

**МУНИЦИПАЛЬНОЕ ОБРАЗОВАНИЕ**

**ГОРОДСКОЙ ОКРУГ «ГОРОД НИЖНИЙ НОВГОРОД»**

**МУНИЦИПАЛЬНОЕ АВТОНОМНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ШКОЛА № 74 С УГЛУБЛЕННЫМ ИЗУЧЕНИЕМ ОТДЕЛЬНЫХ ПРЕДМЕТОВ»**

Научное общество учащихся «Эврика»

**Разработка приложения «Cryptographer — Шифровальщик» для смартфона под управлением ОС Android**

Выполнил: Малышев Тимур

обучающийся 11а класса

Научный руководитель:

Родина Е.А.,

учитель информатики

Н.Новгород

2021

Оглавление

[Введение 3](#_gjdgxs)

[1.](#_30j0zll) Основы криптографической теории 5

[2.](#_1fob9te) Криптоалгоритмы 10

[2.1.](#_3znysh7) Алгоритм DES 10

[2.3.](#_2et92p0) Алгоритм BlowFish 12

[3.](#_tyjcwt) Разработка приложения «Cryptographer — Шифровальщик» 14

[3.1.](#_3dy6vkm) Алгоритм «eXT»: 15

[3.2.](#_1t3h5sf) Графический пользовательский интерфейс (GUI): 17

[3.3.](#_4d34og8) Экран настроек: 18

[3.4.](#_2s8eyo1) Элемент Toast. 18

[3.5.](#_17dp8vu) Выход из приложения. 19

[3.6.](#_3rdcrjn) Смена цвета акцента. 19

[3.7.](#_26in1rg) Обмен ключами шифрования: 19

[3.8.](#_lnxbz9) Оптимизация приложения 20

[Заключение 23](#_35nkun2)

[Список литературы](#_1ksv4uv) [24](#_44sinio)

1. **Введение**

Сегодня все больше людей общаются не лично, а через компьютерную сеть. Честность и безопасность многих протоколов человеческого общения основаны на личном присутствии. И если человек при личной встрече может определить уровень доверия, то компьютеру нужны определенные правила.

Даже в древние времена сохранности информации придавалось большое значение. В наши дни ситуация не изменилась: сегодня ничто не ценится так дорого, как информация, записанная в цифровой форме и передаваемая по сетям. И ничто другое невозможно так легко потерять, как эту информацию.

Для защиты информации применяется ряд мер. К ним относятся:

* организация средств защиты;
* аппаратные средства защиты;
* криптографические средства защиты.

Для обеспечения сохранности информации существуют следующие возможности:

* создание надежного, недоступного для других людей канала связи между абонентами;
* использование общедоступного канала связи но скрыть сам факт передачи информации
* использование общедоступного канала связи, но передавать информацию в таком преобразованном виде, чтобы восстановить ее мог только адресат.

Изучением вопросов шифрования данных занимается наука — криптология, включающая криптографию и криптоанализ.

Криптография изучает методы и алгоритмы шифрования данных (правила построения и использования шифров), направленные на то, чтобы сделать эти данные бесполезными для противника. Методы криптографии также используются для подтверждения подлинности источника данных и контроля целостности данных.

Криптоанализ — это наука о раскрытии исходного текста зашифрованного сообщения без доступа к ключу. Процесс шифрования информации с точки зрения программиста можно представить следующим образом: Открытый текст -> алгоритм шифрования (ключ 1) -> шифрованное сообщение.

А процесс расшифровывания: Шифрованное сообщение -> алгоритм дешифрирования (ключ 2) -> открытый текст.

Целью данной работы является разработка приложения с графическим интерфейсом для смартфона под ОС Android для анонимного общения в сети.

Для достижения поставленной цели предполагается решение ряда задач:

1. Изучение алгоритмов шифрования.
2. Анализ существующих приложений.
3. Проектирование структуры приложения.
4. Программная реализация.
5. Тестирование полученного продукта и устранение недостатков.
6. Организация поддержки программного продукта.

В настоящее время существует большое количество криптоалгоритмов. В ходе работы над проектом автором были изучены следующие алгоритмы: DES, AES, BlowFish. В процессе работы автор разработал свой алгоритм, основанный на алгоритме Цезаря, который получил название eXT.

1. **Основы криптографической теории**

Информация, которая может быть прочитана, осмыслена и понята без какихлибо специальных мер, называется открытым текстом (plaintext, clear text). Метод искажения открытого текста таким образом, чтобы скрыть его суть, называется зашифрованием – (encryption или enciphering). Зашифрование открытого текста приводит к его превращению в непонятную абракадабру, именуемую шифртекстом (ciphertext). Шифрование позволяет скрыть информацию от тех, для кого она не предназначается, несмотря на то, что они могут видеть сам шифртекст. Противоположный процесс по обращению шифртекста в его исходный вид называется расшифрованием (decryption или deciphering).

Рисунок 1 иллюстрирует это.



* 1. **Что такое криптография**

Криптография – это наука об использовании математики для зашифрования и расшифрования данных. Криптография позволяет хранить важную информацию или передавать её по ненадёжным каналам связи (таким как Интернет) так, что она не может быть прочитана никем, кроме легитимного получателя.

В то время как криптография – это наука о защите данных, криптоанализ – это наука об анализировании и взломе зашифрованной связи. Классический криптоанализ представляет собой смесь аналитики, математических и статистических расчётов, а также спокойствия, решительности и удачи. Криптоаналитиков также называют взломщиками. Криптология объединяет криптографию и криптоанализ.

* 1. **Стойкая криптография**

Криптография может быть стойкой, а может быть слабой. Криптографическая стойкость измеряется тем, сколько понадобится времени и ресурсов, чтобы из шифртекста восстановить исходный открытый текст. Результатом стойкой криптографии является шифртекст, который исключительно сложно взломать без обладания определёнными инструментами по дешифрованию. Но насколько сложно? Используя весь вычислительный потенциал современной цивилизации – даже миллиард компьютеров, выполняющих миллиард операций в секунду – невозможно дешифровать результат стойкой криптографии до конца существования Вселенной.

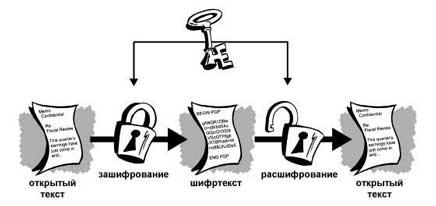
* 1. **Как действует криптография**

Криптографический алгоритм , или шифр , – это математическая формула, описывающая процессы зашифрования и расшифрования. Чтобы зашифровать открытый текст, криптоалгоритм работает в сочетании с ключом – словом, числом или фразой. Одно и то же сообщение одним алгоритмом, но разными ключами будет преобразовываться в разный шифртекст. Защищённость шифртекста целиком зависит от двух вещей: стойкости криптоалгоритма и секретности ключа.

Криптоалгоритм плюс всевозможные ключи и протоколы, приводящие их в действие, составляют криптосистему .

* 1. **Традиционная криптография**

В традиционной криптографии, также называемой шифрованием тайным, или симметричным, ключом, один и тот же ключ используется как для зашифрования, так и для расшифрования данных. Data Encryption Standart (DES) – пример симметричного алгоритма, широко применявшегося на Западе с 70х годов в банковской и коммерческой сферах. В настоящее время его сменяет Advanced Encryption Standard (AES). Рисунок 2 иллюстрирует процесс симметричного шифрования.



* 1. **Шифр Цезаря**

Крайне простой пример симметричного шифрования – это подстановочный шифр. Подстановочный шифр заменяет каждую часть информации другой информацией. Чаще всего это достигается смещением букв алфавита. Алгоритм состоит в том, чтобы сдвинуть алфавит, а ключ – это число букв, на которое произведено смещение.

Допустим, если мы решим зашифровать слово «SECRET», используя ключ Цезаря, равный 3, то сдвинем латинский алфавит так, чтобы он начинался с третьей буквы (D).

Итак, беря исходный вариант,

и смещая всё на 3, получаем,

где D=A, E=B, F=Cи т. д.

Используя эту схему, открытый текст «SECRET» превращается в «VHFUHW». Чтобы ктото мог восстановить исходный текст, вы сообщаете ему, что ключ – 3.

Данный алгоритм иллюстрирует, как действует симметричное шифрование.

* 1. **Симметричное шифрование и управление ключами**

Симметричное шифрование имеет ряд преимуществ. Первое – скорость криптографических операций. Оно особенно полезно для шифрования данных, которые остаются у вас. Однако, симметричное шифрование, применяемое само по себе как средство защиты пересылаемых ценных данных, может оказаться весьма затратным просто изза сложности передачи тайного ключа.

Для установления шифрованной связи с помощью симметричного алгоритма, отправителю и получателю нужно предварительно согласовать ключ и держать его в тайне. Если они находятся в географически удалённых местах, то должны прибегнуть к помощи доверенного посредника, например, надёжного курьера, чтобы избежать компрометации ключа в ходе транспортировки. Злоумышленник, перехвативший ключ в пути, сможет позднее читать, изменять и подделывать любую информацию, зашифрованную или заверенную этим ключом. Глобальная проблема симметричных шифров (от Кольцадекодера капитана Миднайта до DES и AES) состоит в сложности управления ключами.

* 1. **Криптография с открытым ключом**

Проблема управления ключами была решена криптографией с открытым , или асимметричным, ключом , концепция которой была предложена Уитфилдом Диффи и Мартином Хеллманом в 1975 году.

Криптография с открытым ключом – это асимметричная схема, в которой применяются пары ключей: открытый (public key), который зашифровывает данные, и соответствующий ему закрытый (private key), который их расшифровывает. Вы распространяете свой открытый ключ по всему свету, в то время как закрытый держите в тайне. Любой человек с копией вашего открытого ключа может зашифровать информацию, которую только вы сможете прочитать.

Хотя ключевая пара математически связана, вычисление закрытого ключа из открытого в практическом плане невыполнимо. Каждый, у кого есть ваш открытый ключ, сможет зашифровать данные, но не сможет их расшифровать. Только человек, обладающий соответствующим закрытым ключом может расшифровать информацию.



