# HW1 Code:

#!/usr/bin/env python3

### Author: Alex Dunker

### ECN: adunker

### HW: 4

### Filename: ece404\_hw04\_dunker.py

### Due Date: 02/05/2016

**from** BitVector **import** **\***

inmessage **=** "plaintext.txt"

encout **=** "encryptedtext.txt"

decout **=** "decryptedtext.txt"

inKey **=** "howtogettosesame"

AES\_modulus **=** BitVector**(**bitstring**=**'100011011'**)**

## These two tables are created with the genTables function seen below and outlined in the notes

subBytesTable **=** **[]**

invSubBytesTable **=** **[]**

kSchedule **=** **[**BitVector**(**size **=** 32**)]** **\*** 44

roundKeys **=** **[**BitVector**(**size **=** 128**)]** **\*** 11

## Function for creating the substitution tables that are hard coded above to save time

**def** genTables**():**

c **=** BitVector**(**bitstring**=**'01100011'**)**

d **=** BitVector**(**bitstring**=**'00000101'**)**

**for** i **in** range**(**0**,** 256**):**

a **=** BitVector**(**intVal **=** i**,** size**=**8**).**gf\_MI**(**AES\_modulus**,**8**)** **if** i **!=** 0 **else** BitVector**(**intVal**=**0**)**

a1**,**a2**,**a3**,**a4 **=** **[**a**.**deep\_copy**()** **for** x **in** range**(**4**)]**

a **^=** **(**a1 **>>** 4**)** **^** **(**a2 **>>** 5**)** **^** **(**a3 **>>** 6**)** **^** **(**a4 **>>** 7**)** **^** c

subBytesTable**.**append**(**int**(**a**))**

b **=** BitVector**(**intVal **=** i**,** size**=**8**)**

b1**,**b2**,**b3 **=** **[**b**.**deep\_copy**()** **for** x **in** range**(**3**)]**

b **=** **(**b1 **>>** 2**)** **^** **(**b2 **>>** 5**)** **^** **(**b3 **>>** 7**)** **^** d

check **=** b**.**gf\_MI**(**AES\_modulus**,** 8**)**

b **=** check **if** isinstance**(**check**,** BitVector**)** **else** 0

invSubBytesTable**.**append**(**int**(**b**))**

## Add the round key to the state

**def** addRoundKey**(**state**,** word**):**

state **^=** word

**return** state

## Substitute the bytes

**def** SubBytes**(**state**):**

**for** i **in** range**(**4**):**

**for** j **in** range**(**4**):**

state**[**j**][**i**]** **=** hex**(**subBytesTable**[**int**(**state**[**j**][**i**],**0**)])**

**return** state

## Shift rows

**def** ShiftRows**(**state**):**

state**[**1**]** **=** state**[**1**][**1**:]+**state**[**1**][:**1**]**

state**[**2**]** **=** state**[**2**][**2**:]+**state**[**2**][:**2**]**

state**[**3**]** **=** state**[**3**][**3**:]+**state**[**3**][:**3**]**

**return** state

## Column mixer, employs GF(2^8) arithmetic

**def** MixColumns**(**state**):**

## The hex multipliers

hex\_01 **=** BitVector**(** hexstring **=** '01' **)**

hex\_02 **=** BitVector**(** hexstring **=** '02' **)**

hex\_03 **=** BitVector**(** hexstring **=** '03' **)**

**for** r **in** range**(**4**):**

**for** c **in** range**(**4**):**

## Convert the hex list to bitvector to use the BitVector module calls

state**[**r**][**c**]** **=** BitVector**(**intVal **=** int**(**state**[**r**][**c**],** 0**))**

**if** len**(**state**[**r**][**c**])** **<** 8**:**

state**[**r**][**c**].**pad\_from\_left**(**8**-**len**(**state**[**r**][**c**]))**

# Create a temporary and perform a deep copy

temp **=** **[[**BitVector**(** size **=** 8 **)** **for** x **in** range**(**4**)]** **for** x **in** range**(**4**)]**

