Государственное бюджетное общеобразовательное учреждение г. Москвы
«Школа № 2086»
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА-Российский технологический университет»
Технопарк «Альтаир»

СИМВОЛЬНЫЙ ШИФРАТОР

Автор:
11 «А», учащийся ГБОУ «Школа № 2086», Залыгин Вячеслав Константинович

Научный руководитель:
РТУ МИРЭА, преподаватель, Кунин Никита Тимофеевич

Москва, 2021 год

Оглавление

Введение

В связи с активным развитием информационных технологий, в эру 4 промышленной революции, когда человек активно использует различные мессенджеры и социальные сети для передачи личной информации, обыватель не всегда может контролировать степень защищенности этой информации при её передаче через интернет. Поэтому защита конфиденциальности данных становится актуальна, как никогда ранее.

С развитием технологий несанкционированного доступа к информации встаёт вопрос о том, как пользователь может самостоятельно защитить свои данные. В случае неуверенности в защите канала передачи данных, он может использовать дополнительные средства сокрытия, которые смогут гарантировать, что сокрытая информация при доставке сообщения будет защищена от посторонних лиц.

Проанализировав ситуацию на рынке (в частности Google play), я выделил некоторые проблемы, которые может встретить пользователь при использовании существующих средств сокрытия:

1. Отсутствие комплексной защиты. Некоторые приложения реализуют только либо стеганографический функционал, либо криптографический, что неудобно и не обеспечивает должный уровень безопасности.

2. Сложный интерфейс. Разработчики приложений создают не интуитивный, перегруженный интерфейс, в котором обывателю трудно разобраться.

3. Использование только изображений в качестве контейнера. Такое решение перенимает проблемы пересылки изображений: высокий вес сообщения (и, как следствие, требование быстрого соединения), сложный интерфейс обращения и неестественность при активной переписке.

Таким образом, я решил разработать своё программное решение, лишённое поставленных проблем.

Цели и задачи

Цель: разработка android-приложения для сокрытия секретного сообщения внутри текстового контейнера на основе вставки малозаметных спецсимволов и замены букв.

Задачами проекта являются:

1. Разработка алгоритмов вставки на основе добавочных символов и замены символов, схожими по изображению, но разными с точки зрения компьютера. В разработку входили: изучение возможностей замены букв и вставки добавочных символов для кодирования какой-либо информации, создание принципов и схем, на основе которых строились алгоритмы, разделение задачи на более маленькие и независимые единицы-модули.

2. Реализация прототипа на языке Python и тестирование алгоритмов на предмет работоспособности. Язык Python был выбран как язык с наиболее

простым синтаксисом, дабы ускорить разработку прототипа, а также для демонстрации ранних версий алгоритма. Один за другим были написаны, протестированы модули алгоритма кодирования.

3. Реализация алгоритма на Java. После завершения разработки, алгоритм переписывался на язык Java. Такой переход объясняется тем, что Java является нативным языком операционной системы Android, под которую разрабатывалось приложение. Некоторые части модулей были разработаны заново в связи с разницей в работе некоторых функций в языках Python и Java.

4. Создание android-приложения. Для приложения был написан интерфейс, легко понимаемый пользователями. Проведены многочисленные тесты и исправления для повышения стабильности работы приложения, ликвидации ошибок при различных вводных данных и режимах работы.

Методика выполнения работы

Список использовавшихся в работе ресурсов

Таблица 1. Списков использовавшихся в работе ресурсов

Название	**Назначение**	**Кем предоставлялся**	**Условия предоставления**
PyCharm community edition	Работа с прототипом на Python	Компания JetBrains	Community лицензия
IntelliJ IDEA community edition	Работа с алгоритмами на Java	Компания JetBrains	Community лицензия
Android studio	Создание android-приложения	Компания Google	Community лицензия
Телефон под управление os	Тестирование приложения	Личный телефон автора проекта	Безвозмездно

Andoird			
Компьютер под управление os Windows	Разработка программ	Личный компьютер автора проекта	Безвозмездно

Описание интерфейса

Интерфейс android-приложения (Рисунок 1. Интерфейс приложения) состоит из трёх зон: выбор профиля, редактирование профилей работы, ввод данных. В зоне работы с профилями (верх экрана) пользовать может выбрать нужный ему профиль работы, вызвать окно редактирования профиля или создать новой профиль. В нижней части экрана находится зона ввода данных, непосредственно используемых при сокрытии и излечении секрета. Пользователю предлагается выбрать тип операции (сокрытие или извлечение), далее ввести текст-контейнер, а затем секрет. После нажатия на кнопку «Выполнить», снизу появится заполненный контейнер, готовый к отправке. Далее пользователь может либо скопировать секрет в буфер обмена, либо использовать встроенное меню Android для пересылки текста в интересующий мессенджер.

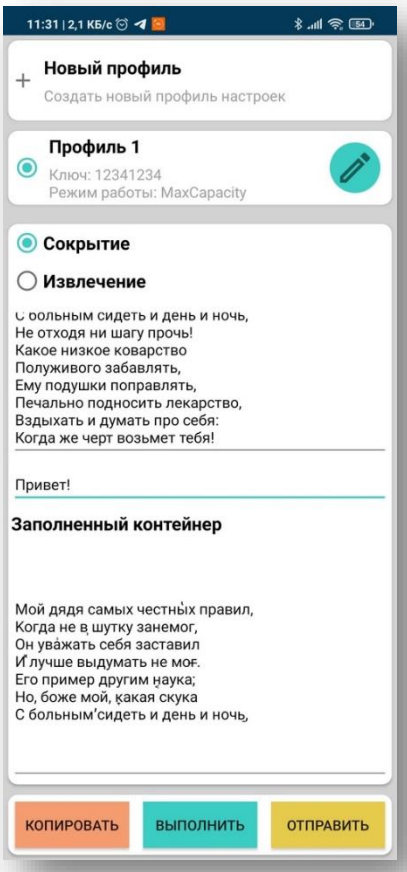

Рисунок 1. Интерфейс приложения

Описание алгоритмов кодирования

Алгоритм кодирования разделён на несколько модулей: DesEncoder, ReplaceEncoder, InsertEncoder. Каждый из модулей независим от других и содержит в себе 2 публичных метода – метод шифровки и метод дешифровки.

Модуль DesEncoder отвечает за шифрование и перевод входного сообщения в вид, требуемый для алгоритмов вставки. Он представляет собой интерфейс для работы с алгоритмом des из встроенной библиотеки Java. Этот модуль является криптографической частью проекта и обеспечивает защищённость данных даже после их обнаружения и извлечения из контейнера. В самом модуле представлены методы шифровки/дешифровки и вспомогательные методы. На вход при шифровании передаётся секретное сообщение, введённое пользователем, и цифровой ключ фиксированной длины (8 цифр), возвращается последовательность чисел, готовая к сокрытию в контейнере.

Модули InsertEncoder и ReplaceEncoder занимаются одним и тем же – вставкой последовательности в контейнере, однако делают это разными способами. InsertEncoder кодирует последовательность путём вставки в контейнер различных малозаметных добавочных символов (комбинируемых символов), а ReplaceEncoder использует существование символов, схожих по рисовке, но имеющих разные порядковые номера в таблице (Например, русская «а» и английская «а»).

Для кодирования вставкой на вход метода encoding нужно подать текст-контейнер и секретное сообщение, представленное последовательностью чисел. В самом модуле хранится массив с 30 различными спецсимволами. Для вставки в контейнер последовательность цифр (секретное сообщение) переводится в систему счисления с основанием 30. Далее каждый элемент последовательности заменяется соответствующим по номеру спецсимволом и вставляется в контейнер. Причём для этого контейнер делится на кластеры так, чтобы в одном кластере содержался один спецсимвол. Такой подход позволяет равномерно распределять спецсимволы по всему тексту. Декодирование работает обратно кодированию: из текста вылавливаются спецсимволы и формируются в последовательность, которая потом переводится в 10-чную систему счисления.

Для кодирования алгоритмом замены в соответствующем модуле содержится массив с парами взаимозаменяемых символов. В метод кодирования надо подать текст-контейнер, сообщение-секрет. Теперь программа переводит сообщение-секрет в двоичную систему счисления. Далее она проходит по всему контейнеру и, найдя символ, который можно заменить, заменяет его либо на русский вариант символа, либо на английский в соответствии с текущим кодируемым элементом из двоичной последовательности (0 – русская буква, 1 – английская). Дешифровка обратна шифровке. В контейнере ищутся символы, и составляется бинарная последовательность, которая далее переводится в 10-чную систему счисления, что и является закодированным сообщением.