Главное достижение асимметричного шифрования в том, что оно позволяет людям, не имеющим существующей договорённости о безопасности, обмениваться секретными сообщениями. Необходимость отправителю и получателю согласовывать тайный ключ по специальному защищённому каналу полностью отпала. Все коммуникации затрагивают только открытые ключи, тогда как закрытые хранятся в безопасности. Примерами криптосистем с открытым ключом являются Elgamal (названная в честь автора, Тахира Эльгамаля), RSA (названная в честь изобретателей: Рона Ривеста, Ади Шамира и Леонарда Адлмана), DiffieHellman (названная, правильно, в честь её создателей) и DSA, Digital Signature Algorithm (изобретённый Дэвидом Кравицом).

1. **Криптоалгоритмы**

Работа основана на использовании следующих алгоритмов шифрования: DES, AES, BlowFish. В процессе работы автор разработал свой алгоритм, основанный на алгоритме Цезаря, который получил название eXT.

* 1. **Алгоритм DES**

DES (Data Encryption Standard) — симметричный алгоритм шифрования, в котором один ключ используется как для шифрования, так и для расшифрования данных. DES разработан фирмой IBM и утвержден правительством США в 1977 году как официальный стандарт (FIPS 46-3). Алгоритм DES широко использовался при хранении и передаче данных между различными вычислительными системами; в почтовых системах, в электронных системах чертежей и при электронном обмене коммерческой информацией. Стандарт DES реализовывался как программно, так и аппаратно. Предприятиями разных стран был налажен массовый выпуск цифровых устройств, использующих DES для шифрования данных. Все устройства проходили обязательную сертификацию на соответствие стандарту.

Несмотря на то, что уже некоторое время эта система не имеет статуса государственного стандарта, она по-прежнему широко применяется и заслуживает внимания при изучении блочных шифров с закрытым ключом. Длина ключа в алгоритме DES составляет 56 бит.

Систему DES вполне можно использовать в небольших и средних приложениях для шифрования данных, имеющих небольшую ценность.

Общая схема DES



* 1. **Алгоритм AES**

В 2001 году после специально объявленного конкурса в США был принят новый стандарт на блочный шифр, названный AES (Advanced Encryption Standard), в основу которого был положен шифр Rijndael, разработанный бельгийскими специалистами. Шифр Rijndael/AES характеризуется размером блока 128 бит, длиной ключа 128, 192 или 256 бит и количеством раундов 10, 12 или 14 в зависимости от длины ключа.

В отличие от шифров, предлагаемых DES основу Rijndael составляют так называемые линейно-подстановочные преобразования. Блок данных, обрабатываемый с использованием Rijndael, делится на массивы байтов, и каждая операция шифрования является байт-ориентированной. Каждый раунд состоит из трех различных обратимых преобразований, называемых слоями. Эти слои следующие.

Нелинейный слой. На этом слое выполняется замена байтов. Слой реализован с помощью S-блоков, имеющих оптимальную нелинейность, и предотвращает возможность использования дифференциального, линейного и других современных методов криптоанализа.

Линейный перемешивающий слой гарантирует высокую степень взаимопроникновения символов блока для маскировки статистических связей. На этом слое в прямоугольном массиве байтов выполняется сдвиг строк массива и перестановка столбцов.

Слой сложения по модулю 2 с подключом выполняет непосредственно шифрование.

Шифр начинается и заканчивается сложением с ключом. Это позволяет закрыть вход первого раунда при атаке по известному тексту и сделать криптографически значимым результат последнего раунда.

В алгоритме широко используются табличные вычисления, причем все необходимые таблицы задаются константно, т.е. не зависят ни от ключа, ни от данных. Функции шифрования и расшифрования различны.

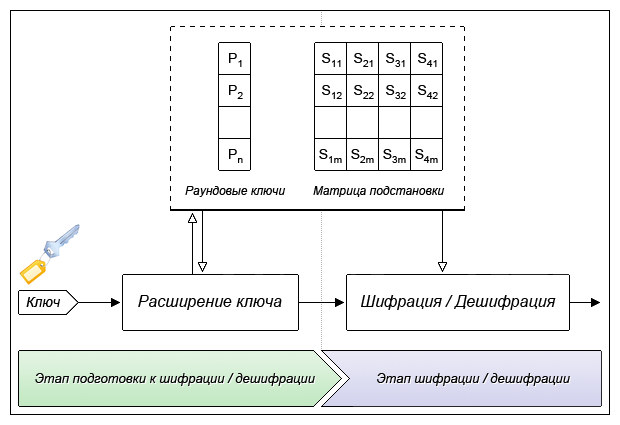
Алгоритм Rijndael хорошо выполняется как в программной, так и в аппаратной реализации. Rijndael имеет небольшие требования к памяти, что делает его пригодным для систем с ограниченными ресурсами. Надежность шифрования алгоритмом Rijndael очень высоко оценивается специалистами.

* 1. **Алгоритм BlowFish**

В 1994 году, на семинаре Fast Software Encryption в Кембридже, а впоследствии и в журнале «Lecture Notes in Computer Science» (#809, 1994), Брюс Шнайер презентовал свой алгоритм блочного шифра, который был назван Blowfish.

