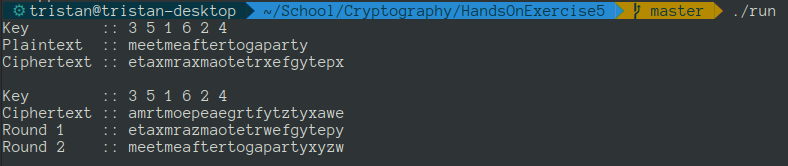
Tristan Erney

October 5th, 2021

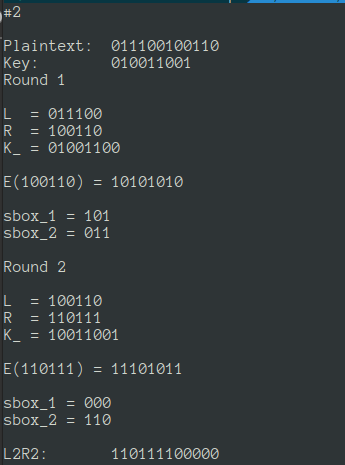
Intro to Cryptology

Hands on Exercise 5

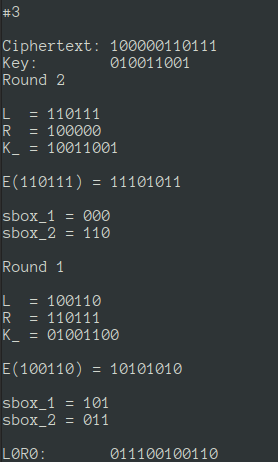
1)



2)



3)



TranspositionCipher.cpp

#include <iostream>

#include <vector>

#include <stdlib.h>

#include <string.h>

// expand string by adding 'x''s to the end if needed

char\* expand\_string(char\* str, int desired\_size) {

int str\_size = (int)strlen(str);

char\* s = (char\*)malloc(sizeof(char) \* desired\_size + 1);

s[desired\_size] = 0;

for (int i = 0; i < desired\_size; i += 1) {

s[i] = (i < str\_size) ? str[i] : 'x';

}

return s;

}

/\*

Row Transposition Encryption:

encrypt by building a table of (k\_size x (c\_size / k\_size))

and rebuild a new ciphertext string by reading the table

by column.

meetmeaftertogaparty

key = {3, 5, 1, 6, 2, 4}

TBL:

m e e t m e

a f t e r t

o g a p a r

t y x y z w

-----------

3 5 1 6 2 4

etaxmrazmaotetrwefgytepy

\*/

char\* row\_transposition\_encrypt(char\* \_P, int\* K, int k\_size) {

size\_t p\_size = strlen(\_P);

size\_t pad = k\_size - (p\_size % k\_size);

size\_t c\_size = p\_size + pad;

char\* P = expand\_string(\_P, c\_size);

char\* C = (char\*)malloc(c\_size + 1);

C[c\_size] = 0;

char\*\* TBL = (char\*\*)malloc(sizeof(char\*) \* (c\_size / k\_size));

for (int i = 0; i < (c\_size / k\_size); i += 1) {

char\* p\_ptr = P + (i \* k\_size);

TBL[i] = (char\*)malloc(sizeof(char) \* k\_size);

for (int j = 0; j < k\_size; j += 1) {

TBL[i][j] = p\_ptr[j];

}

}

int k = 0;

for (int i = 0; i < k\_size; i += 1) {

for (int j = 0; j < (c\_size / k\_size); j += 1) {

C[k++] = TBL[j][K[i] - 1];

}

}

return C;

}

/\*

Row Transposition Decrypt:

decrypt by building a table of (k\_size x (c\_size / k\_size))

by reading in the table by each (c\_size / k\_size) sized chunk

ordered by the key, which is then read into a new plaintext

string by reading the table left-right top-bottom.

key = {3, 5, 1, 6, 2, 4}

#3 #5 #1 #6 #2 #4

amrt moep eaeg rtfy tzty xawe

e t a x m r

a z m a o t

e t r w e f

g y t e p y

etaxmrazmaotetrwefgytepy

\*/

char\* row\_transposition\_decrypt(char\* C, int\* K, int k\_size) {

size\_t c\_size = strlen(C);

size\_t p\_size = c\_size;

char\* P = (char\*)malloc(p\_size + 1);

P[p\_size] = 0;

// create table

char\*\* TBL = (char\*\*)malloc(sizeof(char\*) \* (c\_size / k\_size));

for (int i = 0; i < (c\_size / k\_size); i += 1) {

char\* c\_ptr = C + (i \* k\_size);

TBL[i] = (char\*)malloc(sizeof(char) \* k\_size);

}

// read data into table

for (int i = 0; i < k\_size; i += 1) {

char\* c\_ptr = C + (i \* (c\_size / k\_size));

for (int j = 0; j < (c\_size / k\_size); j += 1) {

TBL[j][K[i] - 1] = c\_ptr[j];

}

}

int k = 0;

for (int i = 0; i < (c\_size / k\_size); i += 1) {

for (int j = 0; j < k\_size; j += 1) {

P[k++] = TBL[i][j];

}

}

return P;

}

int main() {

{

int key[] = {3, 5, 1, 6, 2, 4};

char P[] = "meetmeaftertogaparty";

char\* C = row\_transposition\_encrypt(P, key, 6);

std::cout << "Key :: ";

for (int i = 0; i < (sizeof(key) / sizeof(key[0])); i += 1) {

std::cout << key[i] << " ";

}

std::cout << "\n";

std::cout << "Plaintext :: " << P << "\n";

std::cout << "Ciphertext :: " << C << "\n\n";

