Titkosírás haladóknak II.

(Cipher - exercises for advanced users - Part II)

Pitlik Marcell, Pitlik László, Pitlik Mátyás, Pitlik László (jun) – MY-X team

Kivonat: A KNUTH-i elv értelmében dekódolásról akkor beszélhetünk, ha a teljes folyamat programozott képességként kerül felkínálásra. Vagyis az, hogy emberi intuícióval (találgatással) adott üzenet megfejtettnek minősül, még nem felel meg annak az elvárásnak, hogy a dekódolás hibátlanul képes futni automatikusan. Jelen cikk egy három részes sorozat második eleme, ahol az első cikk a titkosítási eljárást már bemutatta. Itt és most a visszafejtő/dekódoló program kerül bemutatásra, míg a harmadik cikk a kódoló/titkosító eljárás finomhangolásának elvi és gyakorlati vonatkozásait mutatja be. A cikksorozat célja a gamifikált tanulás támogatása.

Kulcsszavak: titkosírás/kódfejtés tervezése, titkosírások nehézségi szintjei

Abstract: Based on the principle of KNUTH, it is just possible to speak about deciphering, if the whole process could be transferred into source code and therefore it can be used in an automatically way. It means the intuitive solution (guessing) is rather a kind of artistic performance as knowledge – independent from its success. This article is the second part of a series having three parts altogether. The first part already presented the ciphering algorithm. Here and now, it is possible to see the deciphering algorithm. The third part will demonstrate the same functionality of ciphering but instead of a solution created by a less trained programmer, the last part will present a more professional (optimized/efficient) source code. One of the goals of the article is the support the gamified learning.

Keywords: planning of de/ciphering, levels of ciphering

# Bevezetés

Ahhoz, hogy a titkosírások magasabb szintjén elvárt szempontok pl.

* szóhatárok felismerhetetlensége
* adott input karakter sokféle alternatív (eltérő hosszúságú) jelsorozattal való helyettesítése
* véletlenhatások bevonása
* adott üzenet alternatív megoldásokkal való lefedni tudása
* felesleges jelek (véletlenszerű) elhelyezése, stb.

algoritmikusan érvényesülni tudjanak, a titkosítás tervezésekor azonnal a visszafejtés mechanizmusait is tervezni kell.

# A dekódoló eljárás

Az alábbi eljárás a cikksorozat első részében bemutatott titkosítás visszafejtésének tapasztalatlan programozó által írt kódja:

#include <iostream>

#include <sstream>

#include <numeric>

#include <bitset>

#include <map>

using namespace std;

int main() {

 string titkod;

 int inits [12];

 int forgatas = 0;

 cout << "Mi a kodolt uzenet?" << endl;

 getline (cin, titkod);

 //cout << titkod << endl;

 if (titkod[0] != '0' || titkod[11] != '1') {

 goto ERR;

 }

 for (int i=0; i<12; i++) {

 inits[i] = titkod[i] - '0';

 if (inits[i] == 0 || inits[i] == 1) continue;

 else goto ERR;

 }

 if (accumulate(inits, inits+12, 0) == 1) {

 cout << endl << "Esedekesseg:\tISMERETLEN" << endl;

 }

 else if (accumulate(inits, inits+12, 0) < 10) {

 stringstream ss;

 string ho, nap1, nap2;

 for (int i=1; i<5; i++) {

 ss << inits[i];

 }

 ss >> ho;

 if (bitset<4>(ho).to\_ulong() > 12) {

 goto ERR;

 }

 stringstream().swap(ss);

 ss << inits[5] << inits[6];

 ss >> nap1;

 stringstream().swap(ss);

 for (int i=7; i<11; i++) {

 ss << inits[i];

 }

 ss >> nap2;

 if (bitset<4>(nap2).to\_ulong() > 9) {

 goto ERR;

 }

 cout << endl << "Esedekesseg:\t" << bitset<4>(ho).to\_ulong() << "/" << bitset<2>(nap1).to\_ulong() << bitset<4>(nap2).to\_ulong() << endl;

 }

 else {

 //cout << "teszt: lefutott" << endl;

 ERR:cout << "ERROR: MSG HDR COMPROMISED!" << endl;

 exit(1);

 }

 stringstream uzenet;

 for (int i=12; i<titkod.length(); i++) {

 char c = titkod[i];

 if (c == '0' || c == '1') {

 forgatas++;

 }

 else {

 if (i>14 && titkod[i-3] == '0' && titkod[i-2] == '0' && titkod[i-1] == '0') {

 if (c == '7' || c == '9') {

 if (titkod[i-4] == '0' && i>15) {

 uzenet << " ";

 } else goto MEGIS\_BETU;

 } else {

 uzenet << " ";

 }

 } else {

 MEGIS\_BETU:if (c == '2') {

 map<int,char> tabla2;

 tabla2[0]= 'a';

 tabla2[1]= 'b';

 tabla2[2]= 'c';

 uzenet << tabla2[forgatas%3];

 }

 else if (c == '3') {

 map<int,char> tabla3;

 tabla3[0]= 'd';

 tabla3[1]= 'e';

 tabla3[2]= 'f';

 uzenet << tabla3[forgatas%3];

 }

 else if (c == '4') {

 map<int,char> tabla4;

 tabla4[0]= 'g';

 tabla4[1]= 'h';

 tabla4[2]= 'i';

 uzenet << tabla4[forgatas%3];

 }

 else if (c == '5') {

 map<int,char> tabla5;

 tabla5[0]= 'j';

 tabla5[1]= 'k';

 tabla5[2]= 'l';

 uzenet << tabla5[forgatas%3];

 }

 else if (c == '6') {

 map<int,char> tabla6;

 tabla6[0]= 'm';

 tabla6[1]= 'n';

 tabla6[2]= 'o';

 uzenet << tabla6[forgatas%3];

 }

 else if (c == '7') {

 map<int,char> tabla7;

 tabla7[0]= 'p';

 tabla7[1]= 'q';

 tabla7[2]= 'r';

 tabla7[3]= 's';

 uzenet << tabla7[forgatas%4];

 }

 else if (c == '8') {

 map<int,char> tabla8;

 tabla8[0]= 't';

 tabla8[1]= 'u';

 tabla8[2]= 'v';

 uzenet << tabla8[forgatas%3];

 }

 else if (c == '9') {

 map<int,char> tabla9;

 tabla9[0]= 'w';

 tabla9[1]= 'x';

 tabla9[2]= 'y';

 tabla9[3]= 'z';

 uzenet << tabla9[forgatas%4];

 } else {

 uzenet << "\*err\*";

 }}

 }

 }

 cout << "Uzenet:" << endl << '"' << uzenet.str() << '"' << endl;

}

# Üzenetek

A cikksorozat első részében jelzett programozói gondolatok titkosítás és visszafejtése tehát lehetséges:





# Konklúziók

A dekódolás és a titkosítás egyszerre, egymással párban fejlesztendő. Sajnos ez a gondolkodásmód csak azt garantálja, hogy a titkosítás és a visszafejtés mindvégig egymásnak megfelelő és kevésbé mozgatja meg az ismeretlen titkosírások visszafejtéséhez szükséges agyi területeket. Így a gamifikáció ilyen formája csak korlátozottan támogatja az innovativitást. Szerencsés esetben a valaki által kialakított titkosítások megértése és a visszafejtés algoritmikusságának garantálni tudása legalább új titkosítási koncepciók fejlesztését képes lehet katalizálni.

A három rész egyben is letölthető: <https://miau.my-x.hu/miau/253/cipher1-2-3.docx>