# Author: Trevor Perrin

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

"""Factory functions for symmetric cryptography."""

import os

from tlslite.utils import python\_aes

from tlslite.utils import python\_rc4

from tlslite.utils import cryptomath

tripleDESPresent = False

if cryptomath.m2cryptoLoaded:

from tlslite.utils import openssl\_aes

from tlslite.utils import openssl\_rc4

from tlslite.utils import openssl\_tripledes

tripleDESPresent = True

if cryptomath.pycryptoLoaded:

from tlslite.utils import pycrypto\_aes

from tlslite.utils import pycrypto\_rc4

from tlslite.utils import pycrypto\_tripledes

tripleDESPresent = True

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Factory Functions for AES

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def createAES(key, IV, implList=None):

"""Create a new AES object.

@type key: str

@param key: A 16, 24, or 32 byte string.

@type IV: str

@param IV: A 16 byte string

@rtype: L{tlslite.utils.AES}

@return: An AES object.

"""

if implList == None:

implList = ["openssl", "pycrypto", "python"]

for impl in implList:

if impl == "openssl" and cryptomath.m2cryptoLoaded:

return openssl\_aes.new(key, 2, IV)

elif impl == "pycrypto" and cryptomath.pycryptoLoaded:

return pycrypto\_aes.new(key, 2, IV)

elif impl == "python":

return python\_aes.new(key, 2, IV)

raise NotImplementedError()

def createRC4(key, IV, implList=None):

"""Create a new RC4 object.

@type key: str

@param key: A 16 to 32 byte string.

@type IV: object

@param IV: Ignored, whatever it is.

@rtype: L{tlslite.utils.RC4}

@return: An RC4 object.

"""

if implList == None:

implList = ["openssl", "pycrypto", "python"]

if len(IV) != 0:

raise AssertionError()

for impl in implList:

if impl == "openssl" and cryptomath.m2cryptoLoaded:

return openssl\_rc4.new(key)

elif impl == "pycrypto" and cryptomath.pycryptoLoaded:

return pycrypto\_rc4.new(key)

elif impl == "python":

return python\_rc4.new(key)

raise NotImplementedError()

#Create a new TripleDES instance

def createTripleDES(key, IV, implList=None):

"""Create a new 3DES object.

@type key: str

@param key: A 24 byte string.

@type IV: str

@param IV: An 8 byte string

@rtype: L{tlslite.utils.TripleDES}

@return: A 3DES object.

"""

if implList == None:

implList = ["openssl", "pycrypto"]

for impl in implList:

if impl == "openssl" and cryptomath.m2cryptoLoaded:

return openssl\_tripledes.new(key, 2, IV)

elif impl == "pycrypto" and cryptomath.pycryptoLoaded:

return pycrypto\_tripledes.new(key, 2, IV)

raise NotImplementedError()