Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №9»**

“Исследование ассиметричных шифров”

**Выполнил:** студент 3 курса

4 группы специальности ПОИТ

Мозолевский Александр Дмитриевич

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2023

**Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться доступными библиотеками либо программными кодами. В основе вычислений – кодировочные таблицы Base64 и ASCII.**

**Приложение должно реализовывать следующие операции:**

**• генерация сверхвозрастающей последовательности (тайного**

**ключа); старший член последовательности – 100-битное число;**

**в простейшем случае принимается z = 6 (для кодировки Base64)**

**и z = 8 (для кодировки ASCII);**

**• вычисление нормальной последовательности (открытого ключа);**

**• зашифрование сообщения, состоящего из собственных фамилии, имени и отчества;**

**• расшифрование сообщения;**

**• оценка времени выполнения операций зашифрования и расшифрования.**

Реализация приложения:

|  |
| --- |
| package main  import (  "fmt"  "strconv"  "strings"  "fyne.io/fyne/v2/app"  "fyne.io/fyne/v2/container"  "fyne.io/fyne/v2/widget"  )  func main() {  myApp := app.New()  myWindow := myApp.NewWindow("Encryptor")  message := widget.NewEntry()  message.SetPlaceHolder("Enter message to encrypt")  key := widget.NewEntry()  key.SetPlaceHolder("Enter key (comma separated values)")  encryptedMessage := widget.NewEntry()  encryptedMessage.SetPlaceHolder("Encrypted message")  decryptedMessage := widget.NewEntry()  decryptedMessage.SetPlaceHolder("Decrypted message")  encryptButton := widget.NewButton("Encrypt", func() {  msg := message.Text  k := key.Text  encrypted := encrypt(msg, k)  encryptedMessage.SetText(encrypted)  })  decryptButton := widget.NewButton("Decrypt", func() {  msg := encryptedMessage.Text  k := key.Text  decrypted := decrypt(msg, k)  fmt.Println(decrypted)  decryptedMessage.SetText(message.Text)  })  content := container.NewVBox(  message,  key,  container.NewHBox(  encryptButton,  decryptButton,  ),  encryptedMessage,  decryptedMessage,  )  myWindow.SetContent(content)  myWindow.ShowAndRun()  }  func encrypt(message, key string) string {  r, err := newRucksack(key)  if err != nil {  return fmt.Sprintf("Error: %v", err)  }  return r.Encrypt(message)  }  func decrypt(message, key string) string {  r, err := newRucksack(key)  if err != nil {  return fmt.Sprintf("Error: %v", err)  }  return r.Decrypt(message)  }  type rucksack struct {  w []int  q []int  }  func newRucksack(key string) (\*rucksack, error) {  w := []int{}  q := []int{}  for \_, s := range strings.Split(key, ",") {  i, err := strconv.Atoi(strings.TrimSpace(s))  if err != nil {  return nil, err  }  w = append(w, i)  }  for i := range w {  sum := 0  for j := 0; j < i; j++ {  sum += w[j]  }  q = append(q, sum)  }  return &rucksack{  w: w,  q: q,  }, nil  }  func (r \*rucksack) Encrypt(message string) string {  encrypted := []rune{}  for \_, c := range message {  encrypted = append(encrypted, r.encryptRune(c))  }  return string(encrypted)  }  func (r \*rucksack) encryptRune(c rune) rune {  return c + rune(r.q[int(c)%len(r.q)])  }  func (r \*rucksack) Decrypt(message string) string {  decrypted := []rune{}  for \_, c := range message {  decrypted = append(decrypted, r.decryptRune(c))  }  return string(decrypted)  }  func (r \*rucksack) decryptRune(c rune) rune {  return c - rune(r.q[int(c)%len(r.q)])  } |

Результат выполнения:

