*Государственное образовательное учреждение высшего профессионального образования*



*«Московский Государственный Технический Университет имени Н.Э. Баумана»*

*(МГТУ им. Н.Э. Баумана)*

*ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»*

*КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»*

КУРСОВОЙ ПРОЕКТ

Android приложение с блочным шифрованием “Кузнечик”

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Бородин А.А.

(подпись, дата)

Разработчик проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кустов И.А.

(подпись, дата)

**Оглавление**

[**Цель**](#_jox971z2px5r) **4**

[**Основные определения**](#_z62yaw3hwpvq) **4**

[**Введение**](#_o8eaew6ydbmg) **5**

[**Требования к проекту**](#_4le00zi77am) **6**

[**План разработки**](#_ii24jok02vmb) **6**

[**Выбор технологий**](#_7q8neosbkwlb) **7**

[**Описание работы системы**](#_i132easf0pzd) **8**

[**Описание технических решений**](#_pl1bgrrxej29) **12**

[**Заключение**](#_nu0xenu2q3gn) **13**

[**Список использованных источников**](#_355r7dr8p9ru) **13**

# 

# **Цель**

Создание мобильного приложения на платформе Android, реализация блочного шифра «Кузнечик» (ГОСТ Р 34.12-2015), скрытие зашифрованной информации в изображение посредством цифровой стеганографии.

# **Основные определения**

* Блочный шифр «Кузнечик» - симметричный алгоритм блочного шифрования с размером блока 128 битов и длиной ключа 256 битов.
* Стеганография - способ передачи и хранения информации с учетом сохранения в тайне самого факта наличия информации
* Контейнер - любая информация, предназначенная для скрытия сообщения
* Стеганоконтейнер - контейнер, содержащий встроенную информацию
* JDK — набор для разработки на языке kotlin
* Хэширование - это процесс преобразования массива входных данных произвольной длины в битовую строку фиксированной длины соласно заданному алгоритму.

# **Введение**

В последнее время даже самые “примитивные” интернет-пользователи стали задумываться о защите той информации, что они хотят донести до своего собеседника или хранить у себя безопасно.

Данный проект представляет собой мобильное приложение, которое позволит скрывать информацию без страха, что посторонний человек сможет узнать ваши секреты. Текстовая информация будет надежно стрыта шифром “Кузнечик”, который в свою очередь побитово будет занесен методом LSB в изображение, которое предоставит пользователь. Приложение позволит без особого труда зашифровать текстовую информацию в стегоконтейнер. Не требует подключения к интернету, что сокращает энергопотребление и дает возможность пользоваться приложением в любое время и в любом месте.

Приложение не получит большой популярности в деловых кругах, но заинтересует молодежь, которая не хочет, чтобы родители контролировали каждый их шаг. Картинки и фотографии не вызовут никакого подозрения среди кучи сообщений о поверхностных темах, в то время как вся необходимая информация будет доходить до адресата без рисков.

# **Требования к проекту**

1. Мобильное приложение на платформе Android
2. Шифрование Кузнечиком и занесение данных в контейнер
3. Дешифрование стеганоконтейнера и получение информации
4. Возможность изменения/удаления ключа для шифрования
5. Возможность сохранения стеганоконтейнера в формате PNG
6. Возможность копирования текстового результата шифрования/дешифрования в буффер

# **План разработки**

1. Проектирование архитектуры проекта - до 09.10.17
2. Разбор шифра и реализация на kotlin - до 16.10.17
3. Ознакомление со стеганографией и реализация - до 30.10.07
4. Реализация мобильного приложения - до 23.10.17
5. Демонстрация прототипа - 31.10.17
6. Исправление недочетов - 07.11.17
7. Демонстрация готового приложения - 2.12.17

# **Выбор технологий**

**Выбор языка программирования**

Мобильные приложения на Android разрабатываются на java и kotlin.

Преимущества kotlin перед java.

* Язык более прост для изучения
* Язык null-безопасен
* Легко читаемый синтаксис
* Специальные классы для хранения данных
* “Умное” приведение к типу

Так же, kotlin может использовать все библиотеки и фреймворки Java.

Исходя из вышеперечисленных преимуществ, kotlin более удобен. Приложение будет написано на нем.

