CSA5122-CRYPTOGRAPHY FOR NETWORK AND SECURITY

LAB PROGRAMS EXECUTION

12. BLOWFISH ALGORITHM

import struct

# Truncated Blowfish P-array and S-boxes for demonstration only!

P = [

0x243F6A88, 0x85A308D3, 0x13198A2E, 0x03707344,

0xA4093822, 0x299F31D0, 0x082EFA98, 0xEC4E6C89,

0x452821E6, 0x38D01377, 0xBE5466CF, 0x34E90C6C,

0xC0AC29B7, 0xC97C50DD, 0x3F84D5B5, 0xB5470917,

0x9216D5D9, 0x8979FB1B

]

S = [

[0xD1310BA6, 0x98DFB5AC, 0x2FFD72DB, 0xD01ADFB7] \* 64, # 256 elements

[0xB8E1AFED, 0x6A267E96, 0xBA7C9045, 0xF12C7F99] \* 64,

[0x24A19947, 0xB3916CF7, 0x0801F2E2, 0x858EFC16] \* 64,

[0x636920D8, 0x71574E69, 0xA458FEA3, 0xF4933D7E] \* 64,

]

def F(x):

a = (x >> 24) & 0xFF

b = (x >> 16) & 0xFF

c = (x >> 8) & 0xFF

d = x & 0xFF

# Each S-box has 256 elements

return (((S[0][a] + S[1][b]) & 0xFFFFFFFF) ^ S[2][c]) + S[3][d] & 0xFFFFFFFF

def blowfish\_encrypt(block, P, S):

left, right = struct.unpack(">2I", block)

for i in range(16):

left ^= P[i]

left &= 0xFFFFFFFF

right ^= F(left)

right &= 0xFFFFFFFF

left, right = right, left

left, right = right, left # Undo last swap

right ^= P[16]

right &= 0xFFFFFFFF

left ^= P[17]

left &= 0xFFFFFFFF

return struct.pack(">2I", left, right)

def blowfish\_decrypt(block, P, S):

left, right = struct.unpack(">2I", block)

for i in range(17, 1, -1):

left ^= P[i]

left &= 0xFFFFFFFF

right ^= F(left)

right &= 0xFFFFFFFF

left, right = right, left

left, right = right, left # Undo last swap

right ^= P[1]

right &= 0xFFFFFFFF

left ^= P[0]

left &= 0xFFFFFFFF

return struct.pack(">2I", left, right)

def pad\_data(data):

pad\_length = 8 - (len(data) % 8)

return data + bytes([pad\_length] \* pad\_length)

def unpad\_data(data):

pad\_length = data[-1]

return data[:-pad\_length]

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

plaintext = b"HelloBlowfish"

padded\_plaintext = pad\_data(plaintext)

# Encrypt each 8-byte block

encrypted = b""

for i in range(0, len(padded\_plaintext), 8):

block = padded\_plaintext[i:i + 8]

encrypted += blowfish\_encrypt(block, P, S)

# Decrypt the data

decrypted = b""

for i in range(0, len(encrypted), 8):

block = encrypted[i:i + 8]

decrypted += blowfish\_decrypt(block, P, S)

decrypted = unpad\_data(decrypted)

print("Encrypted (hex):", encrypted.hex())

print("Decrypted:", decrypted.decode('utf-8'))

