STRUČNI ČLANCI
PROFESSIONAL PAPERS

UPOTREBA RAČUNARA ZA DEŠIFROVANJE PORUKA

Radiša R. Stefanović, *Aleksa* S. Srdanov
Visoka škola strukovnih studija, Požarevac

DOI: 10.5937/vojtehg62-3831

OBLAST: matematika, informatika
VRSTA ČLANKA: stručni članak

Sažetak:

Kriptografija je danas, kao i u prošlosti, veoma aktuelna. Neprekidno postoji potreba da se zaštite podaci i resursi od neovlašćenog upada i korišćenja. Rezultat toga je nastajanje sve jačih zaštita i probijanje većine onih za koje se do nedavno mislilo da su neprobojne. U ovom radu analiziraju se osnovni problemi i njihove implementacije u okviru razbijanja translatornog šifrata koristeći programski jezik C.

Ključne reči: *kriptografija*, *šifrovanje, translatorni šifrat.*

Uvod

Nekada se sa pojavom prvih računara smatralo da je neka šifra neprobojna ako je računaru potrebno, recimo, godina da sirovom pretragom svih slučajeva pronađe rešenje. Nije prošlo ni tridesetak godina od tada, a računari taj isti posao mogu da završe u nekoliko minuta. Šta je sada neprobojna šifra? Mogućnosti i brzine računara su tolike da nijedan posao više nije problem za njih. Ipak, kombinacija čoveka i računara su tandem koji izgleda nema premca (Markagić, 2012, pp.250-265).

Postoji veliki broj različitih vrsta kriptografskih zadataka. Neki od njih su:

– kriptogrami. Dat je šifrovani tekst koji treba dešifrovati. Takvi problemi najčešće se upotrebljavaju u vojnoj, diplomatskoj i ilegalnoj aktivnosti (Kuljanski, 2010, pp.65-77);

e-mail: radisastefanovic@yahoo.com

– kriptoritmi. To je posebna vrsta matematičkih zadataka u kojima slova treba zameniti ciframa i dobiti tačne relacije koje su navedene u postavci;

– logički kriptogrami. Predstavljaju različite matematičko-logičke zadatke u kojima treba naći zamenu korišćenih simbola sa nekim drugim, tako da polazna postavka promeni izgled, ali ne izgubi smisao. Odnosno, dobija se smislena struktura koja ima drugačiji zapis.

Osnovni problemi

Šifrovanje poruka oslobođeno je od svih pravila. Tako gledano, nije moguće napraviti univerzalni algoritam koji bi svaki šifrat mogao dešifrovati (Aho, Hipcroft, Ullman, 1978). Upotreba grube sile ili pretraga svih mogućih slučajeva često nije moguća. Računar u slučaju da se ne radi o nekom pasvordu, koji može proveriti, ne može biti siguran da je posao odradio (Schneier, 1996). Dobijeni dešifrovani tekst ima smisla samo čoveku. Isključiti čoveka iz postupka dešifrovanja još uvek je nemoguće. Čovek je taj koji smišlja alate sa kojima računar samo fizički odradi posao koji bi čoveku bio nemoguća misija. Osnovni problemi su sledeći:

– osmisliti alate kojima će se pokušati razbiti neki šifrat (Scientific American Magazine 2005, pp.65-69),

– neprekidno pratiti proces i donositi odgovarajuće odluke. Čoveka nikako ne treba isključivati iz procesa dešifrovanja, jer kompjuter, čak i ako razbije šifru, ne može to u opštem slučaju sigurno zaključiti,

– implementirati alate unutar nekog programskog jezika (Evseev, et al., 2011, pp.15-39).

