Answer 1

Plain text: 0000 0000 0000 0000 k0: 0000 0000 0000 0000 k1: 0000 0000 0000 0000

Step 1: Add round key0

$\because 0 \oplus 0 = 0$

$\therefore$ plain text $\oplus$ k0 = 0000 0000 0000 0000

Step 2: Substitute bytes

S-Box table

```
// 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00 63 7c 77 7b f2 6b 6f c5 30 01 67 2b fe d7 ab 76
10 ca 82 c9 7d fa 59 47 f0 ad d4 a2 af 9c a4 72 c0
20 b7 fd 93 26 36 3f f7 cc 34 a5 e5 f1 71 d8 31 15
30 04 c7 23 c3 18 96 05 9a 07 12 80 e2 eb 27 b2 75
40 09 83 2c 1a 1b 6e 5a a0 52 3b d6 b3 29 e3 2f 84
50 53 d1 00 ed 20 fc b1 5b 6a cb be 39 4a 4c 58 cf
60 d0 ef aa fb 43 4d 33 85 45 f9 02 7f 50 3c 9f a8
70 51 a3 40 8f 92 9d 38 f5 bc b6 da 21 10 ff f3 d2
80 cd 0c 13 ec 5f 97 44 17 c4 a7 7e 3d 64 5d 19 73
90 60 81 4f dc 22 2a 90 88 46 ee b8 14 de 5e 0b db
a0 e0 32 3a 0a 49 06 24 5c c2 d3 ac 62 91 95 e4 79
b0 e7 c8 37 6d 8d d5 4e a9 6c 56 f4 ea 65 7a ae 08
c0 ba 78 25 2e 1c a6 b4 c6 e8 dd 74 1f 4b bd 8b 8a
```

```
d0 70 3e b5 66 48 03 f6 0e 61 35 57 b9 86 c1 1d 9e
e0 e1 f8 98 11 69 d9 8e 94 9b 1e 87 e9 ce 55 28 df
f0 8c a1 89 0d bf e6 42 68 41 99 2d 0f b0 54 bb 16
```

$\therefore$ lookup 0x00 correspondence value is 0x63
After S-Box

```
63 63 63 63
63 63 63 63
63 63 63 63
63 63 63 63
```

Step 3: Shift row
$\because$ all bytes are 63.
so after shifting row, values are still

```
63 63 63 63
63 63 63 63
63 63 63 63
63 63 63 63
```

Step 4: Mix columns

$$\begin{pmatrix} 63 \\ 63 \\ 63 \\ 63 \end{pmatrix} \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} = \begin{pmatrix} 63 \\ 63 \\ 63 \\ 63 \end{pmatrix}$$

$(0x63 * 0x02) \oplus (0x63 * 0x03) \oplus (0x63 * 0x01) \oplus (0x63 * 0x01) = 0x63$
$\therefore$ all columns are 63 so values remain the same

$$\begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix}$$

Step 5: Add round key1
0x63 $\oplus$ 0x00 = 0x63
$\therefore$ result is:

$$\begin{pmatrix} 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 \end{pmatrix}$$

1.1.3 Question 2

Consider AES with 128-bit keys. Suppose the encryption key is all-one (i.e., 128 ones). Compute the initial subkey (K0) and the round 1 subkey (K1).

1.1.4 Answer 2

Answer:

$\because$ key is 1111 1111 1111 1111

$\therefore$ key in hex form is FF FF FF FF

$\because$

$$\begin{pmatrix} k0 & k4 & k8 & k12 \\ k1 & k5 & k9 & k13 \\ k2 & k6 & k10 & k14 \\ k3 & 7 & k11 & k15 \end{pmatrix} = \begin{pmatrix} FF & FF & FF & FF \\ FF & FF & FF & FF \\ FF & FF & FF & FF \\ FF & FF & FF & FF \end{pmatrix}$$

$\therefore$

$$\begin{pmatrix} w0 \\ w1 \\ w2 \\ w3 \end{pmatrix} = \begin{pmatrix} FF \\ FF \\ FF \\ FF \end{pmatrix}$$

RotWord:

$$\begin{pmatrix} FF \\ FF \\ FF \\ FF \end{pmatrix}$$

SubWord:

$$\begin{pmatrix} 16 \\ 16 \\ 16 \\ 16 \end{pmatrix}$$

w0 = FF FF FF FF $\oplus$ 0x16 0x16 0x16 0x16 $\oplus$ $Rcon(4)$ = 0x17 0x16 0x16 0x16

$\therefore$ w4 = w0 $\oplus$ 0xe8 0xe9 0xe9 0xe9

$\therefore$ w5 = w1 $\oplus$ 0x17 0x16 0x16, 0x16

$\therefore$ w6 = w2 $\oplus$ 0xe8 0xe9 0xe9, 0xe9

$\therefore$ w7 = w3 $\oplus$ 0x17 0x16 0x16 0x16

$\therefore$ K0 = 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff

$\therefore$ K1 = 0xe8 0xe9 0xe9 0xe9 0x17 0x16 0x16 0x16 0xe8 0xe9 0xe9 0xe9 0x17 0x16 0x16 0x16

1.1.5 Question 3

Programming exercise. Let MY60SHA be a hash function which outputs the first 60 bits (15 nibbles) of SHA-1. For example, SHA-1 of "mark" is f1b5a91d4d6ad523f2610114591c007e75d so MY60SHA of "mark" is f1b5a91d4d6ad52. Find any collision for MY60SHA. (Note: you should find two strings such that the unix command echo -n str — sha1sum - — cut -c1-15 produces the same answer when str is replaced by each

string. To enable me to verify your answer, please make sure the two strings are typable on a regular keyboard! Hint: You should not write the code for SHA-1; you should use an existing library. Also, it's a good idea to find shorter collisions first. For example, start off finding a collision for the first 24 bits (6 nibbles); that's a lot easier. The challenge that 60 bits gives you is that you probably can't store all the intermediate hashes you generate in memory (unless you have a lot of memory). Nevertheless, you should be able to write a program which finds a 60 bit collision in a few hours on a regular desktop or laptop computer. My program is about 50 lines and runs in a few hours on my desktop computer that has 4GB RAM.

1.1.6 Answer 3

```
Collision Example:
Key: cca722b08199
Value 1: \j4Us7A*6
Value 2: 4=XXE
```

```
import hashlib
import string
import random
from random import randint
from sys import getsizeof
import datetime
import time
import threading
import os

hash1_table = dict()
hash2_table = dict()

def unique_strings(k: int, hash_number: int,
                   pool: str='1234567890ZAQWSXCDERFVBGTYHNJUIMKLOPzaqwsxcderfvb

    join = ''.join
    token = join(random.choices(pool, k=k))
    return token

def add_hash(tabel_number,table):
    while(len(table) != 10000000):
        random_string = unique_strings(randint(5,10),5)
        random_string_hash = hashlib.sha1(random_string.encode('utf-8')).hexd
        table.update({random_string_hash:random_string})
        print("table" + str(tabel_number) + ":" + str(len(table)))
```

```
def compare(first , second):
    sharedKeys = set(first.keys()).intersection(second.keys())
    for key in sharedKeys:
        print("comparing now")
        if first[key] != second[key]:
            print('Key: {}, Value 1: {}, Value 2: {}'.format(key, first[key],
            os._exit(1)

while True:
    compare(hash1_table,hash2_table)
    hash1_table = {}
    hash2_table = {}
    add_hash(1,hash1_table)
    add_hash(2,hash2_table)
```