Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Звіт

До практичної роботи №3

З дисципліни: «Безпека програм та даних»

Виконав:

ст. гр. ПЗПІ-19-3

Селевич О.В.

Перевірив:

асистент кафедри ПІ

Олійник О.О.

Харків 2022

Тема роботи: Створення програмної реалізації простих алгоритмів шифрування.

Мета роботи: Опрацювати навички з використання простих алгоритмів шифрування для застосування в практичних цілях, створення програмної реалізації одного з простих алгоритмів шифрування і знайомство з методами криптоаналізу або створенням захищеного каналу передачі даних.

Хід роботи:

Для виконання цього практичного заняття я обрав завдання номер 3, що має наступний зміст:

- Реалізувати шифр цезаря програмно;
- Реалізувати атаку на зашифрований текст.

Під час практичного заняття була створена програмна реалізація шифру Цезаря, а також програмна реалізаця алгоритму для атаки на шифр Цезаря за допомогою мови програмування JavaScript. Користувацький інтерфейс для демонстрації роботи алгоритмів був створений за допогою JavaScript-фреймворку Vue.js.

1) Програмна реалізація шифру Цезаря

Для виконання кодування та декодування за допомогою шифру Цезаря були створені наступні функції encrypt та decrypt, що знаходяться у файлі caesarsCipher.js:

```
1. export const decrypt = (shift, cipher, language) => {
2.   const languageArray = getLanguageArray(language);
3.   let encryptionArray = getLanguageArray(language);
4.
5.   if (shift < 0) shift = languageArray.length + shift;
6.
7.   shiftArray(shift, encryptionArray);
8.
```

```
9.   return mapText(cipher, encryptionArray, languageArray, false);
10.  };
11.
12.  export const encrypt = (shift, message, language) => {
13.    const languageArray = getLanguageArray(language);
14.    let encryptionArray = getLanguageArray(language);
15.
16.    if (shift < 0) shift = languageArray.length + shift;
17.
18.    shiftArray(shift, encryptionArray);
19.
20.      return  mapText(message,  languageArray,  encryptionArray,
  false);
21.  };
```

Ці функції приймають по три параметри: вихідне повідомлення/шифр, мова тексту та ключ. Функція encrypt відповідає за кодування, а decrypt відповідно для декодування. Спочатку виконується виклик фунції getLanguageArray, що повертає масив з алфавітом. Далі виконується перевірка на наявність ключа з негативним зсувом. Функція shiftArray відповідає за виконання зсуву масиву для кодування/декодування. Код функції shiftArray:

```
1. export const shiftArray = (shift, encryptionArray) => {
2.   const n = encryptionArray.length;
3.
4.   const shiftedArray = [...Array(n)];
5.   for (let i = 0; i < n; i++) {
6.     shiftedArray[i] = encryptionArray[(i + shift) % n];
7.   }
8.   encryptionArray.splice(0, n, ...shiftedArray);
9. };
```

Далі виконується виклик функції mapText що повертає результуючий шифр/вихідне повідомлення та зіставляє символи з двох масивів (масиву алфавіту та масиву кодування/декодування) для виконання моноалфавітної заміни. Код функції mapText:

```
1. export const mapText = (
2.   text,
3.   textLanguageArray,
```

```
4.   textEncryptionArray,
5.   makeLowerCase
6. ) => {
7.   let result = [];
8.   let textCharacters = [...text.split("")];
9.
10.    textCharacters.forEach((elem) => {
11.      const lowerCaseElem = elem.toLowerCase();
12.      const index = textLanguageArray.indexOf(lowerCaseElem);
13.
14.      if (index === -1) result.push(elem);
15.      else {
16.        let resultCharacter = "";
17.        if (lowerCaseElem === elem || (lowerCaseElem !== elem &&
18.  makeLowerCase)) {
19.          resultCharacter = textEncryptionArray[index];
20.        } else {
21.                                              resultCharacter     =
   textEncryptionArray[index].toUpperCase();
22.        }
23.        result.push(resultCharacter);
24.      }
25.    });
26.
27.    return result.join("");
28.  };
```

