# Author: Trevor Perrin

# See the LICENSE file for legal information regarding use of this file.

"""OpenSSL/M2Crypto 3DES implementation."""

from .cryptomath import \*

from .tripledes import \*

if m2cryptoLoaded:

def new(key, mode, IV):

return OpenSSL\_TripleDES(key, mode, IV)

class OpenSSL\_TripleDES(TripleDES):

def \_\_init\_\_(self, key, mode, IV):

TripleDES.\_\_init\_\_(self, key, mode, IV, "openssl")

self.key = key

self.IV = IV

def \_createContext(self, encrypt):

context = m2.cipher\_ctx\_new()

cipherType = m2.des\_ede3\_cbc()

m2.cipher\_init(context, cipherType, self.key, self.IV, encrypt)

return context

def encrypt(self, plaintext):

TripleDES.encrypt(self, plaintext)

context = self.\_createContext(1)

ciphertext = m2.cipher\_update(context, plaintext)

m2.cipher\_ctx\_free(context)

self.IV = ciphertext[-self.block\_size:]

return bytearray(ciphertext)

def decrypt(self, ciphertext):

TripleDES.decrypt(self, ciphertext)

context = self.\_createContext(0)

#I think M2Crypto has a bug - it fails to decrypt and return the last block passed in.

#To work around this, we append sixteen zeros to the string, below:

plaintext = m2.cipher\_update(context, ciphertext+('\0'\*16))

#If this bug is ever fixed, then plaintext will end up having a garbage

#plaintext block on the end. That's okay - the below code will ignore it.

plaintext = plaintext[:len(ciphertext)]

m2.cipher\_ctx\_free(context)

self.IV = ciphertext[-self.block\_size:]

return bytearray(plaintext)