В целом, сообщение-секрет и контейнер подвергаются цепочке преобразований, состоящей из различных модулей. Для шифрования используется цепочка DesEncoder -> Insert/ReplaceEncoder, а для дешифрования Insert/ReplaceEncoder -> DesEncoder. Для предоставления выбора пользователям были добавлены режимы работы. Они создают различные вариации использования алгоритмов кодирования. Так, режим 1 (стандартный) использует алгоритм замены для кодирования контрольной суммы исходной последовательности, а алгоритм вставки для кодирования самого сообщения. Таким образом, в случае несоответствия контрольной суммы с суммой декодированной последовательности, будет сигнализироваться о изменении текста-контейнера. Режим 2 (максимальная вместимость) кодирует максимально возможное количество символов секрета через алгоритм замены, а оставшуюся часть алгоритмом вставки, тем самым повышая вместительность контейнера и делая кодирование менее заметным. Режимы 3 и 4 используют для кодирования только либо алгоритм замены, либо алгоритм вставки.

В моём приложении предусмотрена система профилей, в каждом профиле хранится ключ и режим работы. Профили пользователь создаёт самостоятельно. Такое решение нужно для удобства пользования приложением.

Например, при активном применении приложения возникает неудобство постоянного заполнения полей ключа и режима работы. Профиль, в свою очередь, можно создать один раз и выбирать нужные параметры одним нажатием.

Минимальная длина контейнера в зависимости от длины сообщения

Таблица 2. Отношение длины секрета к длине контейнера

Категория	Режим работы		Значение	Зависимость
	Операция	Режим		
Минимальная длина контейнера	Вставка	Standard	3,1	Линейная
	Вставка	MaxCapacity	2,5	
	Вставка	OnlyInsert	3,0	
	Вставка	OnlyReplace	13,0	

В данной таблице (Таблица 2. Отношение длины секрета к длине контейнера) приведены значения, соответствующие отношению минимальной длины контейнера и сообщения, кодированного в нем. График данного отношения линеен для всех режимов, то есть эти значения верны при любом размере контейнера и сообщения. Из приведённых данных можно сделать вывод, что текст-контейнер становится наиболее вместительным при работе режима MaxCapacity, так как этот режим первоначально разрабатывался как самый вместительный. Самыми худшими показателями по вместительности обладает режим OnlyReplace, но при этом он имеет неоспоримое преимущество в виде абсолютно незаметной вставки секрета в контейнер.

Коэффициент искажения контейнера

Коэффициентом искажения контейнера является отношение количества добавочных символов к длине всего контейнера до искажения ($k = n/N$, где k – коэффициент искажения, n – количество добавочных символов, N – суммарное количество символов в контейнере до искажения). Данный коэффициент целесообразно рассчитывать только при применении алгоритма вставки, так

как алгоритм замены если и заменяет символ, то делает это на почти или полностью одинаковый по отображению символ, что визуально не искажает контейнер.

Чем ниже коэффициент искажения контейнера, тем реже встречаются добавочные символы в тексте. Путём экспериментов было установлено, что рекомендуемый минимальный коэффициент искажения равняется 0.06 и ниже, при котором на один добавочный символ приходилось 16 обычных.

При средней длине секрета в 60-100 символов (если считать, что секретом является обычное сообщение среднестатистической длины) длина контейнера должна составлять около 900 символов при режиме максимальной вместимости.

Асимптотическая оценка времени работы алгоритма

Таблица 3. Асимптотическая оценка времени работы алгоритма

Категория	**Режим работы**		**Зависимость**	
	Операция	**Режим**	**От длины контейнера**	**От длины секрета**
Асимптотическая оценка времени выполнения	Вставка	Standard	O(N^2)	O(N)
	Вставка	MaxCapacity	O(N^2)	O(N)
	Вставка	OnlyInsert	O(N^2)	O(N)
	Вставка	OnlyReplace	O(N^2)	O(N)
	Извлечение	Standard	O(N^2)	O(N)
	Извлечение	MaxCapacity	O(N^2)	O(N)
	Извлечение	OnlyInsert	O(N^2)	O(N)
	Извлечение	OnlyReplace	O(N^2)	O(N)