2) Програмна реалізація алгоритму, котрий виконує атаку на шифр цезаря

Функція, що виконує атаку на текст, зашифрований за допомогою шифру Цезаря знаходиться у файлі caesarsCipher.js та має назву breakCaesarsCipher. Програмний код функції breakCaesarsCipher:

```
1. export const breakCaesarCipher = (cipher, language,
2. isBestShiftReturned) => {
3.   const languageArray = getLanguageArray(language);
4.   let encryptionArray = getLanguageArray(language);
5.   const languageFrequencyMap = getTableFromJson(language);
6.
7.   const maxShift = languageArray.length;
8.
9.   let tmpDistributionValue = Number.MAX_VALUE;
```

```
    let bestShift = 0;

    for (let i = 0; i < maxShift; i++) {
      encryptionArray = getLanguageArray(language);
      shiftArray(i, encryptionArray);
      const shiftedText = mapText(cipher, languageArray,
  encryptionArray, true);
      const cipherFrequencyMap =
  initializeFrequencyMap(shiftedText, language);
      const squaredXDistribution = countChiSquaredStatistic(
        cipherFrequencyMap,
        languageFrequencyMap,
        cipher.length
      );
      if (squaredXDistribution < tmpDistributionValue) {
        bestShift = i;
        tmpDistributionValue = squaredXDistribution;
      }
    }
    if (isBestShiftReturned) return bestShift;

    encryptionArray = getLanguageArray(language);

    shiftArray(bestShift, encryptionArray);
    return mapText(cipher, languageArray, encryptionArray,

  false);
  };
```

На вхід функція breakCaesarsCipher приймає три параметри: зашифроване повідомлення, мову повідомлення та булевий параметр, що визначає чи треба повернтути ключ або взламане повідомлення. Спочатку використовується виклик функції getLanguageArray, котра повертає алфавіт мови що була передана як вхідний парамтер у функцію. Код функції getLanguageArray:

```
export const getLanguageArray = (language) => {
  switch (language) {
    case "Ru":
      const arr = generateAlphabeticArray("а", "я");
      arr.splice(6, 0, "ё");
      return arr;
    case "En":
      return generateAlphabeticArray("a", "z");
  }
  };
```

Функція breakCaesarsCipher взламує шифр за допомогою частотного аналізу тому у додатку існують два json файлы, котрі я частотними таблицями появ відвовідних літер у тексті для двох мов: російської та англійської, з назвами russianLettersFrequencies.json та englishLettersFrequencies.json. У функції breakCaesarsCipher частотна таблиця отримується з результату виклику функції getTableFromJson. Код функції getTableFromJson:

```
1. export const getTableFromJson = (language) => {
2.   const frequencyMap = new Map();
3.   let json = "";
4.
5.   console.log(language);
6.
7.   switch (language) {
8.     case "Ru":
9.       json = russianLettersData;
10.        break;
11.      case "En":
12.        json = englishLetterData;
13.        break;
14.    }
15.
16.    for (let letter in json) {
17.      frequencyMap.set(letter, json[letter]);
18.    }
19.
20.    return frequencyMap;
21.  };
```

Для виконання атаки у функції breakCaesarsCipher використовуються Chi-Squared-Statistic (статистика x^2). Вона є мірою того, наскільки два категоріальні розподіли відрізняються один від одного. Таким чином, для 2 однакових розподілів оцінка буде дорівнювати 0, а коли розподіли починають відрізнятися, оцінка буде збільшуватися. Формулу для знаходження Chi-Squared-Statistic можна побачити на рисунку 1.

$$X^2 = \sum_{i=A}^{i=Z} \frac{(O_i - E_i)^2}{E_i}$$

Рисунок 1 – Формула Chi-Squared-Statistic

У цій формулі Oi – спостережувана кількість літер у вашому тексті, а Ei – очікувана кількість цієї літери у довжині вашого тексту.

Під час атаки на шифр Цезаря у створеній функції перебираються усі можливі ключі, що могли бути використані для зашифрування. Наприклад для англійського алфавіту ключів може бути 26, бо він містить 26 літер і відповідно 26 значень зсуву. А потім виконується зсув шифру згідно з поточним значенням ключа і розраховується значення Chi-Squared-Statistic для з'ясування чи походить отримане повідомлення на звичайний текст написаний на мові, що була передана як аргумент у функцію. Тобто у результаті чим менше буде отримане значення у результаті розрахунку Chi-Squared-Statistic для поточного ключа тим більша вірогідність, що це і буде ключ, використаний для кодування вихідного повідомлення.

