МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ ЦИФРОВЫХ ТЕХНОЛОГИЙ, ЭЛЕКТРОНИКИ И ФИЗИКИ (ИЦТЭФ)

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И ЭЛЕКТРОНИКИ (ВТиЭ)

**Отчет по лабораторной работе № 6**

по курсу “Защита информация”

**“Целостность информации. Простейшие методы стеганографии. Хэширование.”**

Выполнил студент 506 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Осипенко

Проверил: ст. преп. кафедры ИБ.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.С. Ладыгин.

Лабораторная работа защищена

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

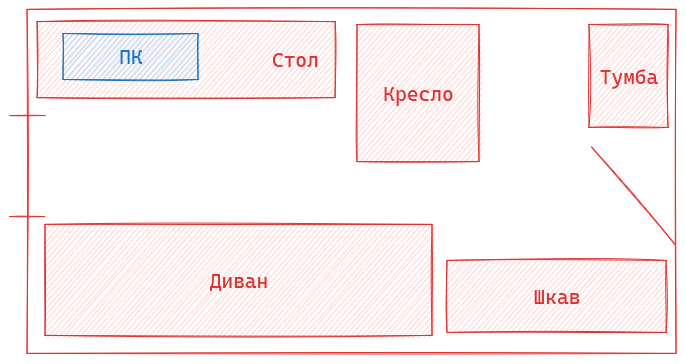
Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Задание

1. Примените LSB к одному из изображений из вашей Модели угроз, предварительно сохранив его в удобном формате.

2. Используя одну из рассмотренных хэш-функций, показать различие или совпадение хэшей двух изображений.

# Стеганография



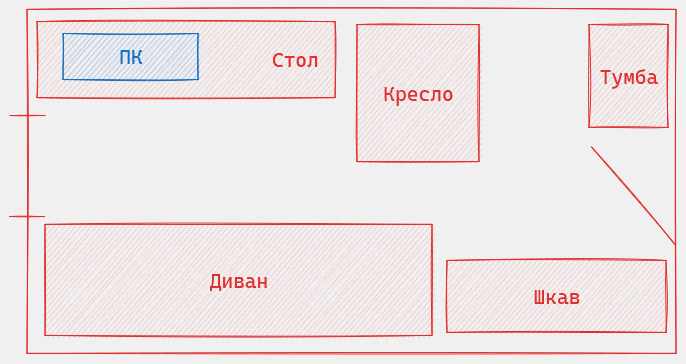


Рис. 1.1 Пример применения алгоритма стеганографии LSB: сверху оригинал, снизу изображение с закодированным текстом

Встраиваемый текст: *Прежде всего, повышение уровня гражданского сознания, а также свежий взгляд на привычные вещи - безусловно открывает новые горизонты для системы массового участия. Для современного мира понимание сути ресурсосберегающих технологий позволяет оценить значение поэтапного и последовательного развития общества. В частности, социально-экономическое развитие однозначно определяет каждого участника как способного принимать собственные решения касаемо форм воздействия.*

Распознанный текст: *Прежде всего, повышение уровня гражданского сознания, а также свежий взгляд на привычные вещи - безусловно открывает новые горизонты для системы массового участия. Для современного мира понимание сути ресурсосберегающих технологий позволяет оценить значение поэтапного и последовательного развития общества. В частности, социально-экономическое развитие однозначно определяет каждого участника как способного принимать собственные решения касаемо форм воздействия.*

**Листинг 1.1:** Исходный код кодировщика и декодировщика.

func Decode(inputPath string) (string, error) {

inputFile, err := os.Open(inputPath)

if err != nil {

return "", err

}

defer inputFile.Close()

img, err := png.Decode(inputFile)

if err != nil {

return "", err

}

strBuf := bytes.NewBufferString("")

err = WalkImage(img, func(x, y int) error {

var symb uint32

r, g, b, \_ := img.At(x, y).RGBA()

symb |= b & 0x000f

symb <<= 4

symb |= g & 0x000f

symb <<= 4

symb |= r & 0x000f

strBuf.WriteRune(rune(symb))

return nil

})

return strBuf.String(), err

}

func Encoder(msg, inputPath, outPath string) error {

inputFile, err := os.Open(inputPath)

if err != nil {

return err

}

defer inputFile.Close()

srcImg, err := png.Decode(inputFile)

if err != nil {

return err

}

bounds := srcImg.Bounds()

dstImg := image.NewNRGBA(image.Rect(0, 0, bounds.Max.X, bounds.Max.Y))

if bounds.Max.X\*bounds.Max.Y <= len(msg) {

return errors.New("message size >= image bounds")

