МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА — Российский технологический университет»

РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

(наименование института, филиала)

Кафедра КБ-14 «Цифровые технологии обработки данных»

(наименование кафедры)

Практическая работа

по дисциплине Криптографическая защита информации

(наименование дисциплины)

Выполнил:

БСБО-05-20

Верхотуров В. С.

Москва — 2024 г.

СОДЕРЖАНИЕ

1 ЗАДАНИЕ 1. ШИФР БИГРАММАМИ

1.1 Задание

Наиболее известный шифр биграммами называется Playfair (использовался в I Мировой войне). Открытый текст разбивается на пары (биграммы). Текст шифруется по следующим правилам:

— Если обе буквы биграммы исходного текста принадлежат одному столбцу таблицы, то буквами шифра считаются буквы, которые лежат под ними. Если буква открытого текста находится в нижнем ряду, то для шифра берется соответствующая буква из верхнего ряда. Биграмма из одной буквы или пары одинаковых букв тоже подчиняются этому правилу.

— Если обе буквы биграммы исходного текста принадлежат одной строке таблицы, то буквами шифра считаются буквы, которые лежат справа от них. Если буква открытого текста находится в правой колонке (в последней), то для шифра берется соответствующая буква из первой колонки.

— Если обе буквы биграммы открытого текста лежат в разных столбцах, то вместо них берутся такие 2 буквы, чтобы вся четверка их представляла прямоугольник. При этом последовательность букв в шифре должна быть зеркальной исходной паре.

Пример для таблицы 5×6 с ключом «Республика»

Открытый текст ПУ СТ ЬК ОН СУ ЛЫ БУ ДУ ТБ ДИ ТЕ ЛЬ НЫ

Шифр УБ РХ СЗ ДО ПБ ИЩ РБ НР ШР ЖЛ ФР КЩ ЗЮ

1.2 Выполнение практической

Результат практической: `https://crypto-tasks.vercel.app/task1`.

Репозиторий `https://github.com/ValeryVerkhoturov/crypto-tasks`.

Листинг 1 – Playfair шифр

```
1 export class PlayfairCipher {
2         private readonly key;
3         private gridSize: number = 6;
4         private grid: string[][] = [];
5         private positionMap: Map<string, {row: number, col: number}> = new Map
              ();
6
```

```
7         constructor(key: string) {
8                 this.key = key;
9                 this.initializeGrid();
10        }
11
12        private initializeGrid(): void {
13                const alphabet = АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"";
14                let keyString = Array.from(new Set(this.key.replace(/\s+/g,
                      '').toUpperCase() + alphabet))
15                .filter(char => alphabet.includes(char)).join('');
16
17                for (let i = 0; i < this.gridSize; i++) {
18                        this.grid[i] = [];
19                        for (let j = 0; j < this.gridSize; j++) {
20                                const char = keyString[(i * this.gridSize) + j
                                      ];
21                                this.grid[i][j] = char;
22                                this.positionMap.set(char, { row: i, col: j })
                                      ;
23                        }
24                }
25        }
26
27        encrypt(plaintext: string): string {
28                return this.processText(plaintext, 'encrypt');
29        }
30
31        decrypt(ciphertext: string): string {
32                return this.processText(ciphertext, 'decrypt');
33        }
34
35        private processText(inputText: string, mode: 'encrypt' | 'decrypt'):
              string {
36                let processedText = inputText.toUpperCase().replace
                      АЯЁ(/[^-]+/g, "");
37                let output = "";
38
39                for (let i = 0; i < processedText.length; i += 2) {
40                        if (i + 1 >= processedText.length) {
41                                processedText += Ь"";
42                        }
43
44                        if (processedText[i] === processedText[i + 1]) {
45                                processedText = processedText.substring(0, i +
                                      1) + Ь"" + processedText.substring(i + 1)
                                      ;
```

```
46                          }
47
48                          const pos1 = this.positionMap.get(processedText[i]);
49                          const pos2 = this.positionMap.get(processedText[i +
                                1]);
50
51                          if (pos1 && pos2) {
52                                  let row1 = pos1.row;
53                                  let col1 = pos1.col;
54                                  let row2 = pos2.row;
55                                  let col2 = pos2.col;
56
57                                  if (row1 === row2) {
58                                          col1 = this.mod((col1 + (mode === '
                                                encrypt' ? 1 : -1)), this.gridSize
                                                );
59                                          col2 = this.mod((col2 + (mode === '
                                                encrypt' ? 1 : -1)), this.gridSize
                                                );
60                                  } else if (col1 === col2) {
61                                          row1 = this.mod((row1 + (mode === '
                                                encrypt' ? 1 : -1)), this.gridSize
                                                );
62                                          row2 = this.mod((row2 + (mode === '
                                                encrypt' ? 1 : -1)), this.gridSize
                                                );
63                                  } else {
64                                          [col1, col2] = [col2, col1];
65                                  }
66
67                                  output += this.grid[row1][col1] + this.grid[
                                        row2][col2];
68                          }
69                  }
70
71                  return output;
72          }
73
74          private mod(n: number, m: number): number {
75                  return ((n % m) + m) % m;
76          }
77  }
```

ЗАКЛЮЧЕНИЕ