Функція countChiSquaredStatistic розраховує значення Chi-Squared-Statistic. Програмний код функції:

```
1. const countChiSquaredStatistic = (
2.   observedFrequencyMap,
3.   expectedFrequencyMap,
4.   cipherTextLength
5. ) => {
6.   let sum = 0;
7.
8.   observedFrequencyMap.forEach((value, key) => {
9.     const expectedCount =
10.         (expectedFrequencyMap.get(key) / 100) * cipherTextLength;
11.      const tmp = Math.pow(value - expectedCount, 2) /
12.  expectedCount;
13.      sum += tmp;
14.    });
15.
16.    return sum;
17.  };
```

У результаті функція breakCaesarsCipher повертає вихідне повідомлення.

3) Демонстрація роботи створених алгоритмів

З метою демонстрації роботи програмної реалізації створених алгоритмів була створена UI сторінка , котру можна побачити на рисунку 2. На цій сторінці знаходяться два елементи, що згортаються та розгортаються з відповідними назвами при натисканні на які відкриваються форми для тестування програмної реалізації алгоритмів. Також на цій сторінці є кнопки для перегляду завдання до практичного заняття та створеного звіту.

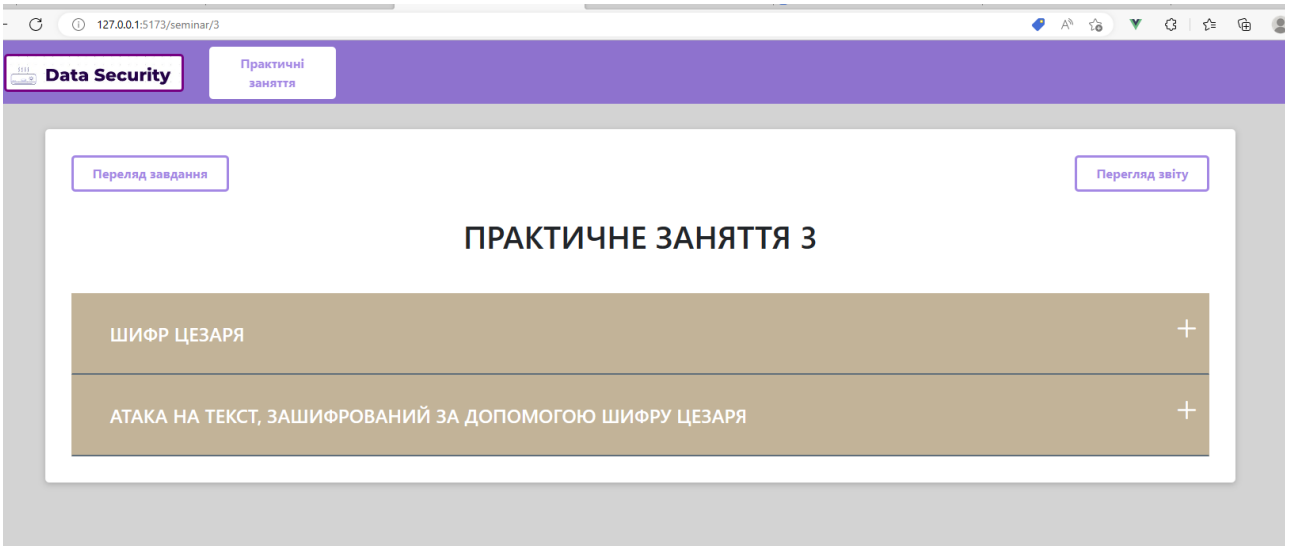

Рисунок 2 – Створена сторінка

Для тестування роботи алгоритму, що виконує кодування/декодування за допомогою шифру Цезаря були використані тестові приклади з методичних вказівок для першого практичного заняття. Приклади роботи створеного застосунку з кодуванням/декодуванням за допомогою шифру Цезаря можливо побачити на рисунках 3, 4.