**Выбор из библиотек**

При написании приложения будут использованы:

* библиотеки JDK 8.0
* библиотека хеширования MessageDiggest для получения мастер ключа по ключевому слову. Библиотека имеет множество возможных генераций ключей, таких как SHA и MD5.
* библиотека Design:25.3.1.

# **Описание работы системы**

**Общая информация о системе**

Система представляет собой мобильное приложение, написанное на алгоритмическом языке kotlin. В основном окне приложения находится фрейм для фрагментов шифрования и дешифрования, кнопки добавления ключа, которая вызывает диалоговое окно, и удаления ключа. Для работы программы необходимо ввести ключ. В случае, если он не введен, кнопки фрагментов, которые осуществляют шифрование/дешифрование, вызывают диалоговое окно для ввода ключа. Для зашифрования информации следует перейти в соответствующий фрагмент приложения, в котором пользователь вводит текст, который требует шифрования и скрытия, и изображение, в которое требуется “спрятать” информацию. Текст передается на обработку в класс шифрования GrassHopper, где разбивается на блоки по 16 символов и шифруется.

Затем результат передается в класс Stego. Сюда же попадает изображение. Исходное изображение пользователь может сохранить на свой телефон в формате PNG, который устойчив к сжатию, что позволяет нам извлекать информацию при желании с крайне малым риском на провал.

В случае, если на вход подан лишь текст, приложение выводит зашифрованный текст в формате Base64. Если текст не введен, приложение выводит сообщение о том, что пользователь должен ввести текст.

Расшифровывание изображение происходит в другом фрагменте приложения, где на вход идет изображение, а на выход пользователь получает текстовую информацию. Если в изображении нет скрытого сообщения, на экран выводится соответствующее сообщение.

Для расшифровывания текста также создан отдельный фрагмент. На вход приложение получает зашифрованный текст в формате Base64, на выход пользователь получает текстовую информацию.

В диалоговом окне можно выбрать, запоминать ключ или нет. если установить “Запомнить ключ”, то при закрытии приложения и открытии заново ключ останется введен. В ином случае, придется вводить ключ снова.

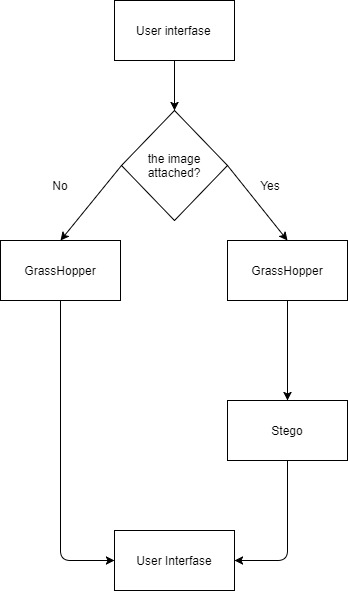


рисунок 1. Общая информация о системе

**Класс GrassHopper**

Класс шифрует и расшифровывает текстовую информацию, которую в него записывают с помощью функции Parsing, которая разбивает текст на блоки по 16 символов. В качестве переменных в классе хранится список из 10 раундовых ключей и текст, который разбит по 16 блоков. В классе реализованы следующие функции: key\_create (создает 10 раундовых кючей), encrypt (шифрование текста), decrypt (расшифрование текста), real (соединение всех блоков текста в одну строку и возвращение в типе string) Кузнечик реализован через следующие итерации:

* 128-битное сложение по XOR
* Линейное преобразование - обычная перестановка
* Нелинейное преобразование - применение к каждому 8-битному подвектору 128-битного входного вектора фиксированной подстановки

Пример шифрования блока из 16 символов:

Входные данные: 6B|6C|6D|6E|77|74|65|65|72|74|77|65|72|79|74|65

Результат шифрования: 46|DC|D7|8D|B5|87|97|10|99|6F|B4|AE|63|2C|3B|F0

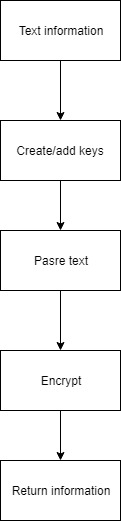


рисунок 2. Алгоритм

шифрования

**Класс Stego**

Хранит битовое представление изображения   
 Имеет 2 функции: скрытие информации в изображение и получение информации из текста.

Использует алгоритм LSB, который заключается в изменении последних битов синего пикселя. Изменяется каждый 4 пиксель, а именно 2 последних бита синего цвета. В первый столбец изображение записывает количество символов зашифрованного текста. Остальной столбец обнуляет для проверки на уникальность при расшифровке.

