6.857 Homework **Problem Set 3** # 3-1 Modes of Operation

Julia Huang
Skanda Koppula
Kimberly Toy

March 16, 2015

6.857 Homework **Problem Set 3** # 3-1 Stream Ciphers
Julia Huang
Skanda Koppula
Kimberly Toy March 16, 2015

6.857 Homework
Julia Huang
Skanda Koppula
Kimberly Toy

Problem Set 3

3-3 AES Distinguisher

March 16, 2015

A. We need to determine if two samples are drawn from the same distribution. For this, we use the Kolmogorov-Smirnov test that can be used to test whether the two underlying probability distributions for two one-dimensional samples are different. This is what we desire when analyzing our two samples $x_1, x_2, x_3, ..., x_n$ and $y_1, y_2, y_3, ..., y_n$. The KS test outputs a p-value, if below an appropriate threshold (e.g. 0.10), the samples are probably have different underlying distributions.

For our implementation of Kolmogorov-Smirnov, we use the `scipy.stats` package:

```
from scipy.stats import ks_2samp
import numpy as np

def run_ks(lst1, lst2):

    '''Returns the p-value of a 2-sample Kolmogorov-Smirnov test'''

    data1 = np.array(lst1, np.int32)
    data2 = np.array(lst2, np.int32)

    return ks_2samp(data1, data2)[0]
```

B and C. For our implementation of AES/Rijndael, we used a Python script based on an implementation by Bram Cohen: `http://wiki.birth-online.de/snippets/python/aes-rijndael`. We modified the script to take the number of rounds as an initialization parameter. Using this, we calculated two KS scores for each 0¡r¡21, where r is the number of rounds in our Rijndael: (1) the KS between the sample of distinct bytes in $AES_r = F(r, p, q)$ and $AES_{10} = F(10, p, q)$, and (2) the KS between the sample of distinct bytes in $AES_r = F(r, p, q)$ and a sample of random bytes of same size. To do this, we wrote the following script:

```
from rijndael import rijndael
import os
import random

key_128 = os.urandom(16)
message = os.urandom(16)
print 'Key is: ', key_128
print 'Message is: ', message

def F(r, p, q, key, message):

    prefix = message[:p]
    suffix = message[p+1:]
    S, T = set(), list()
    for i in xrange(256):
        S.add(prefix + chr(i) + suffix)

    for string in S:
        aes_obj = rijndael(key, block_size = 16, rounds = r)
        ciphertext = aes_obj.encrypt(string)
        T.append(ciphertext[q-1])

    return len(set(T)), T

p,q = 3, 10
F_Y = F(10, p, q, key_128, message)[1]
Y = [ord(char) for char in F_Y]

for r in range(21)[1:]:
    F_X = F(r, 3, 10, key_128, message)[1]
    X = [ord(char) for char in F_X]
    random_bytes = [random.randint(0,255) for _ in range(len(X))]
    print r, run_ks(X, Y), run_ks(X, random_bytes)
```