МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

|  |  |
| --- | --- |
| на тему | Разработка программного обеспечения для шифрования файлов |
|  | |
|  | |

|  |  |
| --- | --- |
| выполнена | Степановым Николаем Нюргустановичем |
| фамилия, имя, отчество студента в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки/ специальности | 38.03.05 |  | Бизнес-информатика |
|  | код |  | наименование направления подготовки/ специальности |
|  | | | |
| наименование направления подготовки/ специальности | | | |
| направленности | Бизнес-информатика | | |
|  | наименование направленности | | |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы № | 8721К |  |  |  | Н.Н.Степанов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ   
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ БАКАЛАВРА

|  |  |  |  |
| --- | --- | --- | --- |
| студенту группы | 8721К |  | Степанов Николай Нюргустанович |
|  | номер |  | фамилия, имя, отчество |

|  |  |
| --- | --- |
| на тему | Разработка программного обеспечения для шифрования файлов |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| утвержденную приказом ГУАП от |  | № |  |

|  |  |
| --- | --- |
| Цель работы: | Заключается в повышении защиты данных от несанкционированного |
| доступа сторонними лицами | |
|  | |

|  |  |
| --- | --- |
| Задачи, подлежащие решению: |  |
|  | |
|  | |

|  |  |
| --- | --- |
| Содержание работы (основные разделы): |  |
|  | |
|  | |
|  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Срок сдачи работы « |  | » |  | 20 | 21 |

Задание принял(а) к исполнению

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| студент группы № | 8721К |  |  |  | Н.Н.Степанов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Реферат

Актуальность темы.

Цель и задачи работы, которые определяются исходя из темы ВКР.

Объект и предмет исследования или разработки.

Оглавление

[Введение. 5](#_Toc68731130)

[1. Обзор области 6](#_Toc68731131)

[1.1. Разработка алгоритмов 6](#_Toc68731132)

[2. 16](#_Toc68731133)

# Введение

# Обзор области

## Разработка алгоритмов

Шифрование большинства объектов файла word происходит по стандартному решению. Шифрование текста - это очень древняя задача, которую за всё это время смогли решить множеством способов, каждый из которых имеет преимущество в своей области. Важные свойства шифровального аппарата - это скорость зашифровки и расшифровки, а также качество безопасности. Эти два понятия связаны межу собой и являются взаимоисключающими, если рассматривать со стороны “безопасность или скорость”. Чем меньше мы хотим тратить времени на шифрование и расшифровку, тем больше уязвим шифр для некоторых типов атак.

Не будем уделять много времени тексту, а рассмотрим шифрование на примере рисунка. Шифрование изображения может показаться абсолютно иной задачей, не имеющим ничего общего с текстовым шифрованием. Но давайте рассмотрим более подробно данный процесс и сделаем обоснованный вывод.

Как мы знаем любое изображение состоит из пикселей, самого простого объекта в изображении. Каждый пиксель выводит информацию о цвете, который должен находиться в данной части изображения. Есть несколько способов хранения информации о цвете, один из них RGBA. С форматом RGB знакомы многие, он хранит цвет в виде наложения друг на друга трёх основных цветов, красный зеленый и синий. Отличие RGBA в том, что помимо трёх предыдущих значений, каждый пиксель имеет информацию еще и о прозрачности относительно других объектов. Наглядный пример как хранятся значения пикселей в формате RGB можно увидеть на рисунке 1.