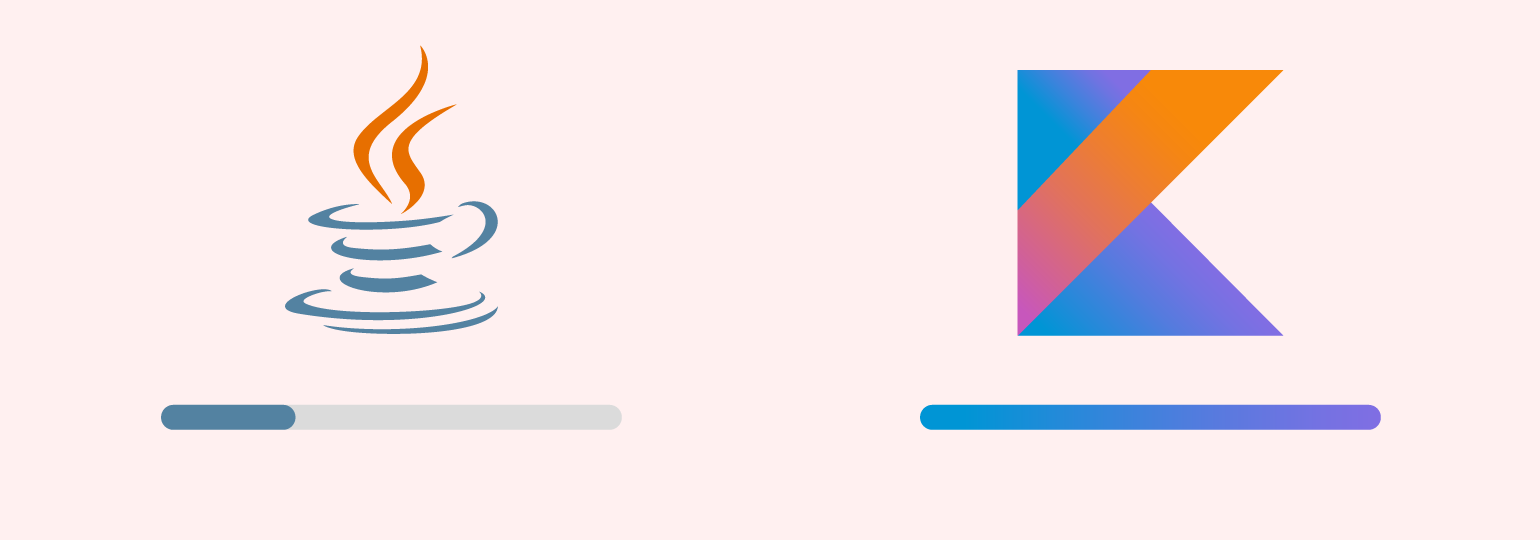
Отличительными особенностями этого алгоритма стала более высокая степень криптостойкости, нежели алгоритма DES (в том числе за счет использования переменной длины ключа, до 448 бит), высокая скорость шифрации/дешифрации (за счет генерации таблиц замены) и конечно — возможность его свободного применение для любых целей.

В общем случае алгоритм состоит из двух этапов — расширение ключа и шифрация/дешифрация исходных данных.



На этапе расширения ключа, исходный ключ преобразуется в матрицу раундовых ключей (P) и матрицу подстановки (S, Substitution-box) (или замены), общим объемом в 4168 байт.

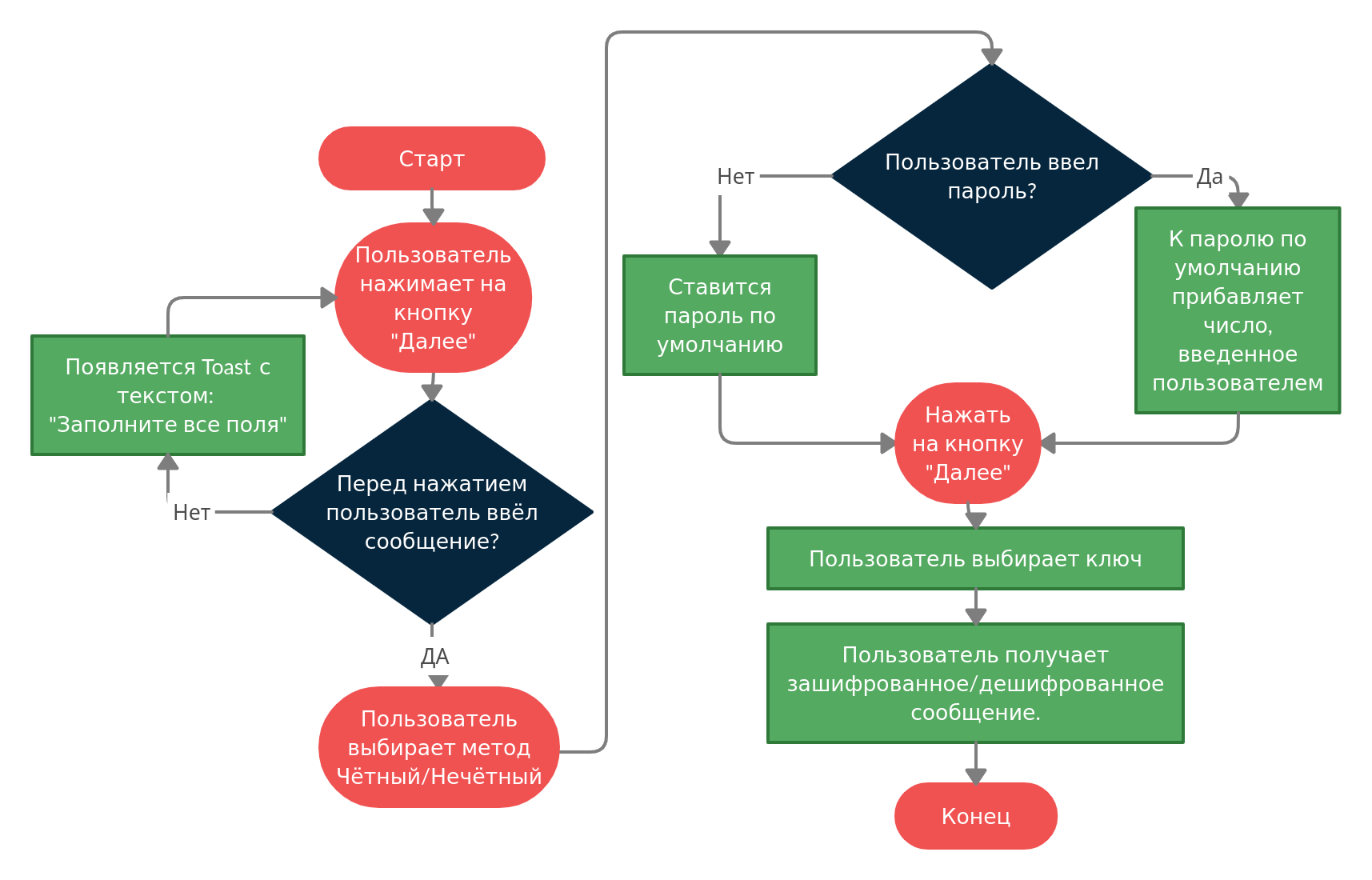
1. **Разработка приложения «Cryptographer — Шифровальщик»**