}

strBuf := bytes.NewBufferString(msg)

err = WalkImage(srcImg, func(x, y int) error {

var newR, newG, newB uint8

r, g, b, a := srcImg.At(x, y).RGBA()

newR = uint8(r) & 0xf0

newG = uint8(g) & 0xf0

newB = uint8(b) & 0xf0

if rn, \_, err := strBuf.ReadRune(); err == nil {

newR |= uint8(rn & 0x000f)

newG |= uint8(rn & 0x00f0 >> 4)

newB |= uint8(rn & 0x0f00 >> 8)

}

dstImg.SetNRGBA(x, y, color.NRGBA{

R: newR,

G: newG,

B: newB,

A: uint8(a),

})

return nil

})

if err != nil {

return err

}

outFile, err := os.Create(outPath)

if err != nil {

return err

}

defer outFile.Close()

err = png.Encode(outFile, dstImg)

if err != nil {

return err

}

return nil

}

func WalkImage(img image.Image, mapper func(x, y int) error) error {

for y := img.Bounds().Min.Y; y < img.Bounds().Max.Y; y++ {

for x := img.Bounds().Min.X; x < img.Bounds().Max.X; x++ {

err := mapper(x, y)

if err != nil {

return err

}

}

}

return nil

}

# Хеширование

В качестве алгоритма хеширования был выбран – MD5.

Алгоритм принимает на вход сообщение произвольной длины и выдает в качестве вывода — 128-битный «отпечаток пальца» или «дайджест сообщения» ввода. Предполагается, что вычислительно невозможно получить два сообщения, имеющие один и тот же дайджест сообщения, или для создания любого сообщение, имеющее заданный заранее заданный целевой дайджест сообщения. MD5. Алгоритм предназначен для приложений цифровой подписи, где большой файл должен быть «сжат» безопасным способом, прежде чем он будет зашифровано закрытым (секретным) ключом в криптосистеме с открытым ключом например РСА.

Таблица 2.1 Хеши изображений

| **Название файла** | **Хеш файла** |
| --- | --- |
| Исходное изображение | 0x27388a544a10e9527c50af846676803d |
| Закодированное изображение | 0x7321a06beea8e34d8f16227ce0e3685c |

**Листинг 2.1:** Исходный код функции хеширования с помощью алгоритма MD5.

func HashFile(filePath string) []byte {

data, err := os.ReadFile(filePath)

if err != nil {

return make([]byte, 0)

}

res := md5.Sum(data)

return res[:]

}

# Вывод

В ходе данной работы были изучены алгоритм стеганографии информации в изображение и алгоритмы хеширования данных.

# Контрольные вопросы

1. В какой деятельности могла бы пригодиться стеганография для вас?

– Сдача лаб по ИБ; Передача/хранение некоторой конфиденциальной информации, закодированной в красивые эротические изображения, которые нежеланному пользователю будет стыдно смотреть на них или он не подумает их проверять.

2. Не используя сети Интернет попробуйте придумать свой способ скрыть сообщение в цифровом контейнере. Опишите в 3-5 предложениях.

– Файлы имеют бинарные заголовки. Данные заголовки имеют в себе пустую, неиспользуемую часть. В данные пустые промежутки можно вставить сообщение. Также информацию можно сохранить в объектных файлах, компиляторы которых сохраняют в них комментарии.

3. Какая хэш-функция наименее защищена от подбора исходного слова на основе хэша?

– MD5.

4. В каких задачах наиболее применим md5?

– Проверка целостности файлов; Сравнение файлов.