}

{

int key[] = {3, 5, 1, 6, 2, 4};

char C[] = "amrtmoepeaegrtfytztyxawe";

char\* R1 = row\_transposition\_decrypt(C, key, 6);

char\* R2 = row\_transposition\_decrypt(R1, key, 6);

std::cout << "Key :: ";

for (int i = 0; i < (sizeof(key) / sizeof(key[0])); i += 1) {

std::cout << key[i] << " ";

}

std::cout << "\n";

std::cout << "Ciphertext :: " << C << "\n";

std::cout << "Round 1 :: " << R1 << "\n";

std::cout << "Round 2 :: " << R2 << "\n\n";

}

return 0;

}

S\_DES.cpp

#include <iostream>

#include <iomanip>

#include <bitset>

#include <stdlib.h>

#include <stdint.h>

void printb3(uint8\_t b) {

std::bitset<3> x(b);

std::cout << x;

}

void printb6(uint8\_t b) {

std::bitset<6> x(b);

std::cout << x;

}

void printb9(uint16\_t b) {

std::bitset<9> x(b);

std::cout << x;

}

void printb12(uint16\_t b) {

std::bitset<12> x(b);

std::cout << x;

}

void printb(uint8\_t b) {

std::bitset<8> x(b);

std::cout << x;

}

uint8\_t sbox1[2][8] = {

{ 5, 2, 1, 6, 3, 4, 7, 0 },

{ 1, 4, 6, 2, 0, 7, 5, 3 }

};

uint8\_t sbox2[2][8] = {

{ 4, 0, 6, 5, 7, 1, 3, 2 },

{ 5, 3, 0, 7, 6, 2, 1, 4 }

};

uint8\_t E(uint8\_t X) {

uint8\_t e = 0;

e |= ((0x20 & X) >> 5) << 7;

e |= ((0x10 & X) >> 4) << 6;

e |= ((0x04 & X) >> 2) << 5;

e |= ((0x08 & X) >> 3) << 4;

e |= ((0x04 & X) >> 2) << 3;

e |= ((0x08 & X) >> 3) << 2;

e |= ((0x02 & X) >> 1) << 1;

e |= ((0x01 & X) >> 0) << 0;

std::cout << "E(";

printb6(X);

std::cout << ") = ";

printb(e);

std::cout << "\n\n";

return e;

}

uint8\_t F(uint8\_t X, uint8\_t K) {

uint8\_t e = E(X);

uint8\_t e\_xor\_k = e ^ K;

uint8\_t sbox\_1 = sbox1[e\_xor\_k >> 7][0x07 & (e\_xor\_k >> 4)];

uint8\_t sbox\_2 = sbox2[(0x0f & e\_xor\_k) >> 3][0x07 & e\_xor\_k];

std::cout << "sbox\_1 = ";

printb3(sbox\_1);

std::cout << "\n";

std::cout << "sbox\_2 = ";

printb3(sbox\_2);

std::cout << "\n\n";

return (sbox\_1 << 3) + sbox\_2;

}

uint16\_t sdes\_encrypt(uint16\_t P, uint16\_t K) {

uint8\_t round = 0;

uint8\_t max\_rounds = 2;

// P = 0bxxxx |oooo oo|oo oooo|

uint8\_t L = (uint8\_t)(P >> 6);

uint8\_t R = (uint8\_t)(0x3f & P);

for (round = 0; round < max\_rounds; round += 1) {

std::cout << "Round " << round + 1 << "\n\n";

std::cout << "L = ";

printb6(L);

std::cout << "\n" << "R = ";

printb6(R);

std::cout << "\n";

uint8\_t K\_ = ((K << round) | (K >> 9 - round)) >> 1;

std::cout << "K\_ = ";

printb(K\_);

std::cout << "\n\n";

uint8\_t L\_ = R;

uint8\_t R\_ = L ^ F(R, K\_);

L = L\_;

R = R\_;

}

return (L << 6) + R;

}

uint16\_t sdes\_decrypt(uint16\_t C, uint16\_t K) {

int8\_t round = 1;

uint8\_t max\_rounds = 0;

uint8\_t R = (uint8\_t)(C >> 6);

uint8\_t L = (uint8\_t)(0x3f & C);

for (round = 1; round >= max\_rounds; round -= 1) {

std::cout << "Round " << round + 1 << "\n\n";

std::cout << "L = ";

printb6(L);

std::cout << "\n" << "R = ";

printb6(R);

std::cout << "\n";

uint8\_t K\_ = ((K << round) | (K >> 9 - round)) >> 1;

std::cout << "K\_ = ";

printb(K\_);

std::cout << "\n\n";

uint8\_t R\_ = L;

uint8\_t L\_ = R ^ F(R\_, K\_);

L = L\_;

R = R\_;

}

return (L << 6) + R;

}

int main() {

{

std::cout << "#2\n\n";

uint16\_t K = 0x0099; // 0 1001 1001

uint16\_t P = 0x0726; // 011100 100110

std::cout << "Plaintext: ";

printb12(P);

std::cout << "\n";

std::cout << "Key: ";

printb9(K);

std::cout << "\n";

uint16\_t C = sdes\_encrypt(P, K);

std::cout << "L2R2: ";

printb12(C);

std::cout << "\n";

}

{

std::cout << "#3\n\n";

uint16\_t K = 0x0099;

uint16\_t C = 0x0837;

std::cout << "Ciphertext: ";

printb12(C);

std::cout << "\n";

std::cout << "Key: ";

printb9(K);

std::cout << "\n";

uint16\_t P = sdes\_decrypt(C, K);

std::cout << "L0R0: ";

printb12(P);

std::cout << "\n";

}

return 0;

}