Проект представляет разработку Android-приложения "MyCryptographer", благодаря которому люди смогут защищать свои данные, а также анонимно общаться в сети.

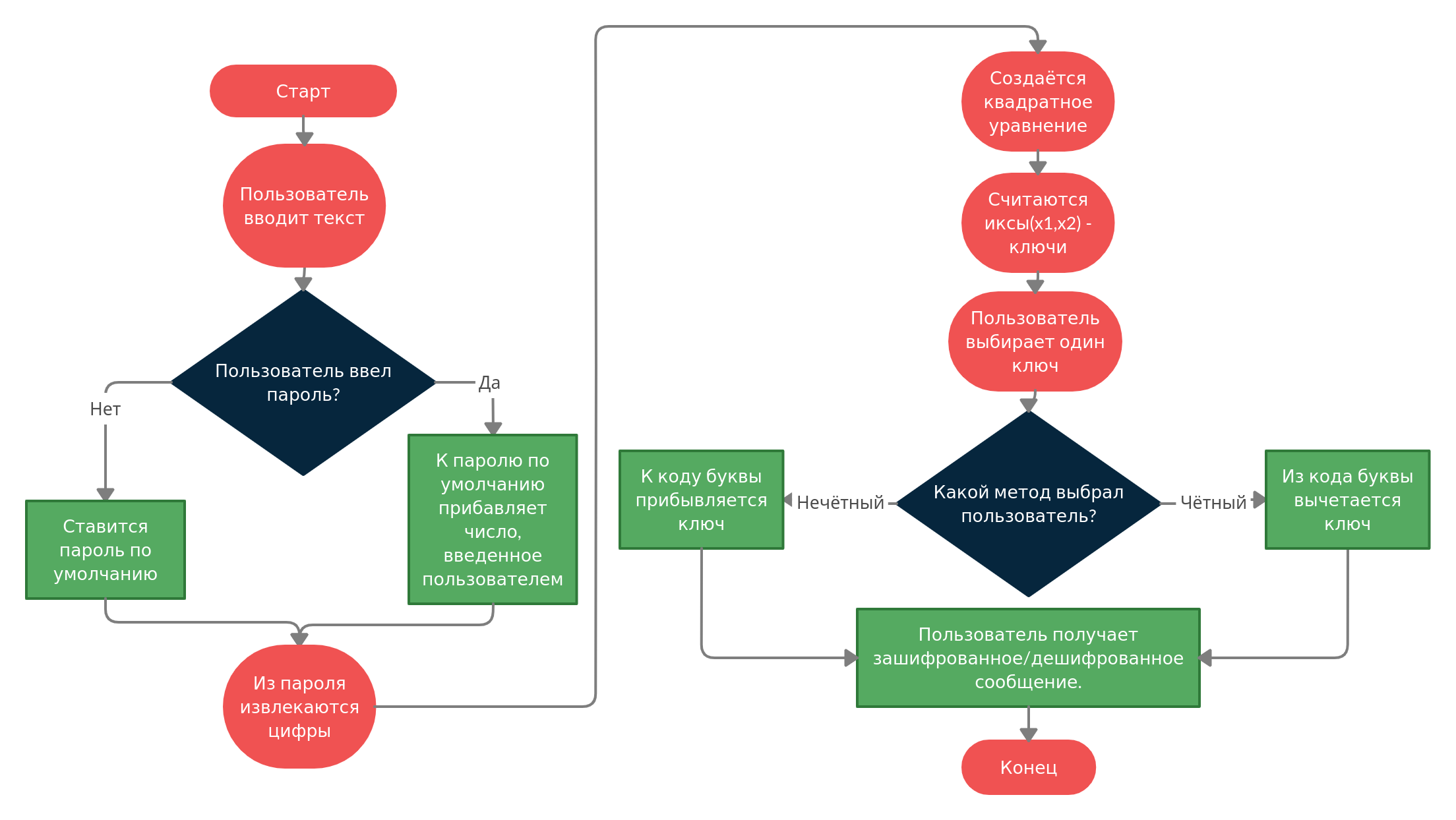
В приложении реализовано 3 шифра (AES, DES, BlowFish) и 3 алгоритма (алгоритм Цезаря, Замена символов, eXT).

