Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Отчёт по лабораторной работе №15**

“Исследование методов текстовой стеганографии”

**Выполнил:** студент 3 курса

4 группы специальности ПОИТ

Мозолевский Александр Дмитриевич

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2023

**Разработать авторское приложение, реализующее один из методов текстовой стеганографии на основе модификации пространственно-геометрических параметров текста-контейнера. Варианты заданий приведены в таблице. Дополнительные параметры согласуются с преподавателем.**

Реализация приложения:

|  |
| --- |
| package main  import (  "archive/zip"  "bufio"  "bytes"  "fmt"  "io/ioutil"  "log"  "strings"  )  const (  containerFilename = "container.docx"  outputFilename = "output.docx"  )  func encrypt(message string) {  containerFile, err := zip.OpenReader(containerFilename)  if err != nil {  log.Fatal(err)  }  defer containerFile.Close()  containerBuffer := new(bytes.Buffer)  for \_, file := range containerFile.File {  if file.FileInfo().IsDir() {  continue  }  fileReader, err := file.Open()  if err != nil {  log.Fatal(err)  }  contentBytes, err := ioutil.ReadAll(fileReader)  if err != nil {  log.Fatal(err)  }  content := string(contentBytes)  encryptedContent := modifySpaces(content, message)  writer := bufio.NewWriter(containerBuffer)  \_, err = writer.WriteString(encryptedContent)  if err != nil {  log.Fatal(err)  }  writer.Flush()  fileReader.Close()  }  err = ioutil.WriteFile(outputFilename, containerBuffer.Bytes(), 0644)  if err != nil {  log.Fatal(err)  }  fmt.Println("Encryption completed. Output file:", outputFilename)  }  func decrypt() {  encryptedFile, err := zip.OpenReader(outputFilename)  if err != nil {  log.Fatal(err)  }  defer encryptedFile.Close()  messageBuffer := new(bytes.Buffer)  for \_, file := range encryptedFile.File {  if file.FileInfo().IsDir() {  continue  }  fileReader, err := file.Open()  if err != nil {  log.Fatal(err)  }  contentBytes, err := ioutil.ReadAll(fileReader)  if err != nil {  log.Fatal(err)  }  content := string(contentBytes)  decryptedMessage := recoverMessage(content)  \_, err = messageBuffer.WriteString(decryptedMessage)  if err != nil {  log.Fatal(err)  }  fileReader.Close()  }  fmt.Println("Decryption completed. Message:", messageBuffer.String())  }  func modifySpaces(content, message string) string {  words := strings.Fields(content)  characters := []rune(message)  for i := 0; i < len(words) && i < len(characters); i++ {  word := words[i]  character := characters[i]  spacesCount := strings.Count(word, " ")  desiredSpaces := int(character) - spacesCount  if desiredSpaces > 0 {  words[i] = word + strings.Repeat(" ", desiredSpaces)  } else if desiredSpaces < 0 {  words[i] = word[:len(word)+desiredSpaces]  }  }  return strings.Join(words, " ")  }  func recoverMessage(content string) string {  words := strings.Fields(content)  messageBuffer := new(bytes.Buffer)  for \_, word := range words {  spacesCount := strings.Count(word, " ")  messageBuffer.WriteRune(rune(spacesCount))  }  return messageBuffer.String()  }  func main() {  message := "Mozolevskiy"  encrypt(message)  decrypt()  } |

Результат выполнения:

