(Deadline: 12 noon,14 Sep 2017)

Write a program to illustrate padding oracle attack. In your program, a secret key is "hardcoded". So, you know the secret key. Nevertheless, your attack mechanism must not know the hardcoded secret key.

Task 1: (AES with padding)

Write a subroutine with these parameters

AES_Padding (*p, v*)

where p is a string and *v* is a 16-byte string.

The output is the ciphertext that is encrypted under AES CBC mode, where

- the IV is *v* .
- the plaintext is *p.*
- the plaintext is padded according to PKCS#7
- the 128-bit secret key is (in hexadecimal representation)

6B 6B 6B 6B 6B 6B 6B 6B 6B 6B 6B 6B 6B 6B 6B 6B

For programming convenient, you can add a parameter *n*

AES_Padding (*p*, *n*, *v*)

where *n* is the length of p.

Task 2: (AES Padding Oracle)

Write a subroutine with these parameters

AES_Valid_Padding (*c,v*)

where c is the ciphertext. The output is 1 if the decrypted plaintext is well-formed under PKCS#7 padding. Otherwise, the output is 0.

For programming convenient, you can add a parameter *m*

AES_Valid_Padding (*c, b, v*)

where *b* is the number of blocks in c.

Task 3: (Padding Oracle Attack)

Write a program that:

1. Reads a string p of alphabets from the standard input. (Assume that the format of the input is correct, i.e. not necessary to carry out input validation on p).

2. Encrypts p using the secret key with the following IV (hexadecimal representation).

   11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00

3. Output the ciphertext in hexadecimal representation (not including the iv).
4. Perform oracle attack and output the plaintext (in ASCII characters).

For ease of grading, your main program should clearly show the following flow:

```
//-----------------------------------//
Read  p,n;          // read in a string of alphabets p  with length n where   0 < n < 300.
v =   ….            // set v to be a 16-byte string with hexadecimal values
                    //          11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
c = AES_Padding (p,n,v);
display c;
m = ATT( c, v, n);   //  This is the attack
display m
//-----------------------------------//
```

What to submit

1. The source code. Any programming language acceptable.
2. A brief report (pdf format) giving the "screen shot" (can be text format) of a few successful test instances. Report file name is <*studentid*>.pdf
3. Upload a zip file <*studentid*>_<*name*>.zip E.g. A1234567F_Alice.zip

Grading scheme.

1. [9 marks] Plaintext found in all cases.
2. [+1 mark] The attack ATT(c,v) does not take in parameter n. That is, it can infer the value of n from c and v.
3. [6,7,8 marks] Able to find the last byte of the plaintext in some cases.
4. [4 marks] Able to encrypt correctly.