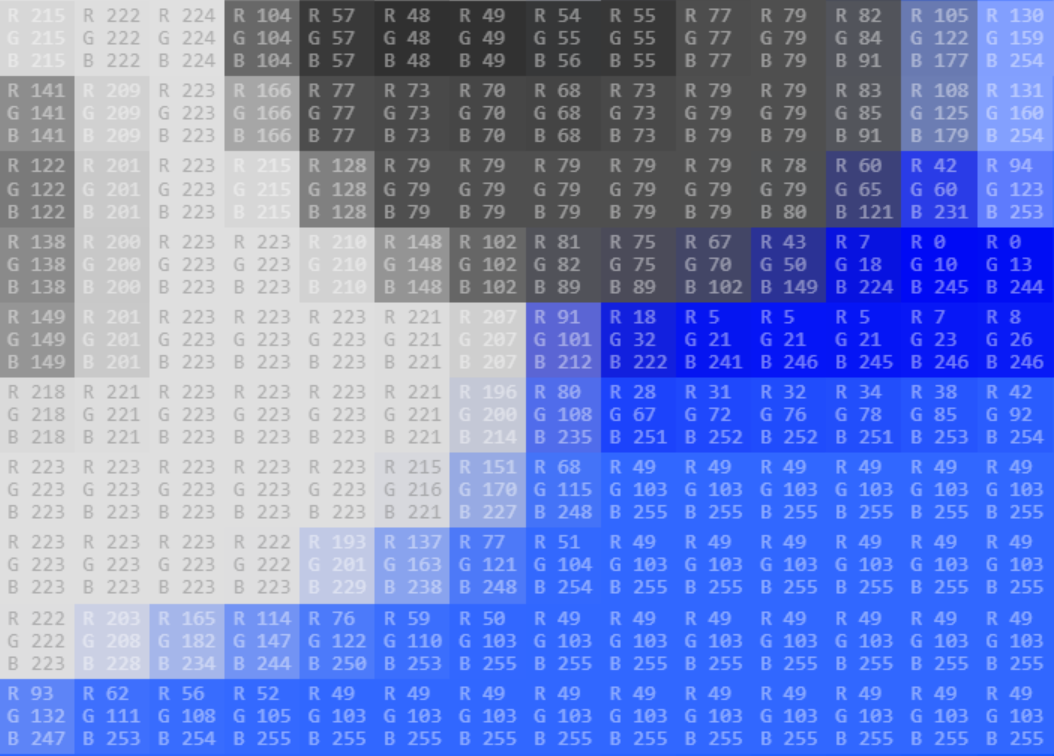


Рисунок 1.- RGB

Итак, получив значения каждого пикселя мы снова сможем свести задачу к текстовому виду для шифрования. Но хранить зашифрованную картинку в текстовом виде очень объёмно, так как даже в самой маленькой картинке 100 на 100 пикселей, уже имеется десять тысяч пикселей, в каждом из которых по 4 значения - итого сорок тысяч значений. А при числовых значениях от одного символа до трёх, данное число еще можно смело умножать на два. Такие объемы текста при большом числе изображений приведут к конечному файлу размером в двести страниц, а самое главное уйдет много времени для вставки и получения такого текста. На мой взгляд лучше всего будет зашифровать картинку и вернуть её в вид изображения.

Если исходить из выше сказанного, то шифрование должно приводить к результату в виде последовательности из наборов по 4 числа. Каждый из наборов представляет собой один пиксель. Данные подлежащие к шифрованию в изображении:

1. Значения пикселей
2. Ширина изображения
3. Длина изображения

Так как, помимо пикселей, изображение имеет еще ширину и длину, такие данные также могут иметь важность при атаке третьих лиц. Именно поэтому конечное изображение будет иметь случайную длину и ширину в определенных рамках.

Для расшифровки все данные должны содержаться в шифре, но мы планируем хранить шифр в виде изображения, при этом сохранив данные исходной ширины и длины рисунка. Существуют алгоритмы, которые могут шифровать значения с неким сокращением по объёму. То есть зашифровав 100 пикселей, получится, к примеру, 99 значащих и один резервный, не несущий никакого значения для расшифровки. При таком подходе, в конечном счете, будет разница между исходным числом пикселей и зашифрованным, именно в них и можно хранить все дополнительные данные.

Единственный минус такого подхода - это одинаковое число пикселей исходного и конечного изображения, что математически сужает круг поиска исходных размеров ширины и длины картинки. Перемножая дину на ширину, можно более точно подобрать исходные размеры, особенно, если они были сделаны по стандартам ширины экранов устройств.

Решением, на мой взгляд, является изменение числа пикселей в конечном изображении. При случайном подборе длины и ширины из неких промежутков, нам будет известна информация о новом количестве пикселей, которое, с большой вероятностью, будет больше, чем число зашифрованных пикселей. В редких случаях оно может быть равно, но никогда не меньше. Так как число зашифрованных пикселей не равно числу исходных пикселей, то нельзя утверждать, что число пикселей зашифрованной картинки всегда больше исходной.

Возникает два исхода, если есть доступные для записи информации пиксели, то записываем данные о ширине и длине исходного изображения в них. Если же свободных пикселей нет или недостаточно, то добавляем новый ряд пикселей в ширину или длину и заполняем данные туда.

Свободные пиксели не должны выделяться на глаз, то есть иметь характерный цвет, допустим черный как 0. Это значит, что для незанятых пикселей должно быть применено какое-то распределение значений, которое скроет от взгляда их особенность, которая может выдать информацию о используемых алгоритмах шифрования.

Прежде, чем проделать все эти действия, требуется получить данные картинки. Рассмотрим алгоритм на рисунке 2*.* Данный алгоритм позволяет выделить из текста файлы изображения для дальнейшего получения.

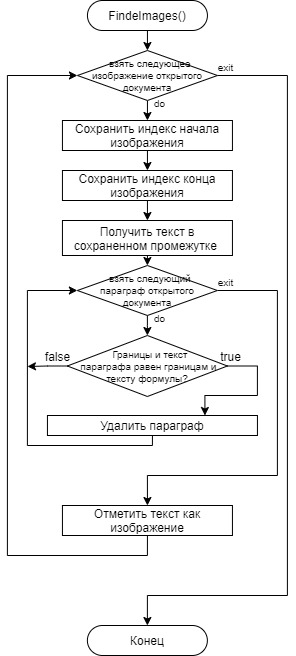
****

Рисунок 2.

Исследуя данную тему, я увидел проблему, с которой столкнулись все, кто пытался получить изображение из файла word программным методом напрямую. Сохранив индекс изображения в файле, и получив специальный тип данных Bitmap, представляющий из себя набор значений типа byte от 0 до 255, соответствующих одному цвету одного из трех накрадываемых фильтров, при попытке загрузить изображение обратно в файл, что нам также потребуется, возникала проблема с качеством изображения. Насколько я понял, проблема заключается в том, что функция получения и записи изображения в файл, работает на основе выделения картинки и записи выделенной картинки в буфер обмена. При такой операции, картинка имеет формат как была изображена в файле word, а так как в файле картинка обычно меньше по размерам, чем исходный формат, по причине не помещения на страницу исходного форматирования, то в буфер обмена попадает копия, которая немного хуже качеством.

Если бы это была одна операция или необратимое действие, то такой тип извлечения нам бы и подошел. Но при получении исходного файла и записи зашифрованного, а затем получения зашифрованного и записи расшифрованного, может быть настолько большая потеря в качестве, которая, в лучшем случае, будет видна пользователю и будет мешать ему в работе с изображением. В худшем случае изображение не сможет быть расшифровано, по причине других наборов данных, которые установлены после очередного сжатия изображения.

Для решения этой проблемы - были разработаны специальные алгоритмы. Первый из которых представлен на рисунке 0. Получение пути до системной папки с изображениями. Данная папка формируется самой программой и служит для одной цели: хранение временных данных. Такими временными данными и будут являться изображения в исходном форматировании без сжатия. Для начала, с помощью алгоритма на рисунке 3., программа получает путь до уже имеющейся или, впоследствии, создаваемой папки с фиксированным именем.

****

Рисунок 3.

Алгоритм выше использует функцию, для получения корневой папки. Алгоритм получения корневой папки представлен на рисунке 4*.* Суть алгоритма в том, чтобы взять все имеющиеся документы и найти папку, которая находится в меньшем числе папок выше. То есть, чем меньше папок в пути до данной папки, тем лучше. Путь, имеющий наименьшее число папок, будет усечен до самой последней папки в данном пути и возращен как результат. Суть усечения в том, что путь обычно указывает на файл, а нам требуется папка, в которой данный файл находится.

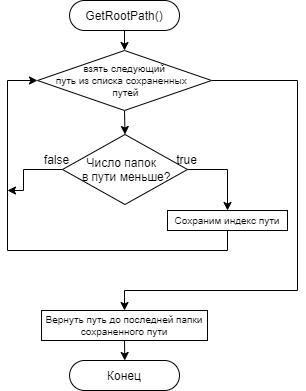
****

Рисунок 4.

Получив данные о том, где находится или должна находится системная папка, можно приступать к сохранению файла в специальном виде. Сохранение файла word в формате .html приведет к тому, что весь текст файла будет записан в файл - имя файла.html, а все изображения поместятся в папку указанной директории и будут в исходном форматировании без сжатия. Если системной папки не существует, то она создается, затем происходит сохранение по пути системной папки.

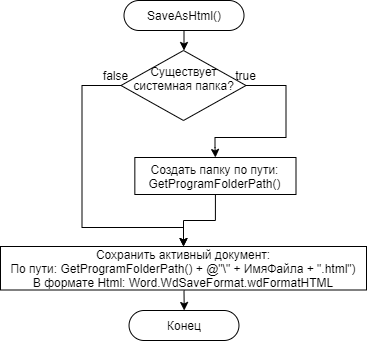
****

Рисунок 5.