При извлечении текста. Считывает количество символов. Если оно не делится целочественно на 16 или оставшиеся пиксели не обнулены, в изображении ничего не скрыто. В ином случае получает текст и возвращает его.

Пример работы::

Информация для скрытия: 1011

Первоначальные биты синего цвета изображения:

10011101 11100101 10001111 11100011 11000111

Биты синего цвета после преобразования:

100111**10** 11100101 10001111 11100011 110001**11**

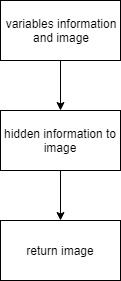


рисунок 3. Алгоритм

занесения текста в контейнер

# **Описание технических решений**

В ходе написания курсового проекта возникали следующие проблемы:

1. **kotlin.** Язык программирования разрабатывался с 2010 года, однако релиз 1.0 вышел 15 февраля 2016 года. По этой причине крайне мало информации об этом языке, наличие руководств по использованию крайне мало, отсутствие полностью доработанного документатора и ряд других проблем (нежелание работать с junit, существует kotlintest, но по нему практически нет примеров работы). Так же, работа в Android Studio в основном описана на Java, поэтому в ходе написание проекта, были получены базовые знания этого языка
2. **Проблемы с NDK.** Так как на java/kotlin не было найдено готового шифра “Кузнечик”, а настройка NDK требовало 80-100 часов реального времени для ознакомления с информацией, пришлось реализовывать шифр самостоятельно.
3. **Вывод текста.** Так как приложение давало возможность как скрывать текст в изображение, так и получать зашифрованный текст напрямую, возникала проблема с выводом текста. А именно с тем, что имелась вероятность получения символа пустой строки в середине текста, из-за чего текст выводился не полностью и отсутствовала возможность расшифровывания. Было решено использовать библиотеку Base64, что позволило избегать проблем с выводом и копированием текста.
4. **Сохранение изображений в галерею.** При сохранении возникала следующая проблема: изображение попадало в хранилище, однако не отображалось в галерее, так как та в свою очередь не обновлялась. Около недели ушло на решение этой проблемы, так как ее причина не была известна
5. **Дизайн.** 3 раза переписывался дизайн. Изначально использовались различные окна приложения. В последнем изменение было решено использовать библиотеку Google Design, с помощью которой остался один главный экран, диалоговое окно и 3 фрагмента для работы с информацией.

# **Заключение**

Было реализовано мобильное приложение для шифрования текстовой информации. Таким образом, приложение предоставляет возможность хранить текст в изображениях, обмениваться ими с другими пользователями, либо шифровать непосредственно в текст. В следующих версиях может быть реализовано:

* Выбор языка платформы и ввода данных
* Подключение VkApi
* Переход на стеганографирование в формат JPEG

# **Список использованных источников**

1. В ГОСТе сидел “Кузнечик”/Хабрахабр: [Электронный ресурс]. 2015. URL: https://habrahabr.ru/post/266359/ (Дата обращения: 17.12.2017).
2. ГОСТ Р 34.12-2015 Криптографическая защита информации: [Электронный ресурс]. 2016. URL: <https://studfiles.net/preview/6173591/> (Дата обращения: 17.12.2017).
3. MessageDigest: [Электронный ресурс]. URL: [https://developer.android.com/reference/java/security/MessageDig](https://developer.android.com/reference/java/security/MessageDigest.html) (Дата обращения: 17.12.2017).
4. Обучение разработке мобильных приложение и игр для ANDROID: [Электронный ресурс]. URL: <http://www.fandroid.info/> (Дата обращения: 17.12.2017).
5. Hello, world! | Try Kotlin: [Электронный ресурс]. 2015. URL: <https://try.kotlinlang.org/#/Kotlin%20Koans/Introduction/Hello,%20world!/Task.kt> (Дата обращения: 17.12.2017).
6. Stack Overflow на русском: [Электронный ресурс]. 2015. URL: <https://ru.stackoverflow.com> (Дата обращения: 17.12.2017).
7. Реализация блочного шифра “Кузнечик” с режимом CFB на C++/Хабрахабр: [Электронный ресурс]. 2015. URL: <https://habrahabr.ru/post/313932/> (Дата обращения: 17.12.2017).