**УО «Белорусский государственный технологический университет»**

Факультет **информационных технологий**

Кафедра **информационных систем и технологий**

Специальность **1-40 05 01-03 «Информационные системы и технологии (издательско-полиграфический комплекс)»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

**по дисциплине «**Защита информации и надежность информационных систем»

**тема «**Реализация и исследование метода кодирования изображений на основе принципа кубика Рубика»

|  |  |  |
| --- | --- | --- |
| **Исполнитель** |  | |
| Студент 3 курса группы1 | подпись, дата | В. В. Пригодич  инициалы и фамилия |
|  |  |  |
| **Руководитель** |  |  |
| Ассистент | подпись, дата | М. Г. Савельева  инициалы и фамилия |

|  |  |
| --- | --- |
| **Курсовая работа защищена с оценкой** | **\_\_\_\_\_\_\_\_\_\_** |
| **Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  (подпись) | М. Г. Савельева  инициалы и фамилия |

Минск 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**ЗАДАНИЕ**

**к курсовому проектированию по дисциплине**«Защита информации и надежность информационных систем»

Специальность: 1-40 05 01 03 «Информационные системы и технологии (издательство-полиграфический комплекс)»

Курс: 3, группа: 1

Студент: Пригодич Вера Валерьевна

**Тема:** Реализация и исследование метода кодирования изображений на основе принципа кубика Рубика

**1. Срок сдачи студентом законченной работы**: «05» мая 2023г.

**2. Исходные данные к проекту:**

**2.1**. Функционально должны быть выполнены следующие задачи:

* Разработка программных функций, реализующих алгоритмы шифрования и дешифрования в соответствии с темой
* Создание приложения с графическими интерфейсом, реализующего разработанные функции
* Сравнение реализованного алгоритма с другими алгоритмами шифрования изображений

**2.2. Требования:**

* Необходимо провести аналитический обзор литературы по теме проекта
* Необходимо описать сферу применимости метода
* Программное средство может быть разработано на любом языке
* Архитектура приложения выбирается разработчиком
* Листинги проекта должны содержать комментарии

**3. Содержание расчетно-пояснительной записки:**

* Введение
* Постановка задачи
* Описание метода
* Описание программного средства
* Тестирование программного средства
* Руководство пользователя
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* + Теоретическая часть курсового проекта должна быть представлена в формате MS Word.
  + Оформление записки должно быть согласно правилам.
  + Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio или копии экрана (интерфейс).
  + Полные листинги проекта представляются в приложении.
  + К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и файлы базы данных.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Задание курсового проекта. Титульный лист | 10.02.2023 |  |
| 2 | Введение | 24.02.2023 |  |
| 3 | Аналитический обзор литературы по теме проекта | 10.03.2023 |  |
| 4 | Разработка метода | 22.03.2023 |  |
| 5 | Разработка прототипа программного средства | 30.03.2023 |  |
| 6 | Разработка программного средства | 08.04.2023 |  |
| 7 | Тестирование программного средства | 15.04.2023 |  |
| 8 | Написание руководства пользователя | 22.04.2023 |  |
| 9 | Оформление пояснительной записки | 30.04.2023 |  |
| 10 | Сдача проекта | 05.05.2023 |  |

**5. Дата выдачи задания** «10» февраля 2023г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *М. Г. Савельева*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

**Оглавление**

[Введение 5](#_Toc130555294)

[1 Аналитический обзор литературы 6](#_Toc130555295)

[1.1 Аналоги 6](#_Toc130555296)

[1.1.1 RC4 6](#_Toc130555297)

[1.1.2 Grain 6](#_Toc130555298)

[1.1.3 Алгоритмы на основе динамического хаоса 8](#_Toc130555299)

[2 Описание метода 9](#_Toc130555300)

[2.1 Алгоритм кодирования 9](#_Toc130555301)

[2.2 Алгоритм декодирования 10](#_Toc130555302)

[3 Описание программного средства 11](#_Toc130555303)

Введение

В свете широкого распространения информационных технологий и Интернета возникает множество проблем, связанны с обеспечением безопасности хранения и передачи данных в виде изображений. Существует множество методов защиты информации и обеспечения ее целостности, в том числе криптографические методы. Однако не все из них могут быть эффективными при защите изображений, которые могут содержать конфиденциальную информацию.

Кодирование изображений является важным аспектом защиты графических данных, так как изображения могут содержать конфиденциальную информацию, которую необходимо сохранить в безопасности. В данной работе рассматривается метод кодирования изображений на основе принципа кубика Рубика.

Принцип кодирования изображений на основе кубика Рубика использует свойства кубика Рубика, такие как симметрия и повторяемость, для защиты конфиденциальных данных в виде изображений.

Кубик Рубика является трехмерным пазлом, который состоит из нескольких слоев, каждый из которых можно вращать вокруг одной из трех осей. При этом повороты могут быть выполнены только на углы 90 градусов.

Использование свойств кубика Рубика позволяет перемешать значения пикселей изображения для защиты конфиденциальных данных в виде изображений. При кодировании каждый блок изображения становится нечитаемым, что обеспечивает его защиту от несанкционированного доступа.

Цель данной работы заключается в реализации метода кодирования изображений на основе принципа кубика Рубика и исследовании его эффективности и надежности. Для достижения поставленной цели были выделены следующие задачи:

* реализовать программное средство кодирования изображений на основе принципа кубика Рубика;
* оценить качество полученных изображений после кодирования и сравнить с другими методами кодирования;
* исследовать зависимости качества изображений от параметров метода кодирования;
* исследовать производительность разработанного метода кодирования изображений и сравнить с другими методами кодирования;
* составить руководство пользователя.

1. Аналитический обзор литературы
   1. Аналоги

Поскольку цифровое изображение состоит из битов, как и все другие цифровые данные, криптографический алгоритмы, используемые для защиты других форм информации, могут быть использованы для изображений.

Это означает, что для шифрования изображения может быть использованы следующие алгоритмы: RC4, Grain. Рассмотрим их подробнее.

1.1.1 RC4

RC4 работает на основе ключа переменной длины и оперирует над байтами. Шифрование RC4 осуществляется путем генерации псевдослучайной последовательности, которая затем комбинируется с открытым текстом для создания зашифрованного сообщения.

Ядро алгоритма состоит из функции генерации ключевого потока. Другая часть алгоритма – функция инициализации, которая использует ключ переменной длины *Ki* для создания начального состояния генератора ключевого потока.

В основе алгоритма – размер блока или слова, определяемый параметром *n*. Обычно *n* = 8, но можно использовать и другие значения. Внутренне состояние шифра определяется массивом слов (*S*-блоком) размером 2*n*.

Другими элементами внутреннего состояния являются 2 счетчика (каждый размером в одно слово; обозначим их *i* и *j*) с нулевыми начальными значениями. В основе вычислений лежит операция по mod 2*n*. Генератор ключевого потока RC4 переставляет значения, хранящиеся в *S*, и каждый раз выбирает различное значение из *S* в качестве результата. В одном цикле RC4 определяется одно *n*-битное слово *K* из ключевого потока, которое в последующем суммируется с исходным текстом для получения зашифрованного текста.

Дешифрование производится аналогичным образом путем генерации той же самой псевдослучайной последовательности и ее применения для получения исходного открытого текста.

RC4 считается быстрым и простым в реализации шифром, но у него есть некоторые уязвимости. Например, если использовать один и тот же ключ для шифрования большого количества сообщений, возможна атака на ключ, при которой злоумышленник может восстановить ключ, изучив зашифрованные сообщения. Кроме того, таблица ключей может быть подвергнута атаке, если она не генерируется достаточно случайным образом.

1.1.2 Grain

Grain ­– это потоковый шифр, который был разработан Мартином Шварцем и Юргеном Вихтенбахом в 2008 году.

Алгоритм Grain состоит из двух основных компонентов: генератора ключей и генератора псевдослучайного потока (PRNG). Генератор ключей используется для получения ключа и начального вектора (*IV*), которые затем используются генератором PRNG для генерации псевдослучайного потока.

Генератор ключей включает в себя несколько элементов: 16-битный регистр, 128-битный ключ и 96-битный *IV*. Регистр состоит из 16 ячеек, каждая из которых хранит один бит. Ключ и *IV* объединяются в одну 224-битную последовательность, которая затем используется для заполнения регистра с помощью операции XOR.

Генератор PRNG состоит из трех частей: линейного обратного связного регистра (LFSR), нелинейного фильтра и выходного шифра. LFSR состоит из 128 ячеек, каждая из которых хранит один бит. Нелинейный фильтр осуществляет преобразования над содержимым LFSR и используется для улучшения статистических свойств выходного потока. Выходной шифр объединяет данные из LFSR и нелинейного фильтра для генерации псевдослучайного потока.