MyCryptographer изначально был написан на языке программирования Java, но в итоге, приложение полностью переведено на язык программирования Kotlin, что позволило ускорить разработку приложения.

**Схема работы приложения.**



* 1. **Алгоритм «eXT»:**



***Зашифровать/Расшифровать:***   
Пользователь вводит текст/зашифрованное сообщение →

* выбирает метод (чётный/нечётный, нечётный/чётный) →
* вводит пароль →
* выбирает ключ →
* получает зашифрованное/расшифрованное сообщение.

***Пример работы алгоритма:***

Из пароля, который должен быть больше 999, извлекаются цифры. Если пользователь не ввёл пароль, то используется пароль по умолчанию.

Например, пароль=1234, извлекаются 1 2 3 и 4. (a=1, b=2, c=3, e=4)

Далее эти числа представляют квадратное уравнение ax²+bx+c=e.

Затем уравнение преобразуется: ax²+bx+c-e=0 -> ax²+bx-1=0.

Потом вычисляется дискриминант = b \* b - 4\*a\*c.

Потом вычисляются ключи (x):

x1 = (-b + Дискр) / (2\*a)

x2 = (-b - Дискр) / (2\*a)

x1 и x2 - это и есть наши ключи.

Далее каждая буква сообщения меняется.

Если пользователь выбрал нечетный метод, тогда:

Извлекается код буквы, и к нему прибавляется x1.

Например, буква А имеет код=1040, x1=3(к примеру) =>

код=1040+3 а это буква "Г".

Если метод четный, тогда из кода буквы вычитается x1=3 =>

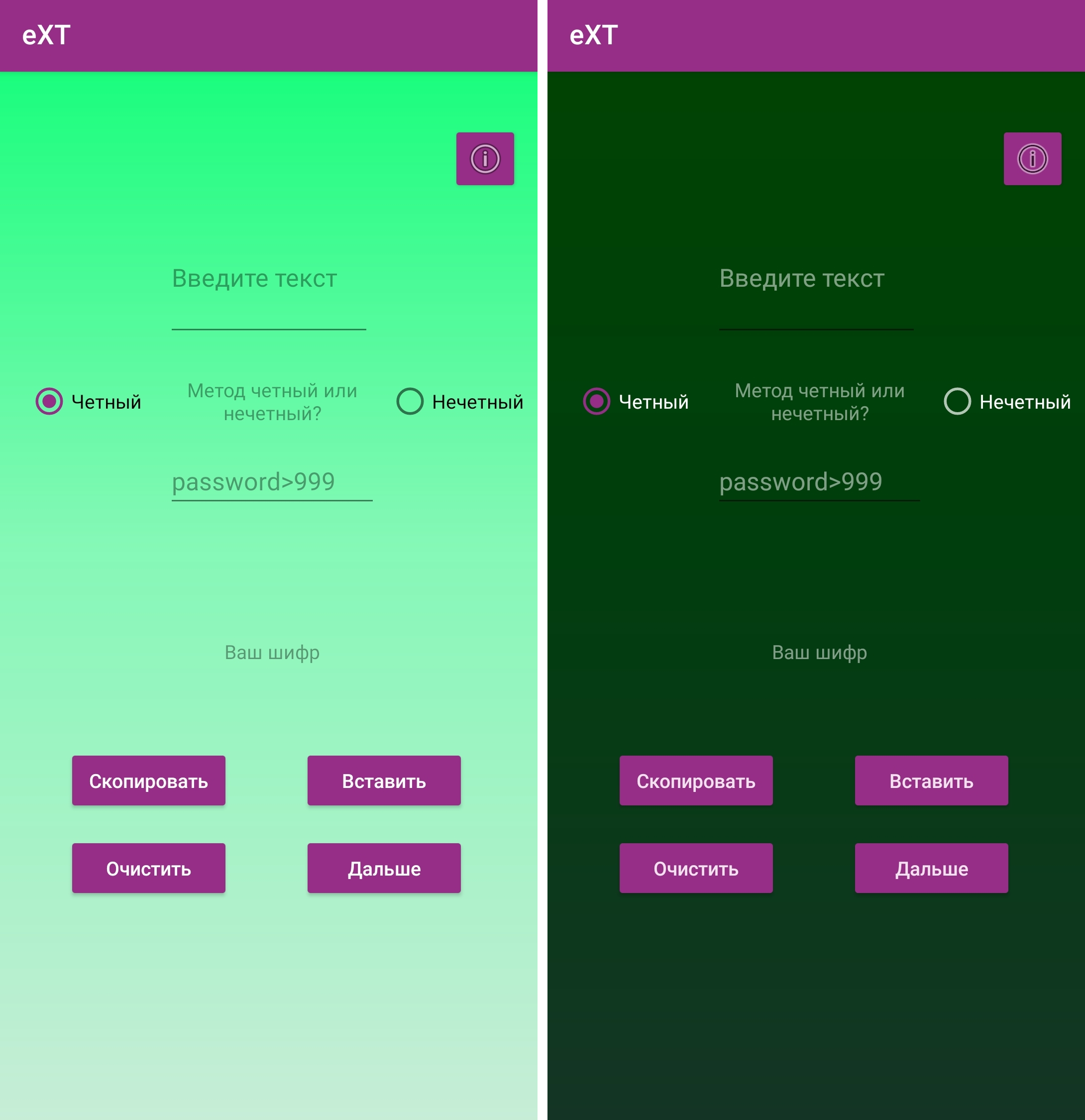
"А" превратиться в "Э".