**for** r **in** range**(**4**):**

**for** c **in** range**(**4**):**

temp**[**r**][**c**]** **=** state**[**r**][**c**].**deep\_copy**()**

## Perform the operation

**for** c **in** range**(**4**):**

state**[**0**][**c**]** **=** temp**[**0**][**c**].**gf\_multiply\_modular**(**hex\_02**,** AES\_modulus**,** 8**)** **^** temp**[**1**][**c**].**gf\_multiply\_modular**(**hex\_03**,** AES\_modulus**,** 8**)** **^** temp**[**2**][**c**]** **^** temp**[**3**][**c**]**

state**[**1**][**c**]** **=** temp**[**0**][**c**]** **^** temp**[**1**][**c**].**gf\_multiply\_modular**(**hex\_02**,** AES\_modulus**,** 8**)** **^** temp**[**2**][**c**].**gf\_multiply\_modular**(**hex\_03**,** AES\_modulus**,** 8**)** **^** temp**[**3**][**c**]**

state**[**2**][**c**]** **=** temp**[**0**][**c**]** **^** temp**[**1**][**c**]** **^** temp**[**2**][**c**].**gf\_multiply\_modular**(**hex\_02**,** AES\_modulus**,** 8**)** **^** temp**[**3**][**c**].**gf\_multiply\_modular**(**hex\_03**,** AES\_modulus**,** 8**)**

state**[**3**][**c**]** **=** temp**[**0**][**c**].**gf\_multiply\_modular**(**hex\_03**,** AES\_modulus**,** 8**)** **^** temp**[**1**][**c**]** **^** temp**[**2**][**c**]** **^** temp**[**3**][**c**].**gf\_multiply\_modular**(**hex\_02**,** AES\_modulus**,** 8**)**

## Convert to hex list

**for** r **in** range**(**4**):**

**for** c **in** range**(**4**):**

state**[**r**][**c**]** **=** hex**(**state**[**r**][**c**].**int\_val**())**

**return** state

## Function to convert my stateArrays to bitvectors and hold the proper 4x4 orientation

**def** sAr2BV**(**state**):**

tBV **=** BitVector**(**intVal **=** int**(**state**[**0**][**0**],** 0**))**

**if** len**(**tBV**)** **<** 8**:**

tBV**.**pad\_from\_left**(**8**-(**len**(**tBV**)%**8**))**

bv **=** tBV

**for** c **in** range**(**0**,**4**,**1**):**

**for** r **in** range**(**0**,**4**,**1**):**

**if((**r**+**c**)** **!=** 0**):**

tBV **=** BitVector**(**intVal **=** int**(**state**[**r**][**c**],** 0**))**

**if** len**(**tBV**)** **<** 8**:**

tBV**.**pad\_from\_left**(**8**-(**len**(**tBV**)))**

bv **=** bv **+** tBV

**return** bv

## Rotate word

**def** RotWord**(**word**):**

word**.**circular\_rot\_left**()**

word**.**circular\_rot\_left**()**

word**.**circular\_rot\_left**()**

word**.**circular\_rot\_left**()**

word**.**circular\_rot\_left**()**

word**.**circular\_rot\_left**()**

word**.**circular\_rot\_left**()**

word**.**circular\_rot\_left**()**

**return** word

## Substitute word

**def** SubWord**(**word**):**

**for** i **in** range**(**0**,**32**,**8**):**

temp **=** BitVector**(** intVal **=** subBytesTable**[**word**[**i**:**i**+**8**].**int\_val**()])**

