**Лабораторна робота №4**

Розробка програм з використанням шифру Хілла

Мета: отримати представлення про шифр Хілла та розробити програмне забезпечення, з використанням шифру Хілла.

Завдання:

Написати програму, що шифру повідомленням, використовуючи шифр Хілла. Розмір блоку для шифрування – 5 символи.

Хід роботи

1. Текст програми

Hill.ts

import chunk from 'lodash/chunk';

import { modInvMat, matMulVec } from './math';

export function encode(msg: string, \_key: string): string {

   const key = prepareKey(\_key);

   return process(padMsg(msg, key[0].length), key);

}

export function decode(msg: string, key: string): string {

   return process(msg, modInvMat(prepareKey(key), 2 \*\* 16));

}

function process(msg: string, key: number[][]): string {

   const codes = chunk(prepareStr(msg), key[0].length)

      .map(m => matMulVec(key, m)).flatMap(r => r);

   return String.fromCharCode(...codes);

}

function prepareKey(key: string) {

   const size = Math.sqrt(key.length) ^ 0;

   return chunk(key.slice(0, size \* size).split('')

      .map(ch => ch.codePointAt(0)), size);

}

function prepareStr(str: string): number[] {

   return str.split('').map(ch => ch.codePointAt(0));

}

function padMsg(msg: string, size: number): string {

   return msg.padEnd(msg.length + size - msg.length % size, ' ');

}

math.ts

import { det as \_det } from 'mathjs';

export function modInvMat(m: number[][], mod: number): number[][] {

   const d = Math.round(det(m));

   const invDet = modInv(modBy(d, mod), mod);

   return adjoint(m).map(r => r.map(a => {

      return Math.round(modBy(a \* invDet, mod));

   }));

}

export function det(m: number[][]): number {

   if (m.length === 2) {

      return m[0][0] \* m[1][1] - m[0][1] \* m[1][0];

   }

   return m.map((n, i) => n[0] \* ((-1) \*\* i) \* minor(m, i, 0))

      .reduce((sum, n) => sum + n, 0);

}

export function matMulVec(m: number[][], v: number[]): number[] {

   if (m.length && m[0].length !== v.length) {

      throw 'The matrix and the vector are inconsistent';

   }

   return m.map((r) => {

      return v.reduce((sum, b, j) => sum + r[j] \* b, 0);

   });

}

export function adjoint(m: number[][]): number[][] {

   return transpose(m.map((r, i) => r.map((\_, j) => confractor(m, i, j))));

}

export function confractor(m: number[][], i: number, j: number): number {

   return ((-1) \*\* (i + j)) \* minor(m, i, j);

}

export function minor(m: number[][], i: number, j: number): number {

   const filtered = m.filter((\_, k) => k !== i)

      .map(r => r.filter((\_, k) => k !== j));

   return det(filtered);

}

export function transpose(matrix: number[][]): number[][] {

   return matrix.map((r, i) => r.map((v, j) => matrix[j][i]));

}

export function modInv(a: number, m: number) {

   a = a % m;

   for (let x = 1; x < m; x++) {

      if ((a \* x) % m == 1) return x;

   }

   console.log(a, m);

   throw "There is no modular multiplicative inverse for the integer";

}

export function modBy(v: number, m: number) {

   return (v < 0) ? m - (-v % m) : v % m;

}

index.ts

const INPUT\_DELAY = 200;

run();

updateCharKeyNumKey();

initEvents();

function initEvents() {

   $key.addEventListener('input', debounce(INPUT\_DELAY, () => {

      run();

      updateCharKeyNumKey();

   }));

   $msg.addEventListener('input', debounce(INPUT\_DELAY, run));

   $encodeMsg.addEventListener('input', debounce(INPUT\_DELAY, run));

   $decodeMsg.addEventListener('input', debounce(INPUT\_DELAY, run));

}

function run() {

   const key = $key.value.trim();

   const msg = $msg.value.trim();

   const encodedMsg = encode(msg, key);

   const decodedMsg = decode(encodedMsg, key);

   $encodeMsg.value = encodedMsg;

   $decodeMsg.value = decodedMsg;

}

function updateCharKeyNumKey() {

   const key = $key.value.trim();

   const size = Math.sqrt(key.length) ^ 0;

   const charKey = chunk(key.slice(0, size \* size).split(''), size);

   const numKey = chunk(key.slice(0, size \* size).split('')

      .map(ch => ch.codePointAt(0)), size);

   $charKey.value = keyToString(charKey);

   $numKey.value = keyToString(numKey);

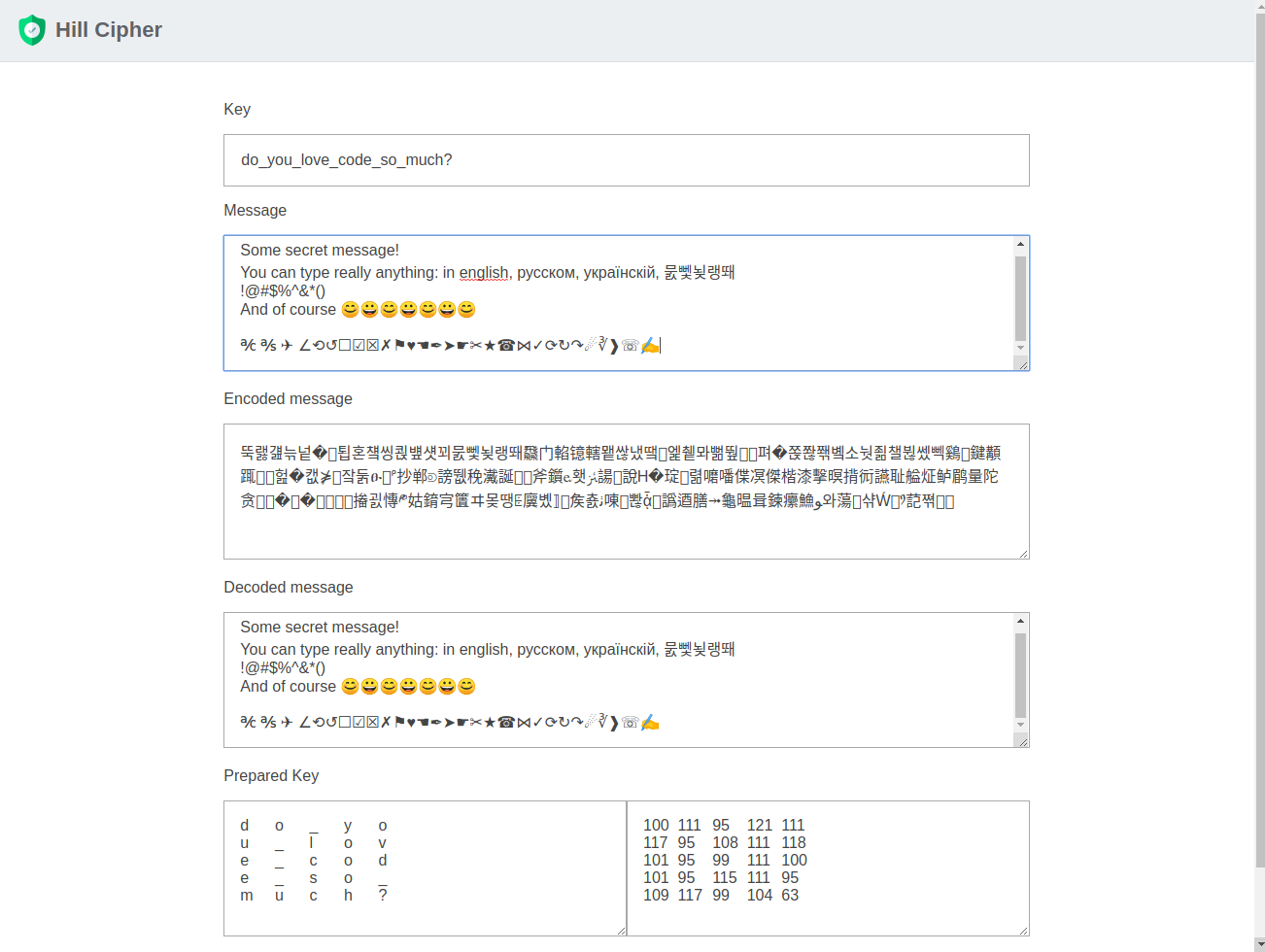
}

function keyToString(key: any[][]) {

   return key.map(r => r.join('\t')).join('\n');

}

1. Результат виконання програми



**Висновок:** на цій лабораторній роботі я отримав представлення про типові схеми ідентифікації та аутентифікації, набув практичних навичок створення програм для ідентифікації та аутентифікації з використанням Playfair Cipher.