Prvi primer

Nedužni osuđenik na smrt čeka izvršenje smrtne kazne. Za to vreme njegov advokat odbrane dobija tajnu poruku i ako uspe da je dešifruje može spasti svog klijenta od nepravednog pogubljenja. Tekst tajne poruke glasi:

ŠTCŽJUMQQŽOLONŠDEHGLLDNURUVDJXDLTYWČĆRLSJNĆNQOJ
UYSŠDULPLTTSRZRENŽYUVGSJKYQQGJFZČEJŠHXFŠZNRKMCSV
ČUČJNMMĆFWŠŠČQUVŠULRČEŽBFIUGYTHVWGLPFMQGSJMIFCP
MNČRCRĐČORTYBQIRRVBYGČPŠPUYGGSOČVPŠUOHKRTDNWS
MCĐFFRLDNORZQKPVSUČXJĐLNČDDOWVNGŠCOĆUVNKHVWLDK
HGLTNLRNLTJQRXSKWVMIPTŽPMMPRVBYMPKRFQSPQVUHKČ.

Analiza problema za prvi primer

Ovakav zadatak nemoguće je rešiti u opštem obliku. Takođe, nije realno da jedan program može dešifrovati svaki tekst. Pretpostavićemo da se u ovom slučaju radi o translatu (Opplinger, 2005). Dakle, u šifrovanju

je korišćena tehnika sa brojem određene dužine koji se postavlja na tekst, pa se svako slovo u originalnom tekstu translira za broj koji na njega pokazuje. Ako slučajno nije tako, moralo bi se nešto drugo probati.

Ovakav zadatak rešava niz pomoćnih programa kao alata, a ne jedan program, jer se ne može znati koji korak će se koristiti posle nekog za koji se misli da bi bio usmeren ka cilju. Prvo ćemo napisati program koji određuje dužine između pojavljivanja istih slova, jer one mogu poslužiti kao dobar početak za nalaženje dužine šifrata, odnosno niza brojeva sa kojima se slova transliraju (Hrg, Budin, Golub, 2004). U ugnježdenim ciklusima računaće se pozicije pojavljivanja svih slova po abecednom redosledu i štampati odgovarajući izveštaj. Dati šifrovani tekst smestićemo u znakovni niz.

Ukoliko se pojavi problem naših latiničnih slova umesto znakova treba koristiti standardne ASCII kodove date sa:

Ž - 64
Š - 91
Đ - 92
Č - 93
Ć - 94.

Prvi algoritamski alat mogao bi imati sledeći oblik:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
const char s[300] =
{'Š','T','C','Ž','J','U','M','Q','Q','Ž','O','L','O','N','Š','D','E','H','G','L','L','D',
'N','U','R','U','V','D','J','X','D','L','T','Y','W','Č','Ć','R','L','S','J','N','Ć','N','
Q','O','J','U','Y','S','Š','D','U','L','P','L','T','T','S','R','Z','R','E','N','Ž','Y','U'
,'V','G','S','J','K','Y','Q','Q','G','J','F','Z','Č','E','J','Š','H','X','F','Š','Z','N','
R','K','M','C','S','V','Č','U','Č','J','N','M','M','Ć','F','W','Š','Š','Č','Q','U','V',
'Š','U','L','R','Č','E','Ž','B','F','I','U','G','Y','T','H','V','W','G','L','P','F','M','
Q','G','S','J','M','I','F','C','P','M','N','Č','R','C','R','Đ','Č','O','R','T','Y','B','
Q','I','R','R','V','B','Y','G','Č','P','Š','P','U','Y','G','G','S','O','Č','V','P','Š','U
','O','H','K','R','T','D','N','W','S','M','C','Đ','F','F','R','L','D','N','O','R','Z','Q
','K','P','V','S','U','Č','X','J','Đ','L','N','Č','D','D','O','W','V','N','G','Š','C','O
','Ć','U','V','N','K','H','V','W','L','D','K','H','G','L','T','N','L','R','N','L','T','J',
'Q','R','X','S','K','W','V','M','I','P','T','Ž','P','M','M','P','R','V','B','Y','M','P','
K','R','F','Q','S','P','Q','V','U','H','K','Č'};
const char a[32] =
{'A','B','C','Č','Ć','D','Đ','E','F','G','H','I','J','K','L','M','N','O','P','Q','R
','S','Š','T','U','V','F','Z','Ž','X','Y','W'};
short  k, m, n;
```

```
short p[32][30] = {0};
for ( n = 0; n < 32; n++)
{
    m = 0;
    for (k = 0; k < 280; k++)
      if (a[n] == s[k])
       {  p[n][m]  = k+1;
          m++; }
}
for (m = 0; m < 32; m++)
{  printf("%c :", a[m]);
  for ( n = 1; n < 20; n++)
  if (p[m][n]-p[m][0] > 1) printf("%2d ", p[m][n]-p[m][0]);
  printf("\n");}
system("PAUSE");
return 0;
}
```

Posle startovanja ovog programa na ekranu se dobije sledeći ispis:

A :
B : 36 42 144
C : 90 138 144 186 218
Ć :
Č :
D : 6 12 15 36 168 179 197 198 216
Đ :
E : 46 64 100
F : 8 26 42 54 62 113 114 191
G : 50 57 104 110 116 144 151 152 200 216
H : 66 108 162 210 216 258
I : 18 36 132
J : 24 36 42 66 72 77 94 132 203 239
K : 19 109 129 155 161 177 195 205
L : 8 9 20 27 42 44 102 118 182 198 219 224 227 230
M : 85 94 95 126 131 136 181 245 251 252 258
N : 9 28 30 50 75 86 130 171 182 197 204 212 224 227
O : 2 35 140 162 168 186 204 211
P : 76 87 110 112 121 147 199 202 205 211 217
Q :37 66 67 101 126 148 192 237 262 265
R :13 35 37 65 90 121 123 127 133 134 157 168 173 215 221 236 243
S :10 19 30 54 96 132 147 164 208 231
Š :14 50 82 86 105 106 111 165 176 219
T :31 55 56 123 151 181 235 241 253
U :18 20 42 47 61 91 104 107 116 162 172 199 218 269

```
V :41 68 84 100 133 148 176 190 198 202 224 235 247
F : 8 26 42 54 62 113 114 191
Z :18 27 138
Ž : 6 61 114 252
X :55 177 217
Y :15 32 39 90 120 128 135 230
W :70 93 151 181 195 215
Press any key to continue . . .
```

U prethodnom ispisu najviše pažnje treba posvetiti onim slovima u kojima se pojavljuje prepoznatljiva periodičnost (Tilborg, 2005). S tim u vezi, interesantna slova su: B, C, I, ... Iz njihovih peroida može se zaključiti da dužina brojne šifre je 6 ili delilac od 6. Ništa se ne gubi pretpostavkom da je dužina veća, što ne važi obrnuto.

Dakle, našli smo da je moguća dužina translatorne šifre 6. Ostaje da nađemo brojeve od kojih se sastoji. Radi toga, pronađimo sve moguće pozicije gde se dešava promena originalnog teksta sa translacijom sa šifrom dužine 6. Dovoljno je samo prethodni program promeniti u jednoj jedinoj naredbi i to onoj u kojoj se vrši štampanje rezultata. Dakle, treba zameniti red:

```
if (p[m][n]-p[m][0] > 1) printf("%2d ", p[m][n]-p[m][0]);
```

sa sledećim:

```
if (p[m][n] > 0) printf("%2d ", p[m][n] % 6);
```

i u istom ciklusu pustiti da se štampa i nulti (prvi)član.

Posle startovanja program daje sledeći rezultat:

```
A :
B : 5  5  5
C : 3  3  3  3  5
Ć :
Č:
D : 4  4  1  4  4  3  3  4  4
Đ :
E : 3  3  3
F : 2  2  0  0  2  5  0  5
G : 3  4  3  3  3  1  2  3  3  1
H : 0  0  0  0  0  0
I : 1  1  1
J : 5  5  5  5  5  4  3  5  4  4
K : 1  1  3  5  5  3  3  1
L : 2  3  2  3  0  2  0  4  2  0  3  2  5  2
M : 2  5  0  1  0  5  2  0  0  1  1
```