После окончания работы с изображениями, требуется удалить системную папку с временными файлами. Сначала происходит проверка наличия такой папки, затем, если такая папка всё же имеется, происходит удаление данной папки со всеми включенными в нее файлами.

Стоит отметить, что создание и удаление данной папки, происходит в одну секунду, так как между ними происходит только один процесс – сохранения изображения в формате Bitmap. Это очень быстрая процедура, позволяющая не привязываться к конкретному файлу, а хранить только его данные. Отсутствие привязки к конкретному файлу, позволяет беспрепятственно удалить исходный рисунок после сохранения его данных в программе.

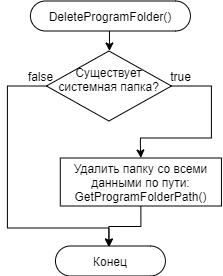
****

Рисунок 6.

Получение данных картинки происходит в алгоритме на рисунке 6*.* Здесь представлены все или большая часть из сохраняемых данных. Все эти данные важны для восстановления картинки в другом месте с помощью специальных функций. Одних пикселей недостаточно для формирования изображения. Программа word требует наличия формата изображения. Формат хранится в специальном классе, который также копируется для возможности редактирования или восстановления. Таким же образом сохраняется класс палитры. Это такой же специальный класс, требуемый для корректного восстановления. Сохранение пикселей происходит не как раньше в формате Bitmap, а в виде массива байтовых чисел. Это нужно потому, что класс BitmapImage, именно так он называется полностью, хранит доступ к исходному файлу и использует его до тех пор, пока объект данного класса не будет удалён. В нашей программе используется системная папка, которая удаляется после получения данных. Если какой-либо процесс использует файл из данной папки, то удаление не будет завершено, что может привести к дальнейшим уязвимостям от пользователя. В данном алгоритме мы сохраняем BitmapImage как массив байтов, что никак не отражается на качестве картинки, а только лишает нас ненужных привязанностей.

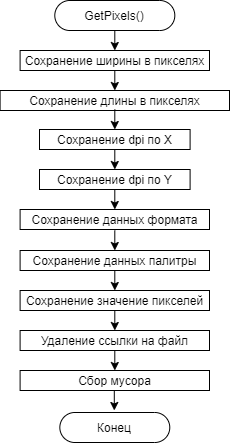


Рисунок 7.

Получение данных картинки происходит упрощенно по алгоритму на рисунке 7*.* Для каждой найденной картинки, применяется выше описанный метод сохранения данных. Если капнуть более подробно, то на самом деле, получение всех изображений может привести к ошибкам программы. Так как, например, формулы файла также сохраняются в виде изображений. Пользователь рассчитывает получить изображения, а получает формулы, которые в файле были распознаны как изображения.

Решением такой проблемы может быть сохранение всех объектов, которые могут быть распознаны как изображения, но в хронологическом порядке документа таковыми не являются. То есть, сохранить в каком порядке чередуются формулы файла с изображениями и получить из папки с изображениями файлы только с тем индексом, которые отмечены как изображения, а не что-то другое.

Проблема тут возникает другая, а именно слепой перебор из файла. В данном случае, мы не знаем точно о том, что тот ли файл мы получаем. Основываемся мы только на том, что мы сохранили файлы в хронологическом порядке, и программа сохранила объекты файла в том же порядке. Но если один из объектов пропадёт по какой - то причине, то весь порядок нарушится и это может не привести к ошибке, а также быть распознано на стадии необратимости. Данные проблемы будут протестированы в разделе тестирования.

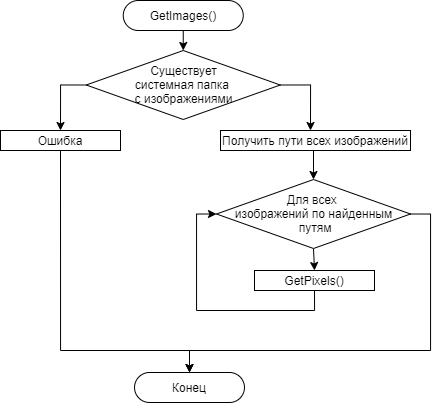


Рисунок 8.

# 

**Заключение**

**Тезариус**

**Список литературы**: