Kriptografija i sigurnost mreža 2023.

2. domaća zadaća

Doris Đivanović

Zadatak 1

Vigenereovom širom iz otovorenog tetksta na hrvatskom jeziku dobiven je šifrat:

RZZRU	VXCAM	ZEAXH	EXGTB	SMYUX	NUVYI	AWMIO	ZQFHW	
OVJHI	IIAYB	ZTUVX	XDWTI	GOYZU	JZFHT	ZOACF	MPEHO	
RXGVA	XZPOS	JGHJG	RHYDE	RFPPD	XKTPY	NHGSJ	BZOUN	
ZZIYI	NTFHN	KCXRN	HGVPZ	HTXED	GSERA	HZNZG	AATZF	KTP

Odredite najprije **duljinu ključne riječi**, potom samu **ključnu riječ**, te **dekriptirajte šifrat**.

Duljina ključne riječi

Prva metoda

Duljinu ključne riječi najprije sam pokušala odrediti pomoću tzv. **Kasiskijevog testa**. Vrlo brzim pregledom šifrata uočila sam da je jedini trigram koji se u šifratu pojavljuje dva puta **UVX**. Četvrto slovo jednog i drugog trigrama ne podudaraju se, pa zaključujem da u šifratu sigurno nema segmenata duljine veće od 3 koji se pojavljuju dva puta. Trigram UVX prvi se put pojavljuje na poziciji 5, a drugi put na poziciji 53, pa razlika njihovih pozicija iznosi $53 - 5 = 48$. Broj 48 možemo faktorizirati na sljedeće načine: $48 = 2^4 \cdot 3 = 16 \cdot 3 = 8 \cdot 6$. Među višekratnicima ove razlike mogu pronaći neke kandidate za duljinu ključne riječi, pa bih mogla pretpostaviti da je duljina ključne riječi $m = 3$ (iako mi je to sumnjivo), ili možda $m = 6$.

Druga metoda

Pribjegavam drugoj metodi koja za svaki kandidat m za duljinu ključne riječi dani šifrat zapisuje stupac po stupac u (krnju) matricu dimenzija $m \times \lceil m/n \rceil$, gdje je n duljina šifrata. Time šifrat dijeli u m podnizova, koji su pojedinačno, ako je m uistinu duljina ključne riječi, šifrirani Cezarovom šifrom s nekim ključem. Također, tada **indeks koincidencije** svakog od podnizova mora biti **jako blizu 0.064** (≈ 0.064 ima tekst pisan na govornom jeziku, a i njegov monoalfabetski šifrat).

Napisala sam sljedeći program koji taj postupak radi za $m = 1, 2, \ldots, 10$. Za svaki m kreira se i na ekran ispisuje matrica. "Višak u matricama" popunila sam znakom *. Zatim se za svaki podniz $z_1, \ldots, z_m$ računa indeks koincidencije $I_c(z_i)$ po formuli i ispisuje se na ekran. Radi lakše provjere koji je m vjerojatnije duljina ključne riječi, za svaki dobiveni indeks računala sam i ispisala njegovo odstupanje od broja 0.064 - u kodu nazvano **GREŠKA**.

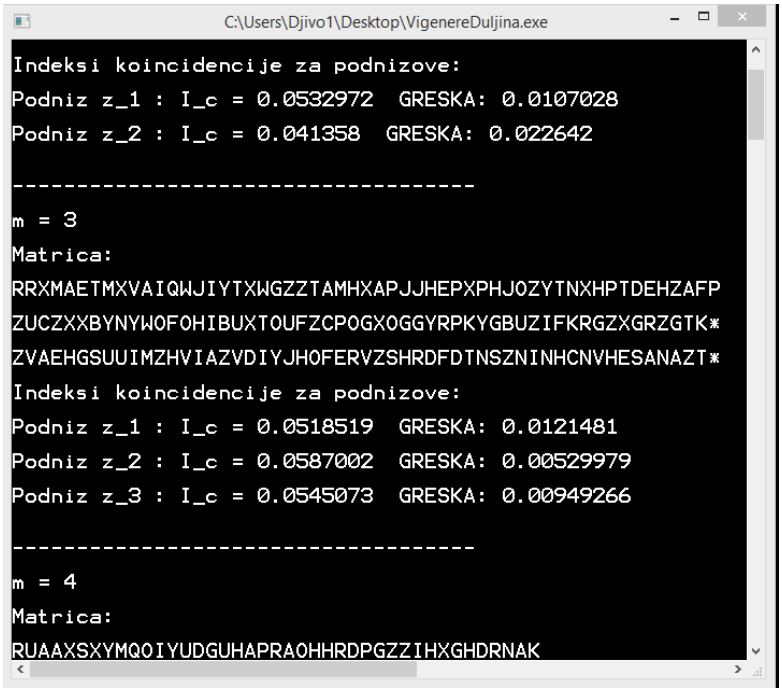
```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>

using namespace std;

```

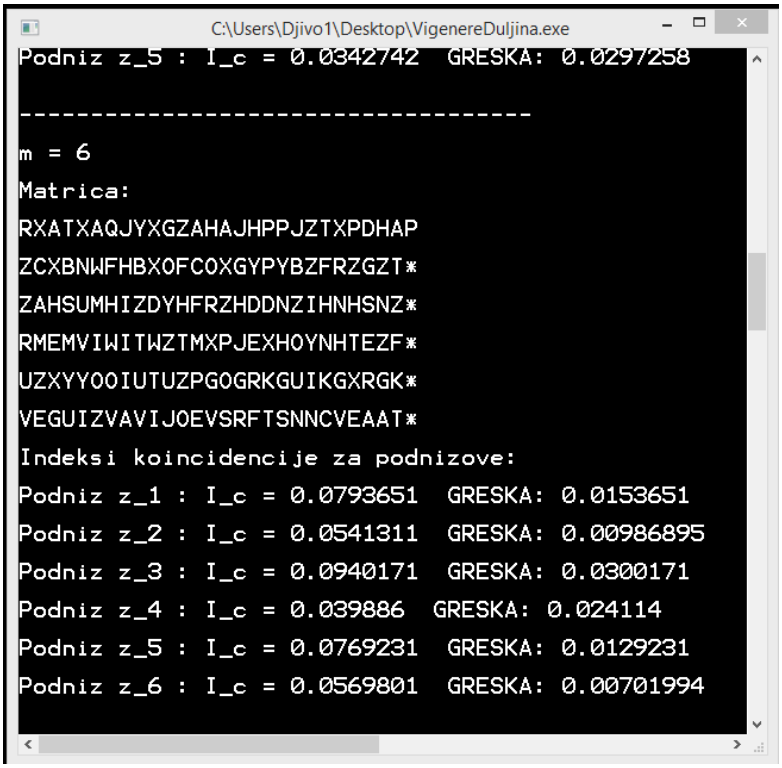
```cpp
int main (void)
{
    string Cyphertext = string("RZZRUVXCAMZEAXHEXGTBSMYUXNUVYIAWMIOZQFHW") +
                        string("OVJHIIIAYBZTUVXXDWTIGOYZUJZFHTZOACFMPEHO") +
                        string("RXGVAXZPOSJGHJGRHYDERFPPDXKTPYNHGSJBZOUN") +
                        string("ZZIYINTFHNKCXRNHGVPZHTXEDGSERAHZNZGAATZFKTP");

    int n = Cyphertext.size();  /// n je duljina zadanog sifrata

    /// slova medunarodne abecede redom 0 - 25
    vector<char> slovo = {'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'};

    cout << "Radit cu metodu napada s matricom za m = 1, 2, ... , 10." << "\n\n";

    for(int m = 1; m <= 10; m++)
    {
        cout << "m = " << m << "\n\n";

        /// Matrica ima m redaka i ceiling(n/m) stupaca
        const int broj_redaka = m;
        const int broj_stupaca = ceil((double)n/(double)m);

        char** matrica = (char**)malloc(broj_redaka*sizeof(char*));
        for(int i = 0; i < broj_redaka; i++)
            matrica[i] = (char*)malloc(broj_stupaca*sizeof(char));

        /// Matricu stupac po stupac punim redom elementima iz sifrata
        for(int j = 0; j < broj_stupaca; j++)
            for(int i = 0; i < broj_redaka; i++)
            {
                if(j*broj_redaka + i < n)
                    matrica[i][j] = Cyphertext[j*broj_redaka + i];
                else
                    /// EVENTUALNI VISAK ELEMENATA MATRICE POSTAVIM NA *
                    matrica[i][j] = '*';
            }

        /// Ispis matrice
        cout << "Matrica: \n\n";
        for(int i = 0; i < broj_redaka; i++)
        {
            for(int j = 0; j < broj_stupaca; j++)
            {
                cout << matrica[i][j];
            }
            cout << endl << endl;
        }

        vector<double> indeks(broj_redaka, 0.0);
        int duljina_podniza, frekvencija;

        /// Racunanje indeksa koincidencije za svaki redak matrice
        for(int redak = 0; redak < broj_redaka; redak++)
        {
            if(matrica[redak][broj_stupaca - 1] != '*')
                duljina_podniza = broj_stupaca;
            else
                duljina_podniza = broj_stupaca - 1;

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

```
                frekvencija = 0;

                for(int l = 0; l < duljina_podniza; l++)
                {
                    if(matrica[redak][l] == slovo[i])
                        frekvencija++;
                }

                indeks[redak] += frekvencija*(frekvencija - 1);
            }

            indeks[redak] = indeks[redak]/(duljina_podniza*(duljina_podniza - 1));
        }

        /// Ispis svih indeksa koincidencije za odredeni broj m
        cout << "Indeksi koincidencije za podnizove: \n\n";
        for(int i = 0; i < indeks.size(); i++)
        {
            cout << "Podniz z_" << i + 1 << " : ";
            cout << "I_c = " << indeks[i];
            cout << "   GRESKA: " << abs(indeks[i]-0.064) << endl << endl;
        }
        cout << endl << "-----------------------------------" << endl << endl;

        for(int i = 0; i < broj_redaka; i++)
            free matrica[i];
        free matrica;
    }

    return 0;
}
```

Druga metoda - analiza rezultata

Gledajući output dobiven na ekranu, čini mi se da su najmanje greške za $m = 3$ i $m = 6$, ali još ne mogu zaključiti koja je sigurno duljina ključne riječi:





Ključna riječ

U svrhu određivanja same ključne riječi, napisala sam još jedan program. On nam omogućava da za fiksiranu, odabranu duljinu ključne riječi m odredimo ključnu riječ. Program najprije fiksira m. Zatim kreira i ispisuje na ekran gore spomenutu matricu samo za tu duljinu. Zatim, postupkom koji smo radili na predavanju, pretpostavljajući da je **otvoreni tekst pisan na hrvatskom jeziku**, računamo zajedničke indekse koincidencije svakog od podnizova $z_1, \ldots, z_m$, tj. svakog retka matrice $j = 1, \ldots, m$, i slučajnog teksta na hrv. jeziku, i time za svaki redak računamo ključ k_j za njegovu Cezarovu šifru. Na kraju, spajanjem tih ključeva u jedan, tj. u jednu riječ, dobivamo i ispisujemo na ekran ključnu riječ duljine m za Vigenereovu šifru. Također, prolazeći po retcima matrice, dešifriramo svaki redak njegovim ključem i dešifrirane elemente spremamo na odgovarajuća mjesta. Dešifriranu matricu ispisujemo kao jedan string prolazeći po stupcima i time dobivamo otvoreni tekst za tu ključnu riječ.

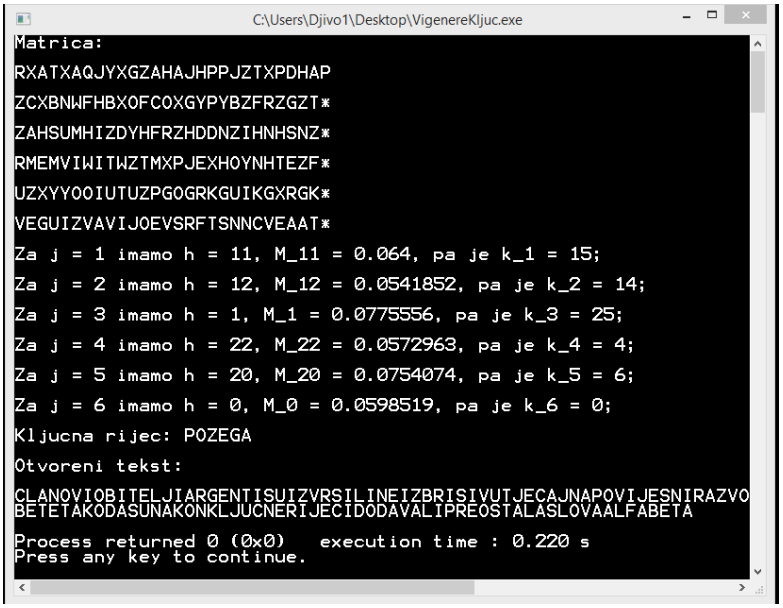
```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
#include <map>

using namespace std;

int main (void)
{
    string Cyphertext = string("RZZRUVXCAMZEAXHEXGTBSMYUXNUVYIAWMIOZQFHW") +
                        string("OVJHIIIAYBZTUVXXDWTIGOYZUJZFHTZOACFMPEHO") +
                        string("RXGVAXZPOSJGHJGRHYDERFPPDXKTPYNHGSJBZOUN") +
                        string("ZZIYINTFHNKCXRNHGVPZHTXEDGSERAHZNZGAATZFKTP");

    int n = Cyphertext.size();  /// n je duljina zadanog sifrata

    /// slova medunarodne abecede redom 0 - 25
    vector<char> slovo = {'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'};

    /// frekvencije slova u HRVATSKOM jeziku redom
    vector<double> p = {0.115, 0.015, 0.028, 0.037, 0.084, 0.003, 0.016, 0.008,
                        0.098, 0.051, 0.036, 0.033, 0.031, 0.066, 0.090, 0.029,
                        0.000, 0.054, 0.056, 0.048, 0.043, 0.035, 0.000, 0.000,
                        0.000, 0.023};

    /// Matrica ima m = 6 redaka i ceiling(n/6) stupaca
    const int m = 6;
    const int broj_stupaca = ceil((double)n/(double)m);

    char** matrica = (char**)malloc(m*sizeof(char*));
    for(int i = 0; i < m; i++)
        matrica[i] = (char*)malloc(broj_stupaca*sizeof(char));

    /// Matricu stupac po stupac punim redom elementima iz sifrata
    for(int j = 0; j < broj_stupaca; j++)
        for(int i = 0; i < m; i++)
        {
            if(j*m + i < n)
                matrica[i][j] = Cyphertext[j*m + i];
            else
                /// EVENTUALNI VISAK ELEMENATA MATRICE POSTAVIM NA *
                matrica[i][j] = '*';
        }

    /// Ispis matrice
    cout << "Matrica: \n\n";
    for(int i = 0; i < m; i++)
    {
        for(int j = 0; j < broj_stupaca; j++)
        {
            cout << matrica[i][j];
        }
        cout << endl << endl;
    }

    vector<int> k (m);  /// ovdje cu redom spremiti slova kljuca (tj. odg. brojeve)

    /// Trazenje k_j za svaki j
    for(int j = 0; j < m; j++)
    {
        int duljina_podniza;
        if(matrica[j][broj_stupaca - 1] != '*')
                duljina_podniza = broj_stupaca;
            else
                duljina_podniza = broj_stupaca - 1;

        vector<int> frekvencija(26, 0);
        for(int i = 0; i < 26; i++)
            for(int l = 0; l < duljina_podniza; l++)
            {
                if(matrica[j][l] == slovo[i])
                        frekvencija[i]++;
            }

        /// Racunanje M_0 - M_25 za fiksirani j
        vector<double> M (26, 0.0);
        for(int g = 0; g < 26; g++)
        {
            for(int i = 0; i < 26; i++)
            {
```

```
                int mod = (i - g) % 26;
                if(mod < 0) mod = 26 + mod;

                M[g] += p[i]*frekvencija[mod];
            }

            M[g] = M[g]/duljina_podniza;
        }

        /// Trazenje maksimalnog M za fiksirani j
        double max = M[0];
        int h = 0;
        for(int g = 1; g < 26; g++)
        {
            if(M[g] > max)
            {
                max = M[g];
                h = g;
            }
        }

        if(h == 0)  k[j] = 0;
        else  k[j] = 26 - h;   /// racunanje k_j

        cout << "Za j = " << j + 1 << " imamo h = " << h;
        cout << ", M_" << h << " = " << max;
        cout << ", pa je k_" << j + 1 << " = " << k[j] << "; \n\n";
    }

    cout << "Kljucna rijec: ";
    for(int j = 0; j < m; j++)
        cout << slovo[k[j]];
    cout << endl << endl;

    /// Korespondencija brojeva i slova - drugi smjer
    map<char, int> mapping = {{'A', 0}, {'B', 1}, {'C', 2}, {'D', 3}, {'E', 4},
                              {'F', 5}, {'G', 6}, {'H', 7}, {'I', 8}, {'J', 9},
                              {'K', 10}, {'L', 11}, {'M', 12}, {'N', 13},
                              {'O', 14},{'P', 15}, {'Q', 16}, {'R', 17},
                              {'S', 18}, {'T', 19}, {'U', 20}, {'V', 21},
                              {'W', 22}, {'X', 23}, {'Y', 24}, {'Z', 25}};
    /// Dekriptiranje
    for(int j = 0; j < m; j++)
        for(int i = 0; i < broj_stupaca; i++)
        {
            if(matrica[j][i] != '*')
            {
                /// od svakog broja u j-tom retku oduzimam k_j mod 26
                int mod = (mapping[matrica[j][i]] - k[j]) % 26;
                if(mod < 0) mod = 26 + mod;

                /// u istu matricu pohranjujem odgovarajuca dekriptirana slova
                matrica[j][i] = slovo[mod];
            }
        }

    /// Ispisujem dekriptiranu matricu kao jedan string, stupac po stupac
    cout << "Otvoreni tekst: \n\n";
    for(int j = 0; j < broj_stupaca; j++)
        for(int i = 0; i < m; i++)
        {
            if(j*m + i >= n) break; /// dosli smo do kraja teksta

```