В таблице (Таблица 3. Асимптотическая оценка времени работы алгоритма) показана асимптотическая оценка времени для разных режимов работы. Сама асимптотическая сложность разделена на зависимости от размера контейнера и от размера секрета. Такое разделение является следствием того,

что каждый из этих параметров связан между собой лишь отношением размеров и самостоятельно влияет на оценку, остальные же используемые параметры являются константами и влияют только на конечное время работы программы. В целом, можно утверждать, что время работы в большей степени зависит от размера контейнера, это следует из того, что алгоритм вынужден по несколько раз проходится по каждому элементу в контейнере (в следствие особенностей работы используемых методов Java), что обуславливает верхнюю планку O(N^2). Также на итоговую оценку влияет длина секрета, но в меньшей степени, т.к. она сама по себе должна быть меньше длинны контейнера, с секретом выполняется относительно меньшее количество операций, при вставке и извлечении основные манипуляции производятся со всем контейнером, а не с секретом. Тем не менее с секретом производятся некоторые операции, объём которых зависит от размера секрета, поэтому верхней асимптотической сложностью будет O(N). Для нахождения асимптотической сложности для каждого режима работы был построен график, где отображено время работы и текущий размер контейнера или секрета. Оценка производилась по степени роста графиков.

Важно понимать, что в первую очередь время работы зависит от размера входных данных, которые, в связи с спецификой использования приложения, как правило, не имеют больших объёмов, что позволяет алгоритму выполняться за несоизмеримо маленькое время, не доставляя тем самым пользователю каких-либо неудобств в использовании. Например, при работе с контейнером длиной около 1000 символов, время работы на большинстве устройств не превышает и 100 миллисекунд.