В процессе генерации псевдослучайного потока LFSR и нелинейный фильтр работают параллельно. LFSR перемещает биты влево на одну позицию и выполняет операцию XOR с определенными ячейками, чтобы определить новый бит. Нелинейный фильтр включает в себя несколько операций, таких как умножение, сложение и битовые сдвиги, которые используются для преобразования содержимого LFSR и улучшения статистических свойств выходного потока.

Выходной шифр объединяет данные из LFSR и нелинейного фильтра и преобразует их с помощью операции XOR. Это преобразование выполняется для каждого байта псевдослучайного потока. Результатом является 128-битный псевдослучайный поток, который используется для шифрования данных.

В целом, Grain является надежным и безопасным потоковым шифром, который может использоваться для шифрования различных типов данных, включая изображения. Однако, как и любой шифр, Grain имеет свои ограничения и недостатки. Например, он не может использоваться для шифрования больших объемов данных, так как его память ограничена 128 битами.

Стоит отметить, что данные алгоритмы разрабатывались в первую очередь для текстовых данных и не учитывают особенностей кодирования изображения. С учетом того, что размер изображений достаточно большой, а некоторым приложениям необходимо работать в режиме реального времени, процесс шифрования должен осуществляться достаточно быстро. Также стоит учесть, что цифровые изображения сравнительно менее чувствительны, чем текстовые данные, так как однократное изменения пикселей не меняет всего изображения. Другими словами, небольшая модификация цифрового изображения приемлема по сравнению с текстовыми данными. Так как данные алгоритмы шифрования разрабатывались без учета этих требований, они не являются наиболее подходящими для данных целей.

1.1.3 Алгоритмы на основе динамического хаоса

В последние годы получают развитие новые методов шифрования изображений. Распространёнными являются алгоритмы шифрования изображений на основе динамического хаоса. Данные алгоритмы используют хаотические отображения для генерации ключей шифрования. Основная идея заключается в том, что непрерывные хаотические отображения проявляют сходны свойства с дискретными криптографическими системами. Популярными являются такие хаотические отображения, как отображение кота Арнольда, отображение Эно, отображение «тент» и так далее. Например, отображение Лоренца, основанное на трех связанных дифференциальных уравнениях, которые описывают движение жидкости внутри замкнутой системы. Значения переменных, используемых в уравнениях, рассматриваются как координаты точки в трехмерном пространстве. Используя начальные значения для этих переменных, можно смоделировать движение точки во времени. Затем, полученные координаты используются для перестановки и замены пикселей изображения.

Алгоритмы шифрования изображений на основе динамического хаоса имеют ряд преимуществ, включая высокую стойкость к криптоанализу, возможность быстрого шифрования больших объемов данных, а также устойчивость к изменениям в исходном изображении. Тем не менее, алгоритмы шифрования изображений на основе хаоса требуют больших вычислительных затрат. Более того, система хаоса определяется на действительных числах, тогда как криптосистемы определяются на конечных множествах целых чисел. Одномерные хаотические криптосистемы ограничены небольшим пространством ключей и слабой безопасностью.

Каждый алгоритм использует свой подход к шифрованию изображений и может быть использован в зависимости от требований конкретного приложения. На основе выделенных преимуществ и недостатков, для реализации был выбран метод, сочетающий в себе преимущества других аналогов. Проектирование, разработка и тестирование программного средства будет описана в следующих главах.

1. Описание метода
   1. Алгоритм кодирования

Пусть дано изображение *I*0 с глубиной цвета *a-bit* и размером *M* × *N*. Шаги алгоритма кодирования следующие:

1. Сгенерировать 2 случайных вектора *KR* и*KC* длины *M* и *N*, соответственно. Элементы *KR*(*i*)и *KC*(*i*)принимают случайное значение из набора *A* = {0, 1, 2, …, 2*a* – 1}*.* Отметим, что *KR* и*KC* не должны содержать константных значений.
2. Определить количество итераций, *ITERmax*, и инициализировать *ITER* как 0.
3. Инкрементировать счетчик на 1: *ITER* = *ITER* + 1.
4. Для каждой строки *i* изображения *I*0,
   1. подсчитать сумму всех элементов в строке *i*, эта сумма обозначается *a*(*i*)
   2. разделить *a*(*i*) по модулю 2, обозначаем *Ma*(*i*),
   3. строка *i* циклически сдивгается вправо или влево на *KR*(*i*) позиций, по следующему правилу: если *Ma*(*i*) = 0, сдвиг вправо, иначе сдвиг влево.
5. Для каждого столбца *j* изображения *I*0,
   1. подсчитать сумму всех элементов в столбце *j*, эта сумма обозначается 𝛽(*j*)
   2. разделить 𝛽(*j*) по модулю 2, обозначаем *M*𝛽(*j*),
   3. столбец *j* циклически сдвигается вверх или вниз на *KC*(*j*) позиций, по следующему правилу: если *M*𝛽(*j*) = 0, сдвиг вверх, иначе сдвиг вниз.

Шаги 4 и 5 выше создадут закодированное изображения, обозначаемое *I*SCR.

1. Используя вектор *KC*, побитовая операция XOR применяется к каждой строке закодированного изображения *I*SCR используя следующее выражение:

где и представляют побитовую операцию XOR и отображение вектора *KC* слева направо, соответственно.

1. Используя вектор *KR*, побитовая операция XOR применяется к каждому столбцу изображения *I*1 используя следующее выражение:

где представляют отображение вектора *KR* слева направо.

1. Если *ITER* = *ITERmax*, тогда зашифрованное сообщение *IENC* создано и процесс кодирования завершен; иначе алгоритм переходит к шагу 3.

Векторы *KR*,*KC* и максимальное количество итераций *ITERmax* считаются секретными ключами в данном алгоритме шифрования. Однако для быстрой работы алгоритма предпочтительно установиться *ITERmax* = 1 (единственная итерация). И наоборот, если *ITERmax* > 1, тогда алгоритм более надежный, поскольку пространство ключей больше, чем для *ITERmax* = 1

* 1. Алгоритм декодирования

Расшированное изображение, *I*0, восстанавливается из зашифрованного изображения, *IENC* , и секретных ключей, *KR*,*KC* и *ITERmax* как показано далее:

1. Инициализировать *ITER* = 0.
2. Инкрементировать счетчик на 1: *ITER* = *ITER* + 1.
3. Побитовая операция XOR применяется к вектору *KR* и каждому столбцу изображения *IENC* используя следующее выражение:
4. Далее, используя вектор *KC*, побитовая операция XOR применяется к каждой строке изображения *I*1 используя следующее выражение:
5. Для каждого столбца *j* изображения *ISCR*,
   1. подсчитать сумму всех элементов в столбце *j*, эта сумма обозначается 𝛽*SCR*(*j*)
   2. разделить 𝛽*SCR*(*j*) по модулю 2, обозначаем *M*𝛽(*j*),
   3. столбец *j* циклически сдвигается вверх или вниз на *KC*(*j*) позиций, по следующему правилу: если *M*𝛽(*j*) = 0, сдвиг вниз, иначе сдвиг вверг.
6. Для каждой строки *i* изображения *I*0,
   1. подсчитать сумму всех элементов в строке *i*, эта сумма обозначается *aSCR*(*i*)
   2. разделить *aSCR*(*i*) по модулю 2, обозначаем *Ma*(*i*),
   3. строка *i* циклически сдивгается вправо или влево на *KR*(*i*) позиций, по следующему правилу: если *Ma*(*i*) = 0, сдвиг влево, иначе сдвиг вправо.

Если *ITER* = *ITERmax*, тогда зашифрованное сообщение *IENC* расшифровано и процесс декодирования завершен; иначе алгоритм переходит к шагу 2.

1. Описание программного средства

В данной главе описаны технологии, которые были использованы для разработки программного средства, для кодирования изображений на основе принципа кубика Рубика. Для начала рассмотрим системы и технологии, которые мы будем использовать в данном курсовом проекте.

Python 3 – высокоуровневый язык программирования общего назначения. Стандартная библиотека включает большой набор полезных переносимых функций. Дополнительные возможности могут реализовываться посредством обширного количества сторонних библиотек. В данном проекте были использованы следующие библиотеки:

* Pillow – оригинальная библиотека Python для работы с изображениями;
* NumPy – библиотека, применяемая для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй;
* Tkinter – это кроссплатформенная библиотека для разработки графического интерфейса на языке Python.

Текстовый редактор Visual Studio Code ­– это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

1. Тестирование программного средства
2. Руководство пользователя