```
147             cout << matrica[i][j];
148         }
149     cout << endl;
150 
151     for(int i = 0; i < m; i++)
152         free matrica[i];
153     free matrica;
154 
155     return 0;
156 }
```

Uočimo da nam ovaj program može poslužiti kao konačan alat za određivanje duljine ključne riječi. Jednostavno, mijenjanjem parametra m, možemo vidjeti za koji će dobivena ključna riječ biti smislena (iako to ne mora biti), ali možemo vidjeti i za koji će m otvoreni tekst biti smislena poruka, što nam je sigurna potvrda. Uvrštavanjem $m = 3$ u gornji program, dobiva se da je ključna riječ = PGZ, a otvoreni tekst uopće nije smislen. Međutim, uvrštavanjem $m = 6$, dobiva se da je ključna riječ = POŽEGA i dobiva se smisleni otvoreni tekst:



Dakle, zaključujemo da je:

- **duljina ključne riječi = 6**,
- **ključna riječ = POŽEGA**,
- **otvoreni tekst** glasi:

CLANOVI OBITELJI ARGENTI SU IZVRSILI NEIZBRISIV UTJECAJ NA POVIJESNI RAZVOJ KRIPTOLOGIJE. SASTAVLJALI SU SIFARSKE ALFABETE TAKO DA SU NAKON KLJUCNE RIJECI DODAVALI PREOSTALA SLOVA ALFABETA.

Zadatak 2

Šifrirajte otvoreni tekst

PARKER HITT

pomoću **Playfairove šifre** s ključnom riječi CRYPTOGRAPHY.

Rješenje

Matrica slova dimenzije 5×5 uzimajući u obzir ključnu riječ:

C	R	Y	P	T
O	G	A	H	B
D	E	F	IJ	K
L	M	N	Q	S
U	V	W	X	Z

Rastav otvorenog teksta na bigrame:

PA RK ER HI TT

Bigram od istih slova rastavlja se s X, broj slova do parnog nadopunjuje se s X:

PA RK ER HI TX TX

Šifriranje:

- P i A nasuprotni vrhovi pravokutnika $\Rightarrow$ PA $\rightarrow$ YH,
- R i K nasuprotni vrhovi pravokutnika $\Rightarrow$ RK $\rightarrow$ TE,
- E i R u istom stupcu $\Rightarrow$ ER $\rightarrow$ MG,
- H i I u istom stupcu $\Rightarrow$ HI $\rightarrow$ IQ,
- T i X nasuprotni vrhovi pravokutnika $\Rightarrow$ TX $\rightarrow$ PZ.

Šifrat:

YHTEMG IQPZPZ

Zadatak 3

Odredite ključ K u **Hillovoj šifri** ako je poznato da je $m = 2$, te da otvorenom tekstu

HEBERN

odgovara šifrat

RVBHCF.

Rješenje

Budući da je $m = 2$, Hillovom se šifrom dva uzastopna slova u otvorenom tekstu zamjenjuju s dva uzastopna slova u šifratu. Stoga, kako znamo da se otvoreni tekst HEBERN šifrira u RVBHCF, uz tablicu korespondencije slova međunarodne abecede i skupa ostataka modulo 26, možemo zaključiti da za traženi ključ K vrijedi:

$$\begin{aligned} e_K(7,4) &= (17,21), \\ e_K(1,4) &= (1,7), \\ e_K(17,13) &= (2,5). \end{aligned}$$

Dakle, imamo tri para uređenih parova

$$(x_{1j}, x_{2j}) \quad \text{i} \quad (y_{1j}, y_{2j}) \quad \text{t. d.} \quad e_K(x_{1j}, x_{2j}) = (y_{1j}, y_{2j}).$$

Budući da je $m = 2$, za analizu su nam dovoljna dva uređena para, npr. $(7,4)$ i $(17,13)$. Iz gornjih dviju jednadžbi te definicije Hillovog kriptosustava dobivamo sljedeću matričnu jednadžbu (**sve operacije su mod** 26):

$$\begin{vmatrix} 7 & 4 \\ 17 & 13 \end{vmatrix} K = \begin{vmatrix} 17 & 21 \\ 2 & 5 \end{vmatrix}.$$

Ako je matrica X invertibilna, ključ K možemo dobiti na sljedeći način:

$$K = X^{-1}Y.$$

Znamo da je matrica A u $\mathbb{Z}_{26}$ invertibilna akko joj je determinanta invertibilna u $\mathbb{Z}_{26}$, tj. ako je $(\det A, 26) = 1$, i tada je

$$A^{-1} = (\det A)^{-1} \begin{vmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{vmatrix}.$$

Računamo

$$\det X = \det \begin{vmatrix} 7 & 4 \\ 17 & 13 \end{vmatrix} = 7 \cdot 13 - 17 \cdot 4 = 91 - 68 = 23 \bmod 26 = 23.$$

Sada iz tablice inverza u $\mathbb{Z}_{26}$

1	3	5	7	9	11	15	17	19	21	23	25
1	9	21	15	3	19	7	23	11	5	17	25

vidimo da je

$$(\det X)^{-1} = 23^{-1} = 17,$$

pa je X invertibilna matrica i vrijedi

$$X^{-1} = 17 \begin{vmatrix} 13 & -4 \\ -17 & 7 \end{vmatrix} = \begin{vmatrix} 13 & 10 \\ 23 & 15 \end{vmatrix}$$

Konačno, **možemo izračunati ključ** K:

$$K = \begin{vmatrix} 13 & 10 \\ 23 & 15 \end{vmatrix} \begin{vmatrix} 17 & 21 \\ 2 & 5 \end{vmatrix} = \begin{vmatrix} 7 & 11 \\ 5 & 12 \end{vmatrix}.$$

Za kraj ćemo provjeriti odgovara li dobiveni ključ K informaciji

$$e_K(1,4) = (1,7).$$

Dobivamo:

$$e_K(1,4) = (1,4)K = (1,4) \begin{vmatrix} 7 & 11 \\ 5 & 12 \end{vmatrix} = (1,7),$$

dakle u redu je.