```
N : 5  0  2  4  5  4  0  5  4  1  2  4  4  1
O : 1  4  1  5  5  5  5  0
P : 5  4  3  5  2  4  2  5  2  2  2
Q : 3  3  2  3  1  2  0  2  5  0  3
R : 2  0  2  0  1  2  4  2  2  3  2  1  0  0  0  3  4
S : 2  5  4  4  4  4  1  0  2  1
Š : 3  3  5  3  4  5  4  4  3  4
T : 3  3  4  5  3  3  3  3  3
U : 0  2  0  5  1  1  2  5  2  0  4  1  2  5
V : 2  5  3  1  4  1  5  1  3  1  5  4  4
F : 2  2  0  0  2  5  0  5
Z : 1  4  1
Ž : 4  5  4  4
X : 1  3  1
Y : 1  0  1  4  4  0  1  0
W : 3  2  0  0  2  4
Press any key to continue . . .
```

Ovu tabelu tumačimo tako što vidimo da se slovo B uvek povećava na 5. poziciji, slovo H na 6, itd. Sada je potrebno malo eksperimentisati, odnosno napraviti neku pretpostavku, pa je zatim proveriti. Recimo, C se najčešće povećava na 3. poziciji, a to je najverovatnije od slova A, a B se najčešće javlja na 5. mestu, što je najverovatnije takođe uvećanje od slova A. Slovo D se najčešće povećava na 4. mestu, što omogućava pretpostavku da je na tom mestu broj 5 s obzirom na slovo A.

Sledećim alatom možemo, na osnovu neke hipoteze, izvršiti dekriptovanje i time tačno odrediti i ostale cifre u nepoznatom brojnom translatu.

U program unosimo niz od 6 brojeva, a on dati šifrat prevrće prema tom translatu. Navodimo samo deo koji je u okviru glavnog programa i sa kojim testiramo neku od pretpostavki.

```
short p[6] = {4,3,2,3,1,2};  /* niz koji sadrži pretpostavku o translatu */
for ( k = 0; n < 280; k++)
   {   for (n = 0; n < 32; n++)
       if (s[k] == a[n])
              {   m1 = k % 6;
          m1 = n  - p[m1];
          if (m1 < 0) m1 = m1 + 32;
          printf("%c", a[m1]);
            }
   }
```

Na kombinaciju koju smo naveli program daje sledeći ispis:

PRAVI VINOVNIK KRAĐE DIJAMANATA I UBISTVA VOJNIKA KOJI SU PRATILI SPROVOD IZVRŠENIH U NOĆI DVADESET DRUGOG JANUARA ILJADU OSAMSTOTINA DVADESET ŠESTE GODINE NIJE I DAKLE ŽOAM DA COSTA NEOPRAVDANO OSUĐEN NA SMRT NEGO SAM TO JA BEDNI ČINOVNIK UPRAVE DIJAMANTSKE OBLASTI JEST JA JEDINI KOJI SE POTPISUJEM SVOJIM PRAVIM IMENOM ORTEGA

Drugi primer

Sledeći problem postavljen je davne 1906. godine. U narednih 50 godina bilo je poznato samo jedno rešenje, a kasnije se navodi još 6 nađenih rešenja. Zadatak je da se dešifruje sledeće deljenje tako što se na svakom mestu nalazi neka cifra. Za nule ispred brojeva u međurezultatima posebno se ne naglašava da li su dozvoljene ili ne. Za glavno rešenje pretpostavićemo da sa nulama ne počinje zapis brojeva (Petković, 1988).

