#! /usr/bin/python

from \_\_future\_\_ import with\_statement

# ignobleepub.pyw, version 3.5

# modified by shani and adapted it for android

\_\_license\_\_ = 'GPL v3'

import sys

import os

class IGNOBLEError(Exception):

pass

def \_load\_crypto\_libcrypto():

from ctypes import CDLL, POINTER, c\_void\_p, c\_char\_p, c\_int, c\_long, \

Structure, c\_ulong, create\_string\_buffer, cast

from ctypes.util import find\_library

import ctypes

libcrypto = ctypes.cdll.LoadLibrary("libcrypto.so")

AES\_MAXNR = 14

c\_char\_pp = POINTER(c\_char\_p)

c\_int\_p = POINTER(c\_int)

class AES\_KEY(Structure):

\_fields\_ = [('rd\_key', c\_long \* (4 \* (AES\_MAXNR + 1))),

('rounds', c\_int)]

AES\_KEY\_p = POINTER(AES\_KEY)

def F(restype, name, argtypes):

func = getattr(libcrypto, name)

func.restype = restype

func.argtypes = argtypes

return func

AES\_cbc\_encrypt = F(None, 'AES\_cbc\_encrypt',

[c\_char\_p, c\_char\_p, c\_ulong, AES\_KEY\_p, c\_char\_p,

c\_int])

AES\_set\_decrypt\_key = F(c\_int, 'AES\_set\_decrypt\_key',

[c\_char\_p, c\_int, AES\_KEY\_p])

AES\_cbc\_encrypt = F(None, 'AES\_cbc\_encrypt',

[c\_char\_p, c\_char\_p, c\_ulong, AES\_KEY\_p, c\_char\_p,

c\_int])

class AES(object):

def \_\_init\_\_(self,userkey,iv):

self.\_blocksize = len(userkey)

self.iv=iv

if (self.\_blocksize != 16) and (self.\_blocksize != 24) and (self.\_blocksize != 32) :

raise IGNOBLEError('AES improper key used')

return

key = self.\_key = AES\_KEY()

rv = AES\_set\_decrypt\_key(userkey, len(userkey) \* 8, key)

if rv < 0:

raise IGNOBLEError('Failed to initialize AES key')

def decrypt(self, data):

out = create\_string\_buffer(len(data))

ivcopy = create\_string\_buffer(self.iv)

rv = AES\_cbc\_encrypt(data, out, len(data), self.\_key, ivcopy , 0)

if rv == 0:

raise IGNOBLEError('AES decryption failed')

return out.raw

return AES