Описание архитектуры приложения

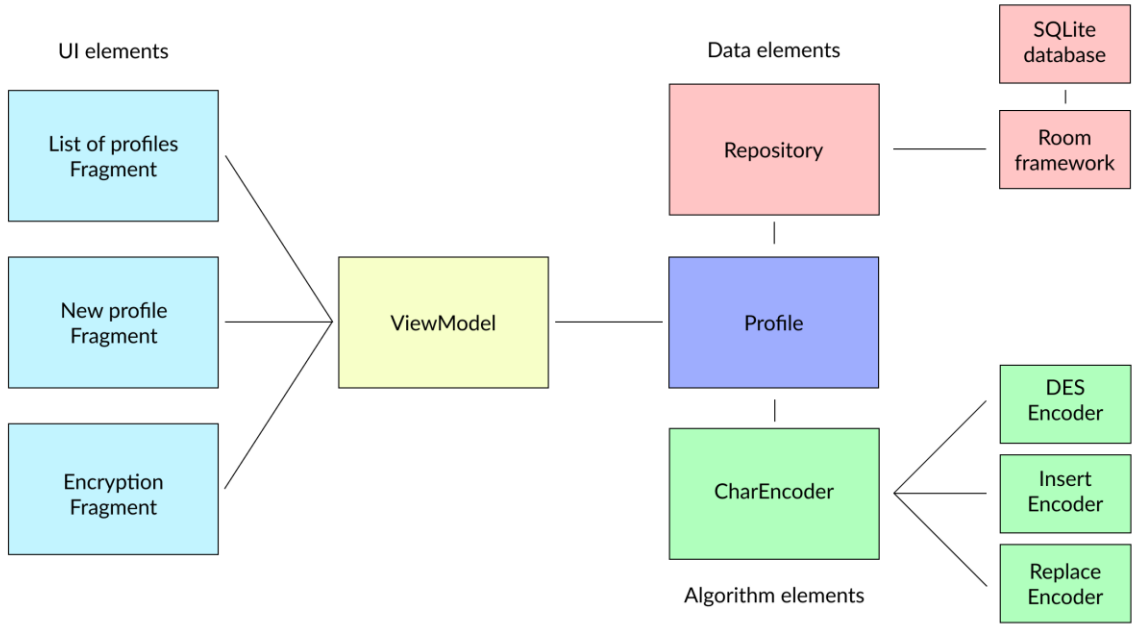

Рисунок 2. Архитектура приложения

Архитектура приложения (Рисунок 2. Архитектура приложения) соответствует шаблону проектирования MVVM. Для модели (класс Profile) работа с источником данных осуществляется за счёт класса Repository и одноимённого паттерна проектирования. Таким образом, существует возможность создавать несколько мест хранения профилей и взаимозаменять источники, не меняя остальной логики. В данный момент для хранения профилей используется локальная база данных SQLite с фреймворком Room, который нужен для облегчения устройства общения репозитория с базой данных. Точка доступа к алгоритмам сокрытия тоже находится в модели (класс Profile) через обращение к классу CharEncoder, который является фасадом к 3 используемым алгоритмам и реализует внутри себя поведение при разных режимах работы. Объект класса Profile хранит внутри себя ключ шифрования и режим работы, реализуя собой программное представление профиля работы, в нём содержится статичный набор методов для получения списка профилей и некоторых действий с ними, а также метод сокрытия, куда передаётся текст и секрет. Класс ViewModel хранит LiveData объект, представляющий из себя список профилей, также она предоставляется данные, полученные из Profile, ui-

компонентам. Интерфейс приложения представлен 3 фрагментами. List of profiles Fragment имеет в своей разметке RecyclerView для отображения списка профилей и кнопку для создания нового. Edit profile Fragment представляет собой диалоговый фрагмент, вызываемый всякий раз, когда пользователю нужно совершить какие-то операции над профилями. Encryption Fragment отображает форму для операций сокрытия и извлечения.

Потребители

Мной была определена основная группа пользователей приложения — это активные пользователи соцсетей, что энергично поддерживают анонимность в интернете, а также хорошо заботятся о конфиденциальности своих данных. Это молодые люди от 16 до 30 лет, активно ведущие переписки.

Также существует ещё один сегмент потребителей – специалисты, нуждающейся в быстрой отправке зашифрованных данных по незащищённым каналам. Таким людям бывает нужно своевременно передать конфиденциальные данные, не тратя время на создание закрытого канала.

Бизнес-модель

Рисунок 3. Бизнес-модель

Результаты и перспективы

В результате было разработано приложение для устройств под управлением операционной системы android, которое обладает простым и понятным интерфейсом, способно осуществлять вставку и извлечение секретного текстового сообщения из текстового контейнера с использованием различных режимов кодирования и дополнительного криптографического шифрования в виде алгоритма DES.

В дальнейшем планируется осуществить следующие задачи: оптимизация алгоритмов вставки, добавление новых алгоритмов дополнительного шифрования (AES, RSA и т.д.), углубление интеграции приложения для увеличения удобства пользования, а также разработка корпоративного клиента для компаний, чья проблема заключается в утечке данных, корпоративных тайн из-за ненадёжных каналов связи.

Выводы

Мобильное приложение CharS для смартфонов под управлением операционной системы Android реализует стеганографические методы сокрытия секретного сообщения в текст-контейнере. В программе есть 4 режима кодирования с различными особенностями, а также есть встроенное шифрование секрета с помощью алгоритма DES. Приложение поможет просто и быстро зашифровать любой текстовый секрет в любом другом тексте и переслать его в интересующее Вас место.

Ссылки

1. Код клиента: https://github.com/VyacheslavZalygin/CharSApplication
2. Код сервера: https://github.com/VyacheslavZalygin/CharSBackend
3. Демонстрация работы: https://disk.yandex.ru/i/xgeUYRYHkJ6NMw
4. Демонстрация системы профилей: https://disk.yandex.ru/i/PmIn1fyRUaad9Q

Список используемой литературы

1. Python 3.9.1 documentation – URL: https://docs.python.org/3/index.html (дата обращения: 22.12.2020) – Текст: электронный.
2. Документация по Java – URL: https://docs.oracle.com/en/java/ (дата обращения: 01.02.2021) – Текст: электронный.
3. Горошко Е.И. Землякова Е.А.«Полиформатный мессенджер как жанр 2.0 (на примере мессенджера мгновенных сообщений Telegram)» - Жанры речи, 2017. №1(15). С. 92-100. – Текст: электронный // https://cyberleninka.ru/ - URL: https://cyberleninka.ru/article/n/poliformatnyy-messendzher-kak-zhanr-2-0-na-primere-messendzhera-mgnovennyh-soobscheniy-telegram/viewer (дата обращения: 15.12.2020).
4. Документация по разработке для Android – URL: https://developer.android.com/ (дата обращения: 29.01.2021) – Текст: электронный.