

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Информационные сети. Основы безопасности»

ОТЧЁТ
к лабораторной работе № 1

Выполнил студент группы 053505
Слуцкий Никита Сергеевич

Проверил ассистент
каф.информатики
Протьюко Мирослав Игоревич

Минск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Теоретические сведения	5
1.1 Шифр Цезаря	5
1.2 Шифр Виженера.....	5
2 Выполнение работы	6
3 Тестирование программного продукта	7
Заключение	8
ПРИЛОЖЕНИЕ А Листинг кода.....	9

ВВЕДЕНИЕ

Целью данной лабораторной работы ставится изучить принцип работы примитивных алгоритмов шифрования Цезаря и Виженера, реализовать изученный алгоритм в программном продукте и проверить корректность реализованного алгоритма на предоставленных тестовых данных.

1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Шифр Цезаря

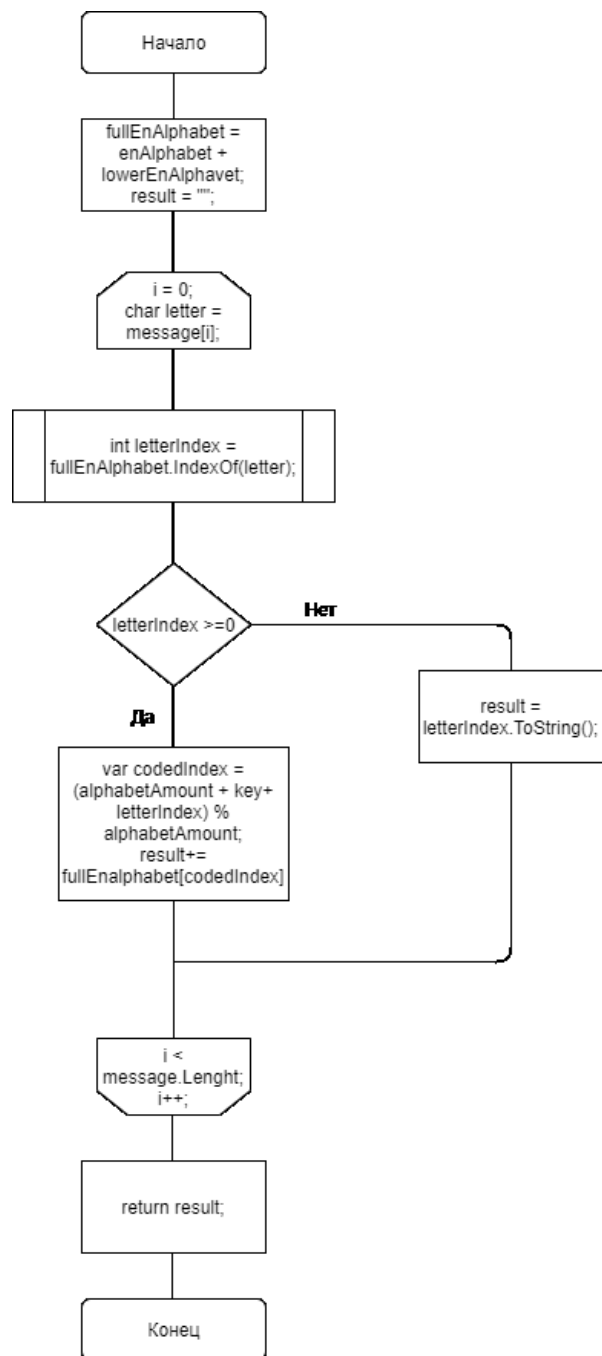
Шифр Цезаря, также известный, как шифр сдвига, код Цезаря или сдвиг Цезаря – один из самых простых и наиболее широко известных методов шифрования.

Шифр Цезаря – это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом находящимся на некотором постоянном числе позиций левее или правее него в алфавите. Например, в шифре со сдвигом 4 А была бы заменена на Г, Б станет Д, и так далее.

1.2 Шифр Виженера

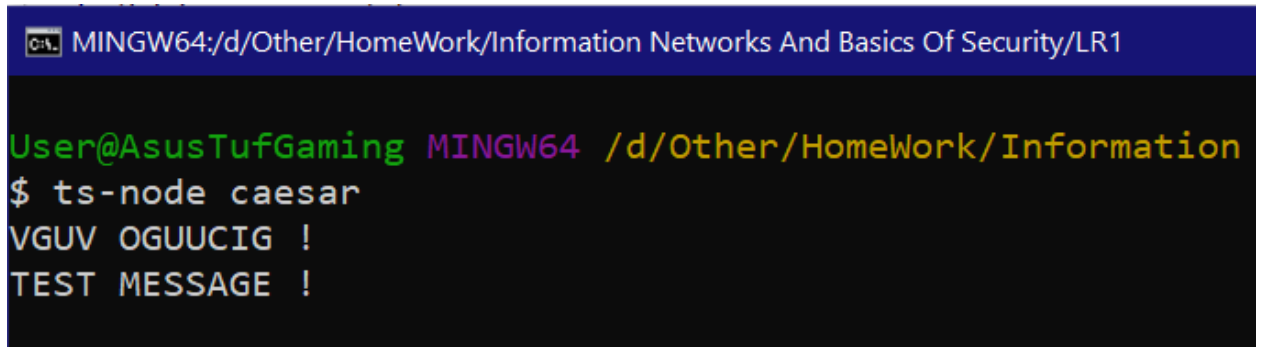
Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для простоты восприятия можно считать, что просто для каждой буквы в сообщении применяется разный сдвиг. Этот сдвиг обусловлен номером соответствующей буквы в ключе. Ключ имеет ту же длину, что и длина сообщения. Для упрощения в данной работе все небуквенные символы будут игнорироваться и оставаться необработанными.

2 ВЫПОЛНЕНИЕ РАБОТЫ



3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

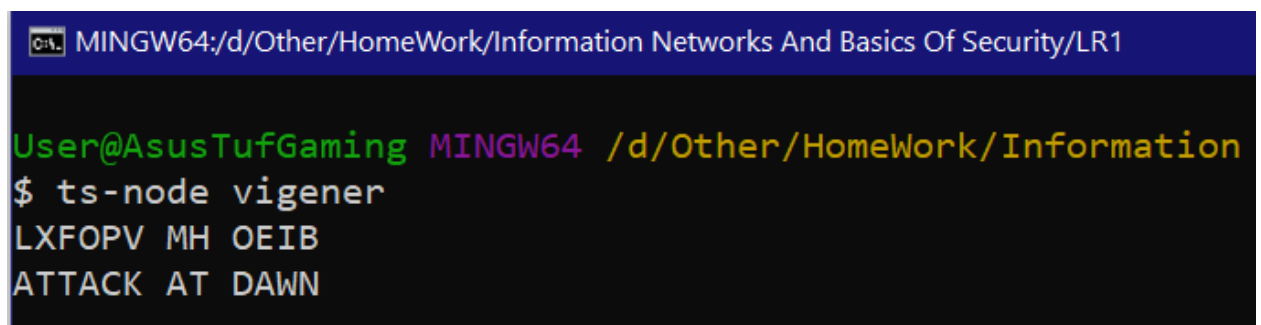
Программный продукт был разработан на языке программирования TypeScript и исполнен в среде NodeJS. На рисунках 1-2 представлен скриншот вывода работы программного продукта в консольном окне для демонстрации работы шифра Цезаря и шифра Виженера соответственно.



```
MINGW64:/d/Other/HomeWork/Information Networks And Basics Of Security/LR1

User@AsusTufGaming MINGW64 /d/Other/HomeWork/Information
$ ts-node caesar
VGUV OGUUCIG !
TEST MESSAGE !
```

Рисунок 1. Результат вывода программного продукта для шифра Цезаря



```
MINGW64:/d/Other/HomeWork/Information Networks And Basics Of Security/LR1

User@AsusTufGaming MINGW64 /d/Other/HomeWork/Information
$ ts-node vigenere
LXFOPV MH OEIB
ATTACK AT DAWN
```

Рисунок 2. Результат вывода программного продукта для шифра Виженера

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы на практике был реализован достаточно примитивный алгоритм шифрования Цезаря, а также его более устойчивая версия – алгоритм шифрования Виженера. И если в шифре Цезаря длина ключа – условно 1, то в шифре Виженера длина ключа равняется длине сообщения. Соответственно, такой шифр при отсутствии ключа практически невозможно раскодировать в читаемое сообщение. Причём на этапе шифрования в шифр Виженера необязательно, собственно, составлять таблицу из строк, содержащих все сдвиги алфавита. Это относительно трудоёмкая операция, которая в данном случае избыточная и также увеличивает время отработки алгоритма.

Цели лабораторной работы можно считать достигнутыми. Работа выполнена.

ПРИЛОЖЕНИЕ А

Листинг кода

```
const encryptByCaesar = (message: string, key: number): string => {
  const transformLetter = (initial: string): string => {
    const initialIndex: number = ALPHABET.indexOf(initial)
    if (initialIndex === -1) {
      return initial
    }

    return ALPHABET[(initialIndex + key) % ALPHABET.length]
  }

  return Array.from(message).map((symbol: string): string => transformLetter(symbol)).join('')
}

const decryptFromCaesar = (encryptedMessage: string, key: number): string => {
  const transformLetter = (initial: string): string => {
    const initialIndex: number = ALPHABET.indexOf(initial)
    if (initialIndex === -1) {
      return initial
    }

    return ALPHABET[(initialIndex + ALPHABET.length - key) % ALPHABET.length]
  }

  return Array.from(encryptedMessage).map((symbol: string): string => transformLetter(symbol)).join('')
}

const main = (): void => {
  const testMessage: string = 'TEST MESSAGE !'

  const encrypted: string = encryptByCaesar(testMessage, 2)
  const decrypted: string = decryptFromCaesar(encrypted, 2)

  console.log(encrypted)
  console.log(decrypted)
}

const generateFullKey = (initialKey: string, length: number): string => {
  let response: string = ''

  while (response.length < length) {
    response += initialKey
  }

  response = response.slice(0, length)

  return response
}

const encryptByVigener = (message: string, fullKey: string): string => {
  const transformLetter = (symbol: string, index: number): string => {
    const initialIndex: number = ALPHABET.indexOf(symbol)

    if (initialIndex === -1) {
      return symbol
    }

    const shiftValue: number = ALPHABET.indexOf(fullKey[index])
    const finalIndex: number = (initialIndex + shiftValue) % ALPHABET.length
    return ALPHABET[finalIndex]
  }

  return [...message].map(transformLetter).join('')
}

const decryptFromVigener = (message: string, fullKey: string): string => {
  const transformLetter = (symbol: string, index: number): string => {
    const initialIndex: number = ALPHABET.indexOf(symbol)

    if (initialIndex === -1) {
```



```

        return symbol
    }

    const shiftValue: number = ALPHABET.indexOf(fullKey[index])

    const finalIndex: number = (initialIndex + ALPHABET.length - shiftValue) % ALPHABET.length

    return ALPHABET[finalIndex]
}

return [...message].map(transformLetter).join('')
}

const main = (): void => {
    const baseKey: string = 'LEMON'

    const message: string = 'ATTACK AT DAWN'
    const fullKey: string = generateFullKey(baseKey, message.length)

    const encrypted: string = encryptByVigener(message, fullKey)
    console.log(encrypted)

    const decrypted: string = decryptFromVigener(encrypted, fullKey)
    console.log(decrypted)
}

```