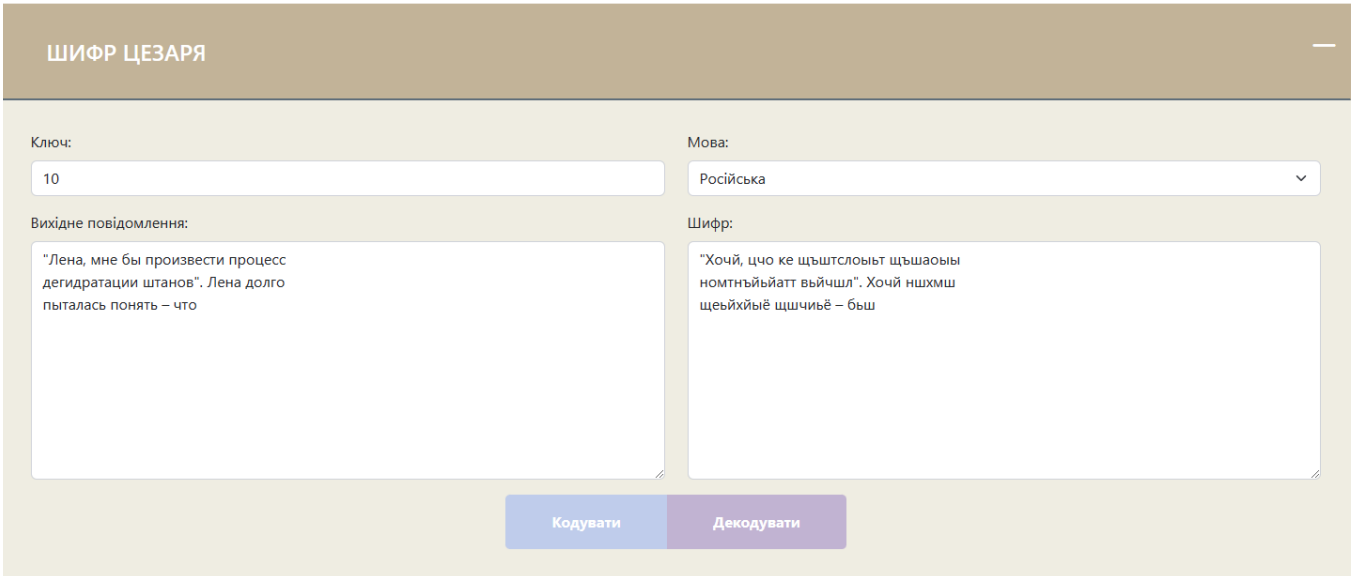

Рисунок 3 – Приклад кодування за допомогою шифру Цезаря

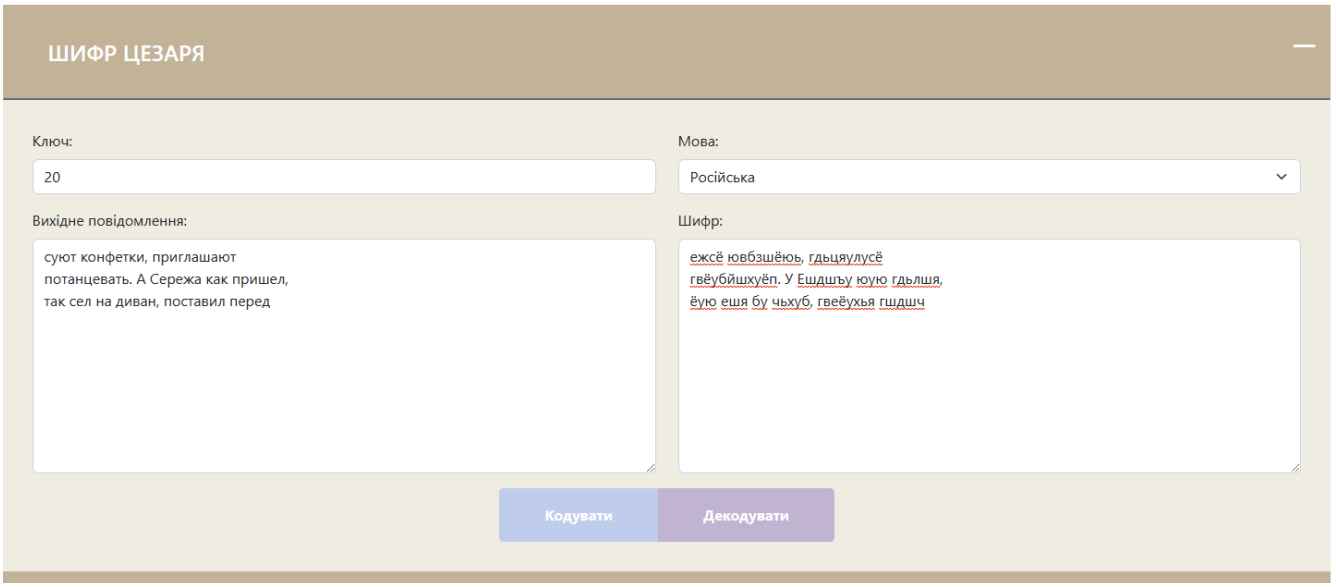

Рисунок 4 – Приклад декодування за допомогоюю шифру Цезаря

Приклади роботи створеного застосунку для здійснення атаки на повідомлення, зашифроване за допомогою шифру цезаря можливо побачити на рисунках 5, 6.