**if** len**(**temp**)** **<** 8**:**

temp**.**pad\_from\_left**(**8**-**len**(**temp**))**

word**[**i**:**i**+**8**]** **=** temp

**return** word

## Create the key schedule

**def** getKeySchedule**(**bv**):**

kArray **=** **[[**BitVector**(**size **=** 8**)** **for** x **in** range**(**4**)]** **for** x **in** range**(**4**)]**

key **=** BitVector**(** textstring **=** inKey**)**

words **=** **[None,** **None,** **None,** **None]**

## Generate round constant bytes

bufferBits **=** BitVector**(**intVal **=** 0**,** size **=** 24**)**

two **=** BitVector**(**intVal **=** 2**,** size **=** 8**)**

roundConstant **=** **[**0**]** **\*** 10

roundConstant**[**0**]** **=** BitVector**(**intVal **=** 1**,** size **=** 8**)**

## Start creating the round constants.

**for** i **in** range**(**1**,**10**):**

roundConstant**[**i**]** **=** roundConstant**[**i**-**1**].**gf\_multiply\_modular**(**two**,** AES\_modulus**,** 8**)**

## Expand them to the proper size

**for** i **in** range**(**10**):**

roundConstant**[**i**]** **+=** bufferBits

**for** i **in** range**(**4**):**

**for** j **in** range**(**4**):**

kArray**[**i**][**j**]** **=** key**[**32**\***i **+** 8**\***j**:**32**\***i**+**8**\*(**j**+**1**)]**

**for** i **in** range**(**4**):**

words**[**i**]** **=** kArray**[**i**][**0**]** **+** kArray**[**i**][**1**]** **+** kArray**[**i**][**2**]** **+** kArray**[**i**][**3**]**

kSchedule**[**i**]** **=** words**[**i**]**

## Create the rest of the words

**for** i **in** range**(**4**,**44**,**1**):**

temp **=** kSchedule**[**i**-**1**].**deep\_copy**()**

**if** i **%** 4 **==** 0**:**

temp **=** RotWord**(**temp**)**

temp **=** SubWord**(**temp**)**

temp **^=** roundConstant**[(**i**/**4**)-**1**]**

kSchedule**[**i**]** **=** temp **^** kSchedule**[**i**-**4**]**

## Assemble the final roundKeys

**for** i **in** range**(**0**,**44**,**4**):**

roundKeys**[**i**/**4**]** **=** kSchedule**[**i**]+**kSchedule**[**i**+**1**]+**kSchedule**[**i**+**2**]+**kSchedule**[**i**+**3**]**

**def** AES\_enc**(**key**,** message**,** fileout**):**

## Get the key

kBV **=** BitVector**(** textstring **=** inKey **)**

## Create the bitvector for the file

bv **=** BitVector**(** filename **=** message **)**

FILEOUT **=** open**(**fileout**,** 'wb'**)**

stateArray **=** **[[**0 **for** x **in** range**(**4**)]** **for** x **in** range**(**4**)]**

nextArray **=** **[[**0 **for** x **in** range**(**4**)]** **for** x **in** range**(**4**)]**

**while(**bv**.**more\_to\_read**):**

## Read 128 bit block

bitvec **=** bv**.**read\_bits\_from\_file**(**128**)**

## Ensure that length is 128, if not pad

**if((**bitvec**.**length**()** **%** 128**)** **!=** 0**):**

bitvec**.**pad\_from\_right**(**128**-(**bitvec**.**length**()** **%** 128**))**

getKeySchedule**(**bitvec**)**

bitvec **=** addRoundKey**(**bitvec**,** roundKeys**[**0**])**

**for** i **in** range**(**4**):**

**for** j **in** range**(**4**):**

nextArray**[**j**][**i**]** **=** hex**(**bitvec**[**32**\***i **+** 8**\***j**:**32**\***i**+**8**\*(**j**+**1**)].**int\_val**())**

**for** x **in** range**(**9**):**

stateArray **=** nextArray

stateArray **=** SubBytes**(**stateArray**)**

stateArray **=** ShiftRows**(**stateArray**)**

stateArray **=** MixColumns**(**stateArray**)**

stateArray **=** addRoundKey**(**sAr2BV**(**stateArray**),** roundKeys**[**x**+**1**])** ##returns bitvector

**for** r **in** range**(**4**):**

**for** c **in** range**(**4**):**

nextArray**[**c**][**r**]** **=** hex**(**stateArray**[**8**\***c**+**32**\***r**:**8**\*(**c**+**1**)+**32**\***r**].**int\_val**())**

stateArray **=** SubBytes**(**nextArray**)**

stateArray **=** ShiftRows**(**stateArray**)**

stateArray **=** addRoundKey**(**sAr2BV**(**stateArray**),** roundKeys**[**10**])**

FILEOUT**.**write**(**stateArray**.**get\_text\_from\_bitvector**())**

FILEOUT**.**close**()**

**def** InvShiftRows**(**state**):**

state**[**1**]** **=** state**[**1**][-**1**:]+**state**[**1**][:-**1**]**

state**[**2**]** **=** state**[**2**][-**2**:]+**state**[**2**][:-**2**]**

state**[**3**]** **=** state**[**3**][-**3**:]+**state**[**3**][:-**3**]**

**return** state

**def** InvSubBytes**(**state**):**

**for** i **in** range**(**4**):**

**for** j **in** range**(**4**):**

state**[**j**][**i**]** **=** hex**(**invSubBytesTable**[**int**(**state**[**j**][**i**],** 16**)])**

**return** state

**def** InvMixColumns**(**state**):**

## Hex codes for gf\_multiplicaiton

hex\_0E **=** BitVector**(** hexstring **=** '0e' **)**

hex\_0B **=** BitVector**(** hexstring **=** '0b' **)**

hex\_0D **=** BitVector**(** hexstring **=** '0d' **)**

hex\_09 **=** BitVector**(** hexstring **=** '09' **)**

**for** r **in** range**(**4**):**

**for** c **in** range**(**4**):**

state**[**r**][**c**]** **=** BitVector**(**intVal **=** int**(**state**[**r**][**c**],** 0**))**

**if** len**(**state**[**r**][**c**])** **<** 8**:**

state**[**r**][**c**].**pad\_from\_left**(**8**-**len**(**state**[**r**][**c**]))**

temp **=** **[[**BitVector**(** size **=** 8 **)** **for** x **in** range**(**4**)]** **for** x **in** range**(**4**)]**

**for** r **in** range**(**4**):**

**for** c **in** range**(**4**):**

temp**[**r**][**c**]** **=** state**[**r**][**c**].**deep\_copy**()**

**for** c **in** range**(**4**):**

state**[**0**][**c**]** **=** temp**[**0**][**c**].**gf\_multiply\_modular**(**hex\_0E**,** AES\_modulus**,** 8**)** **^** temp**[**1**][**c**].**gf\_multiply\_modular**(**hex\_0B**,** AES\_modulus**,** 8**)** **^** temp**[**2**][**c**].**gf\_multiply\_modular**(**hex\_0D**,** AES\_modulus**,** 8**)** **^** temp**[**3**][**c**].**gf\_multiply\_modular**(**hex\_09**,** AES\_modulus**,** 8**)**

state**[**1**][**c**]** **=** temp**[**0**][**c**].**gf\_multiply\_modular**(**hex\_09**,** AES\_modulus**,** 8**)** **^** temp**[**1**][**c**].**gf\_multiply\_modular**(**hex\_0E**,** AES\_modulus**,** 8**)** **^** temp**[**2**][**c**].**gf\_multiply\_modular**(**hex\_0B**,** AES\_modulus**,** 8**)** **^** temp**[**3**][**c**].**gf\_multiply\_modular**(**hex\_0D**,** AES\_modulus**,** 8**)**

state**[**2**][**c**]** **=** temp**[**0**][**c**].**gf\_multiply\_modular**(**hex\_0D**,** AES\_modulus**,** 8**)** **^** temp**[**1**][**c**].**gf\_multiply\_modular**(**hex\_09**,** AES\_modulus**,** 8**)** **^** temp**[**2**][**c**].**gf\_multiply\_modular**(**hex\_0E**,** AES\_modulus**,** 8**)** **^** temp**[**3**][**c**].**gf\_multiply\_modular**(**hex\_0B**,** AES\_modulus**,** 8**)**

state**[**3**][**c**]** **=** temp**[**0**][**c**].**gf\_multiply\_modular**(**hex\_0B**,** AES\_modulus**,** 8**)** **^** temp**[**1**][**c**].**gf\_multiply\_modular**(**hex\_0D**,** AES\_modulus**,** 8**)** **^** temp**[**2**][**c**].**gf\_multiply\_modular**(**hex\_09**,** AES\_modulus**,** 8**)** **^** temp**[**3**][**c**].**gf\_multiply\_modular**(**hex\_0E**,** AES\_modulus**,** 8**)**

**for** r **in** range**(**4**):**

**for** c **in** range**(**4**):**

state**[**r**][**c**]** **=** hex**(**state**[**r**][**c**].**int\_val**())**

**return** state

**def** AES\_dec**(**key**,** message**,** fileout**):**

## Get the key

kBV **=** BitVector**(** textstring **=** inKey **)**

bv **=** BitVector**(** filename **=** message **)**

FILEOUT **=** open**(**fileout**,** 'wb'**)**

stateArray **=** **[[**0 **for** x **in** range**(**4**)]** **for** x **in** range**(**4**)]**

nextArray **=** **[[**0 **for** x **in** range**(**4**)]** **for** x **in** range**(**4**)]**

**while(**bv**.**more\_to\_read**):**

bitvec **=** bv**.**read\_bits\_from\_file**(**128**)**

## Ensure that length is 128, if not pad

**if((**bitvec**.**length**()** **%** 128**)** **!=** 0**):**

bitvec**.**pad\_from\_right**(**128**-(**bitvec**.**length**()** **%** 128**))**

getKeySchedule**(**bitvec**)**

bitvec **=** addRoundKey**(**bitvec**,** roundKeys**[**10**])**

**for** i **in** range**(**4**):**

**for** j **in** range**(**4**):**

nextArray**[**j**][**i**]** **=** hex**(**bitvec**[**32**\***i **+** 8**\***j**:**32**\***i**+**8**\*(**j**+**1**)].**int\_val**())**

## Different process for AES\_dec

**for** x **in** range**(**9**):**

stateArray **=** nextArray

stateArray **=** InvShiftRows**(**stateArray**)**

stateArray **=** InvSubBytes**(**stateArray**)**

stateArray **=** addRoundKey**(**sAr2BV**(**stateArray**),** roundKeys**[**9**-**x**])**

**for** i **in** range**(**4**):**

**for** j **in** range**(**4**):**

nextArray**[**j**][**i**]** **=** hex**(**stateArray**[**32**\***i **+** 8**\***j**:**32**\***i**+**8**\*(**j**+**1**)].**int\_val**())**

nextArray **=** InvMixColumns**(**nextArray**)**

stateArray **=** InvShiftRows**(**nextArray**)**

stateArray **=** InvSubBytes**(**stateArray**)**

stateArray **=** addRoundKey**(**sAr2BV**(**stateArray**),** kBV**)**

FILEOUT**.**write**(**stateArray**.**get\_text\_from\_bitvector**())**

FILEOUT**.**close**()**

**def** main**():**

genTables()

AES\_enc**(**inKey**,** inmessage**,** encout**)**

AES\_dec**(**inKey**,** encout**,** decout**)**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

main**()**

# Input:

This is an unusual paragraph. I'm curious how quickly you can find out what is so unusual about it. It looks so plain you would think nothing is wrong with it. In fact, nothing is wrong with it! It is unusual though. Study it, and think about it, but you still may not find anything odd. But if you work at it a bit, you might find out. Try to do so without any coaching!

# Encrypted Hex Output:



# Decrypted Output:

This is an unusual paragraph. I'm curious how quickly you can find out what is so unusual about it. It looks so plain you would think nothing is wrong with it. In fact, nothing is wrong with it! It is unusual though. Study it, and think about it, but you still may not find anything odd. But if you work at it a bit, you might find out. Try to do so without any coaching!