```
xx7xxxxxxx : xxxx7x = xx7xx
-xxxxxx
 xxxxx7x
-xxxxxxx
    x7xxxx
   -x7xxxx
    xxxxxxx
   -xxxx7xx
       xxxxxx
      -xxxxxx
            0
```

Analiza problema za drugi primer

Metoda „grube sile“ je najjednostavniji način da se napravi potreban algoritam. Međutim, mora se biti veoma oprezan sa primenom metode „grube sile“. Ako bismo birali nepoznate potrebno je razmotriti sve slučajeve unutar tri gornja broja. Tako, ako bismo uzeli za nepoznate cifre od deljenika i delioca imali bismo 16 punih ciklusa sa po 10 mogućnosti, što je još uvek prevelik zalogaj i za današnje računare. Zadatak se može interpretirati kao problem množenja jednog šestocifrenog broja sa jednim petocifrenim brojem tako da se dobije desetocifreni broj uz dodatna ograničenja koja su vidljiva iz datih međurezultata. Recimo, množenje sa brojem 7 sa deliocem sastoji se od tačno 6 cifara. Kako su prethodni i nared-

ni međurezultat sedmocifreni to su prethodna i naredna cifra u deliocu veće od 7. Osim toga, prva cifra u deliocu može biti samo 1.

U algoritam treba ugraditi funkciju koja će proveravati da li se u međurezultatima cifra 7 nalazi na označenim mestima. Pretpostavićemo da se traže samo rešenja koja svakom nepoznatom mestu dodeljuje cifru, a prvom mestu cifru različitu od nule. Drugačije rečeno, da međurezultati i deljenik ne počinju sa nulom. To uzrokuje da zadatak ima jedinstveno rešenje. Ostala rešenja se dobijaju kada dozvolimo da međurezultati počnu cifrom nula ili da dati iksovi ne predstavljaju brojeve dužine koliko je naznačeno.

Program bi mogao da ima sledeći oblik:

```
#include <stdio.h>
        #include <stdlib.h>
        int dcifra(double broj, int mesto);
        int main( )
        {
          int a, b, c, d, m, n, p, q;
          double x, y, k1, k2, k3, k4;
          for(a = 0; a < 10; a++)
            for (b = 0; b < 10; b++)
              for (c = 0; c < 10; c++)
                for( d = 0; d < 10; d++)
                  for (m = 1; m < 10; m++)
                    for (n = 8; n < 10; n++)
                      for(p = 8; p < 10; p++)
                        for (q = 1; q < 10; q++)
                          {
                          y = 100000 + 10000*a + 1000*b + 100*c + 70 + d;
                            x = y*(10000*m + 1000*n + 700 + 10*p + q);
          /* deljenik mora biti desetocifren sa cifrom 7 kao osmom
od pozadi */
                          if ((x < 1e9) || (x > 1e10) || (dcifra(x, 8) != 7))
break;
          /* medjurezultati pri deljenju */
                          k1 = m*y;
                          k2 = n*y;
                          k3 = 7*y;
                          k4 = p*y;
          /* medjrezultati ne smeju poceti sa nulom */
                        if ((k1 < 1e5) || (k2 < 1e6) || (k3 < 1e5) || (k4 < 1e6))
break;
```

```
          /* provera postojanja ostalih sedmica u medjurezultatima */
                              x = x - k1*10000;
                              if (dcifra(x,5) != 7) break;
                              x = x - k2*1000;
                              if (dcifra(x, 7) != 7) break;
                              if (dcifra(k3, 5) != 7) break;
                              if (dcifra(k4,3) != 7) break;
                              printf(" 1%d%d%d7%d * %d%d7%d%d \n",
a,b,c,d,m,n,p,q);
                           }
          system("PAUSE");
          return 0;
         }
         int dcifra(double broj, int mesto)
         {
          long vrati;
          double medju;
          switch(mesto)
         {
          case 3:
            medju = broj/100;
            break;
          case 5:
            medju = broj/10000;
            break;
          case 7:
             medju = broj/1000000;
              break;
          case 8:
             medju = broj/10000000;
          }
          vrati = (long)medju;
         vrati = vrati - (vrati/10)*10;
          return (int) vrati;
         }
```

Posle startovanja programa dobija se jedinstveno rešenje:

```
125473 * 58781
Press any key to continue . . .
```

Za dobijanje ostalih rešenja potrebno je iz programa isključiti proveru dužine međurezultata i pustiti ograničenja za indekse u punom opsegu. Tada program nalazi 378 različitih rešenja koje zbog ograničenosti prostora ne navodimo. Obuhvaćena su i ona u kojima delilac ima manje od 10 cifara. Jedino je poštovan položaj sedmica gledano zdesna ulevo, što za brojeve i predstavlja stvarnu poziciju.

Zaključak

U radu su prikazani rezultati istraživanja mogućnosti dešifrovanja šifriranog teksta uz korišćenje računara primenom programskog jezika C. Analizirani su primeri dešifrovanja slovnog i brojčanog šifrata, koji mogu biti posebno interensantni za primenu u vojnim, policijskim i diplomatskim službama. Provedena analiza pokazuje da se uz dobru pretpostavku čoveka, i upotrebom računara sa nizom pomoćnih programa kao alata, u razumnom vremenu može doći do željenog cilja (Lomonaco, 1998, Mollin, 2007). Kako nije moguće napraviti univerzalni algoritam koji bi svaki šifrat mogao dešifrovati, to je logičko zaključivanje i odabir alata neizostavno obaveza čoveka. Na kraju, i sam dešifrovani sadržaj ima smisla samo za dalju aktivnost čoveka, dok je računar samo fizički pomogao da se u znatno kraćem vremenu odradi zadati posao. Na dva praktična primera istražene su mogućnosti dešifrovanja i opravdanost konstatacije da tandem čoveka i računara nema premca, kada je u pitanju rešavanje kriptografskih zadataka.

Literatura

Aho, A., Hopcroft, J., Ullman, J., 1978. “The design and analysis of computer algorithms”, Addison-Wesley Publishing company.

Evseev, S. P. (Евсеев, С. П.), Dorohov, O. V. (Дорохов, А. В.), Korolj, O. G. (Король, О. Г.), 2011. “Mechanisms of protection of information in computer networks and systems“ (Механизмы и протоколы защиты информации в компьютерных сетях и системах), Vojnotehnički glasnik/Military Technical Courier, Vol. 59, No. 4, pp.15–39, Ministarstvo odbrane Republike Srbije, Beograd.

Hrg, D., Budin, L., Golub, M., 2004. “Quantum Cryptography and Security of Information Systems“, Proceedings of the 15th International Conference on Information and Intelligent Systems.

Kuljanski, S., 2010. “RSA algoritam i njegova praktična primena“, Vojnotehnički glasnik/Military Technical Courier, Vol. 58, No. 3, pp. 65–77, Ministarstvo odbrane Republike Srbije, Beograd.

Lomonaco, S.J., 1998. “A Quick Glance at Quantum Cryptography“, archive eprint quant.ph/9811056.

Markagić, M., 2012. “Protokoli i pravci razvoja kvantne kriptografije“, Vojnotehnički glasnik/Military Technical Courier, Vol. 60, No. 1, pp.250–265, Ministarstvo odbrane Republike Srbije, Beograd.

Mollin, R.A., 2007. “An Introduction to Cryptography“, 2nd Edn, Chappman & Hall/CRC.

Oppliger, R., 2005. “Contemporary Cryptography“, Artech House

Petković, M., 1988. “Zanimljivi matematički problemi“, Naučna knjiga, Beograd.

Schneier, B., 1996. “Applied Cryptography“, 2nd Edn. John Wiley& Sons.

Scientific American Magazine, 2005. Best-Kept Secrets, p. 65-69, January.

Tilborg, H., 2005. “Encyclopedia of Cryptography Security“, Springer.

USE OF COMPUTERS FOR DECRYPTING MESSAGES

FIELD: Mathematics, Information Technology
ARTICLE TYPE: Professional Paper

Summary:

Cryptography is as current as ever. The need to protect data and resources from unauthorised access and use has never ceased. This resulted in creating stronger protection and breaching most of those thought to be unbreachable. This paper analyses core problems and their implementation in breaching the translational cipher by using the programme language C.

Introduction

When the first computers appeared, a code was deemed to be unbreachable if a computer needed,e.g. a year to find a solution though raw search of all cases. After mere 30 years, computers can finish that job in minutes. What code is unbreachable now? Possibilities and speed of modern computers are such that there are no tasks representing a problem for them. Still, a combination of a man and a computer represents an unmatched team.

Core problems