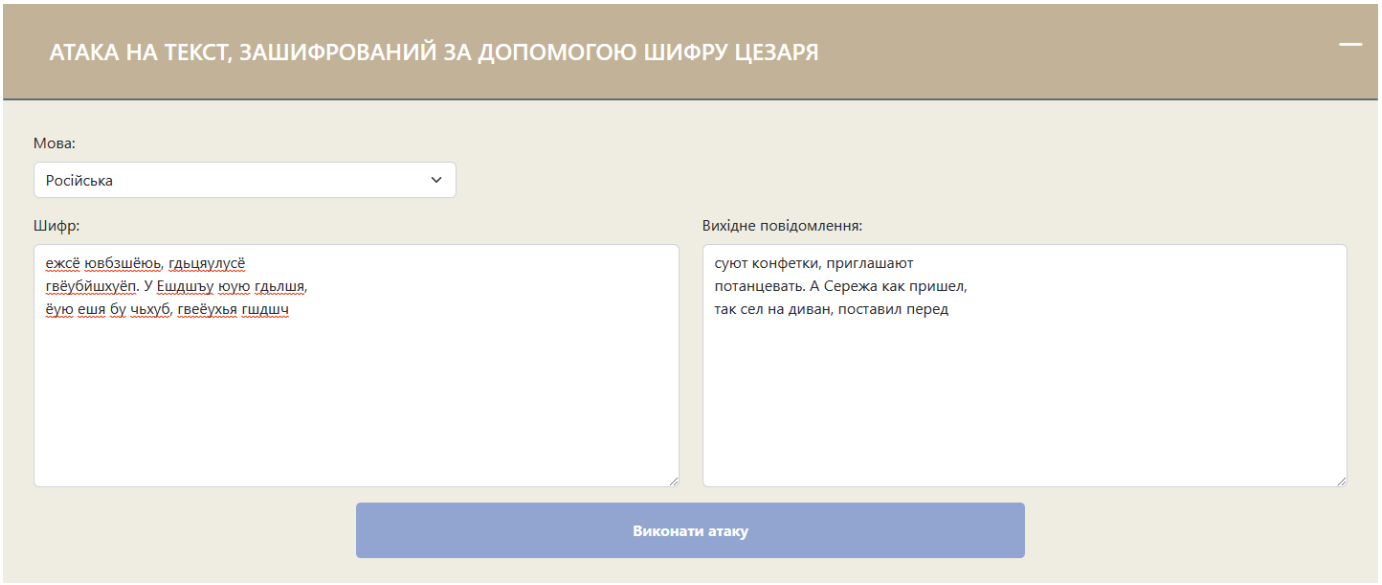

Рисунок 5 – Приклад виконання атаки на текст, зашифрований за допомогою шифру Цезаря

