Tristan Erney

September 21st, 2021

Hands on Exercise 3

1) plaintext = eveexpectseggsforbreakfast



2)



3)



FrequencyAnalysis.cpp:

#include <iostream>

#include <iomanip>

#include <stdlib.h>

#include <string.h>

struct CipherFreqTable {

char letter[26];

float freq[26];

};

int mod(int x, int y) {

if (x < 0) {

while (x < 0) x += y;

return x;

} else {

return x % y;

}

}

CipherFreqTable\* letter\_freq(const char\* p) {

CipherFreqTable\* t = new CipherFreqTable;

for (int i = 0; i < 26; i += 1) {

t->letter[i] = 'A' + i;

t->freq[i] = 0;

}

for (int i = 0; i < strlen(p); i += 1) {

int j = p[i] - 'A';

t->freq[j] += 1;

}

return t;

}

char\* shift\_decrypt(const char\* c, int k) {

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

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

p[c\_size] = 0;

for (size\_t i = 0; i < c\_size; i += 1) {

p[i] = mod((c[i] - 'A') - k, 26) + 'a';

}

return p;

}

int main() {

const char\* ciphertext = "LCLLEWLJAZLNNZMVYIYLHRMHZA";

// frequency analysis

int k = -1;

CipherFreqTable\* freq\_table = letter\_freq(ciphertext);

int score = 0;

char letter = ' ';

for (size\_t j = 0; j < 26; j += 1) {

if (score < freq\_table->freq[j]) {

score = freq\_table->freq[j];

letter = freq\_table->letter[j];

}

}

letter += 32;

std::cout << "Most Frequent Letter = " << letter << "\n";

k = letter - 'e';

char\* plaintext = shift\_decrypt(ciphertext, k);

std::cout << "Key == " << k << "\n";

std::cout << "Plaintext == " << shift\_decrypt(ciphertext, k) << "\n";

return 0;

}

Vigenere.cpp:

#include <iostream>

#include <iomanip>

#include <string.h>

int mod(int x, int y) {

if (x < 0) {

while (x < 0) x += y;

return x;

} else {

return x % y;

}

}

char\* vigenere\_encrypt(const char\* p, const char\* k) {

int p\_size = (int)strlen(p),

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

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

c[p\_size] = 0;

for (size\_t i = 0; i < p\_size; i += 1) {

c[i] = mod(((p[i] - 'a') + (k[i % k\_size] - 'a')), 26) + 'A';

}

return c;

}

char\* vigenere\_decrypt(const char\* c, const char\* k) {

int c\_size = (int)strlen(c),

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

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

p[c\_size] = 0;

for (size\_t i = 0; i < c\_size; i += 1) {

p[i] = mod(((c[i] - 'A') - (k[i % k\_size] - 'a')), 26) + 'a';

}

return p;

}

int main() {

const char\* p = "explanation";

const char\* k = "leg";

const char\* c = vigenere\_encrypt(p, k);

std::cout << "plaintext = " << p << "\n";

std::cout << "key = " << k << "\n";

std::cout << "ciphertext = " << c << "\n";

std::cout << "check = " << vigenere\_decrypt(c, k) << "\n";

return 0;

}

One\_Time\_Pad.cpp:

#include <iostream>

#include <string.h>

int mod(int x, int y) {

if (x < 0) {

while (x < 0) x += y;

return x;

} else {

return x % y;

}

}

char\* one\_time\_pad\_decrypt(const char\* c, const char\* k) {

int size = strlen(c);

char\* p = (char\*)malloc(sizeof(c) \* size + 1);

p[size] = 0;

for (int i = 0; i < size; i += 1) {

p[i] = mod((c[i] - 'A') - (k[i] - 'a'), 26) + 'a';

}

return p;

}

int main() {

const char\* c = "WOJDMNVHIC";

const char\* k1 = "pkysyhkdvp";

const char\* k2 = "rgszmuzzxr";

const char\* p1 = one\_time\_pad\_decrypt(c, k1);

const char\* p2 = one\_time\_pad\_decrypt(c, k2);

std::cout << "p1 = one\_time\_pad\_decrypt(" << c << ", " << k1 << ") := " << p1 << "\n";

std::cout << "p2 = one\_time\_pad\_decrypt(" << c << ", " << k2 << ") := " << p2 << "\n";

return 0;

}