Message encoding is free of any rules. If one conceives it in that way one cannot make a universal algorithm that could decode any cipher. The use of brute force or searching all possible cases is often impossible. In cases when there is no a password that could be checked, the computer cannot be sure if the job was done. A received decoded text has sense only to the man. It is still impossible to exclude the man from the deciphering process. The man creates tools used by the computer for the physical job that would be impossible for the man to achieve by himself.

The first example

An innocent death-row convict is waiting for the sentence to be executed. In the meantime, his lawyer receives a coded message, and if he succeeds in deciphering it, he might save his client’s life from unjust death sentence.

Analysis of the problem for the first example

Such a task cannot be solved in a general form. Also, one cannot expect a single programme to decipher any text. Such a task is resolved by a string of auxiliary programmes as tools, not a single programme, because one does not know what step to use after another one that was thought to lead to a goal. First, we would code a programme designating the lengths between the appearances of the same letters, because they can serve as a good example for finding the length of the cipher, i.e. a string of numbers the letters are translated with. In nested cycles, one will calculate positions of appearance of all letters in the alphabetical order and print an appropriate report. The given coded text is placed into the stream of symbols.

The second example

This problem was established long ago, in 1906. In the following 50 years, only one solution was found and another six were found later. The task is to decipher the following division, by placing a digit to each position. For zeros in front of numbers in interim results, one does not specify especially whether they are allowed or not. For the final solution, we would assume that a stream of numbers does not start with zeroes.

Analysis of the problem for the second example

The task can be interpreted as a problem of multiplying one six-digit number with a five-digit number so that one receives a ten-digit number with additional limitations visible from the given interim results. The algorithm should include the function that would check whether the number 7 is in designated positions in interim results. We would assume that one seeks only for solutions that assign a digit to an unknown position, and a digit different from zero to the first position. In other words, interim results and the numerator should not start with zero. This causes the task to have a unique solution.

Conclusion

The paper shows the results of researching possibilities to decipher an encoded text by using the computer with the application of the programming language C. The paper shows some examples of deciphering an alphabetical and numerical cipher that could be especially interesting for the application in the military, police or diplomacy. The analysis shows that, with a good assumption by the man, and using computers with a stream of auxiliary programmes as tools, one can reach the goal in reasonable time. Sinceit is not possible to create a universal algorithm that could decode any cipher, the logical conclusions and selection of tools are the prerequisite of the

man. Finally, the very contents of the deciphered text have sense only for further human activities, while computers help only physically to finish the task in a significantly shorter period. The two practical examples explore possibilities of deciphering as well as a justification of the statement that a team made of the man and the computer is unmatched when it comes to resolving cryptographic tasks.

Key words: *cryptography, encoding, translational cipher.*

Datum prijema članka/Paper received on: 26. 04. 2013.
Datum dostavljanja ispravki rukopisa/Manuscript corrections submitted on: 27. 02. 2014.
Datum konačnog prihvatanja članka za objavljivanje/ Paper accepted for publishing on: 28. 02. 2013.