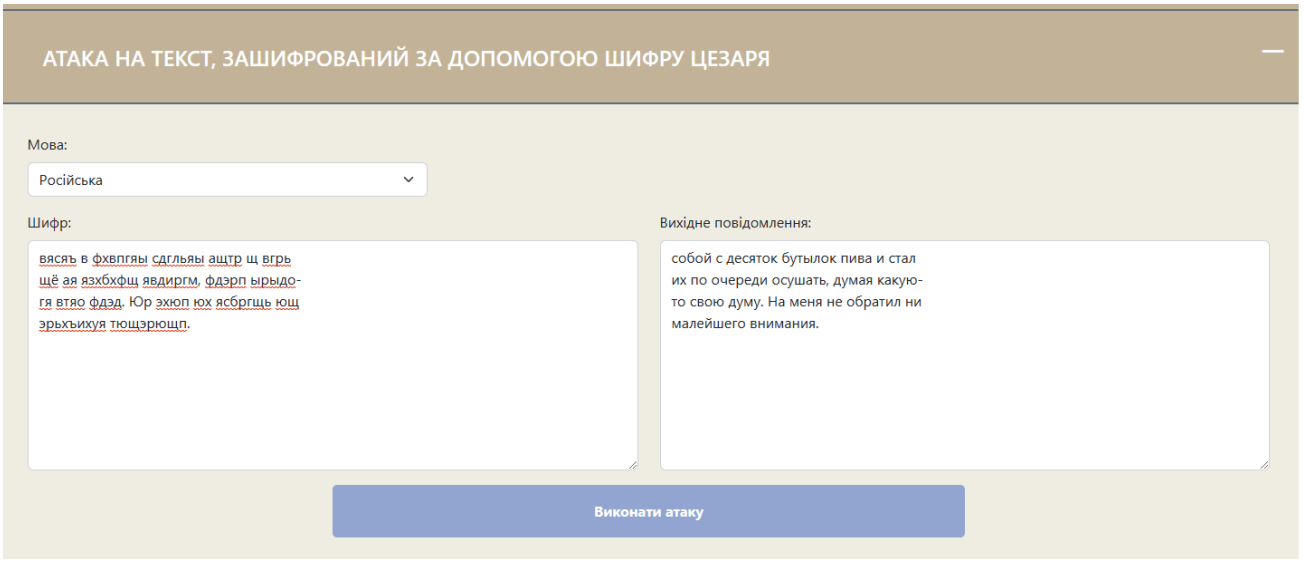

Рисунок 6 – Приклад виконання атаки на текст, зашифрований за допомогою шифру Цезаря

Висновки: Опрацював навички з використання простих алгоритмів шифрування для застосування в практичних цілях, створив програмну реалізацію шифру Цезаря та програмну реалізацію алгоритму, що виконує атаку на повідомлення, зашифроване за допомогою шифу Цезаря.

Контрольні питання

1. Що таке атака (криптоаналіз)?

Криптоаналіз – наука про методи отримання вихідного значення зашифрованої інформації, не маючи доступу до секретної інформації (ключа), необхідної для цього. У більшості випадків під цим мається на увазі знаходження ключа.

2. Які вимоги існують до основних методів криптоаналізу, опишіть їх?

Головними вимогами до основних методів криптоаналізу є те що вони повинні бути:

- Ефектиними за часом (час, витрачений на кількість обчислювальних кроків наприклад, тестових шифрувань, які необхідно виконати);
- Ефективними за використаною пам’яттю (обсяг пам'яті, необхідний для виконання атаки);
- Ефектиними за кількостю даних (Кількість і тип відкритих і зашифрованих текстів, необхідних для методу криптоаналізу).

3. Опишіть алгоритм Цезаря

Під час шифрування за допомогою алгоритму Цезаря кожен символ вихідного тексту замінюється іншим, віддаленим від нього в алфавіті на фіксоване число позицій. Наприклад якщо ключем у алгоритмі є 1 і мова повідомлення є англійською то усі букви “a” стають буквами “b”.

4. Опишіть алгоритм “Одноразовий блокнот”

Алгоритм “Одноразовий блокнот” використовує істинно випадковий секретний симетричний ключ що є одноразовим (не може бути використаний для шифрування двох повідомлень) та має довжину, що дорівнює довжині тексту для шифрування. Ключ та текст для шифрування переводяться у числові вирази (у двійковій або дясятковій системах счислення), а потім використовується XOR операція (або додавання/віднімання по модулю 10 у разі використання десяткової системи счислення).

5. Опишіть використання методу повного перебору для атаки на алгоритм Цезаря.

Алгоритм Цезаря є алгоритмом моноалфавітної заміни та ключ є цілочисельним значенням зсуву. Наприклад, у разі використання російського алфавіту з ключем 2 буква “а” стає буквою “в” і так далі. Цей алгоритм можливо досить легко взламати за допомогою частотного аналізу бо у нас є усього n можливих варіантів зсуву (n – кількість літер у алфавіті). Тому можливо виконати зсув шифру n разів і виконати частотний аналіз на отриманих кандидатах у вихідне повідомлення.

6. Чому повний перебір не використовується при атаці на алгоритм шифрування “Одноразовий блокнот”?

Повний перебір не використовується бо варіантів ключа може бути безліч та і варіантів отриманого повідомлення тоже може бути безліч і не можливо далі використати частотний аналіз або інші методи для з’ясування того, що це і є вихідне повідомлення.