Таким образом шифруется/дешифруется всё сообщение.

Фишка алгоритма в том, что ключи получаются из пароля, с помощью «переделанного Дискриминанта», который всегда будет возвращать именно два ключа.

Алгоритм, на данный момент, работает только с русскими словами.

* 1. **Графический пользовательский интерфейс (GUI):**

Тема приложения меняется в зависимости от глобальной темы в Android. Если у пользователя стоит светлая тема, то приложение будет со светлой темой. Если стоит темная, то приложение будет в темном оформлении.

Но пользователь может менять цвет акцента кнопок и тулбара.

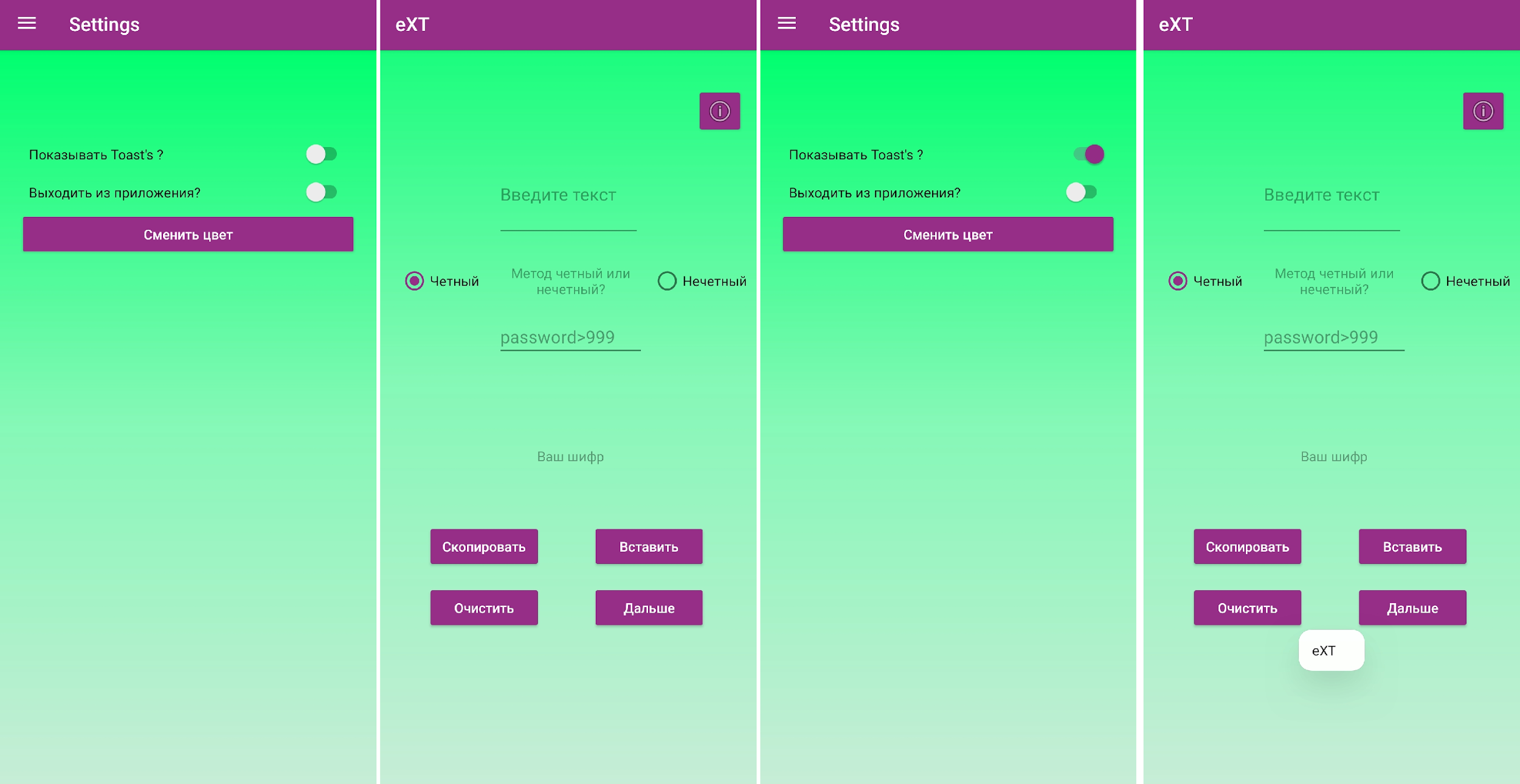
* 1. **Экран настроек:**

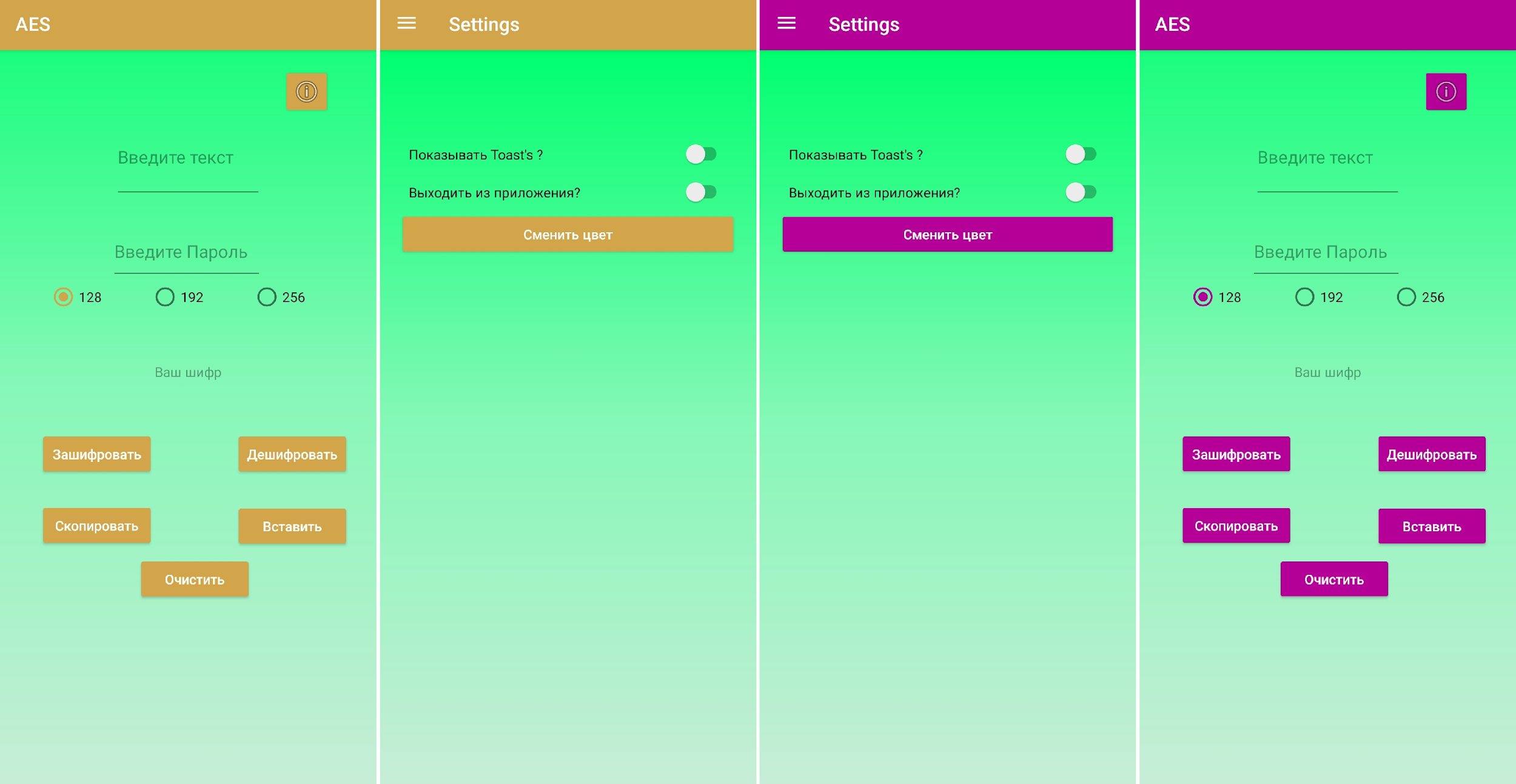
На данный момент у приложения всего пару настроек.

1) Показывать/Скрывать Toast.

2) Выходить из приложения или нет.

3) Смена цвета акцента.

* 1. **Элемент Toast.**

Он включен по умолчанию. Toast нужен для того, чтобы новый пользователь быстрее привык к интерфейсу программы. Если пользователь знает, что он делает, то он может выключить Toast. Например, если пользователь выбирает алгоритм/шифр, то появляется Toast:

* 1. **Выход из приложения.**

У пользователей Android-устройств часто бывают такие моменты, когда они случайно нажимают кнопку «Обратно». Особенно такое часто происходит на маленьких экранах. Если пользователь встречается с данной проблемой достаточно часто, то он может обезопасить себя, выключив пункт «Выходить из приложения?». Чтобы выйти из приложения кнопку «Обратно» нажимать бесполезно, необходимо использовать кнопки «Домой» или «Задачи». С жестами тоже самое.

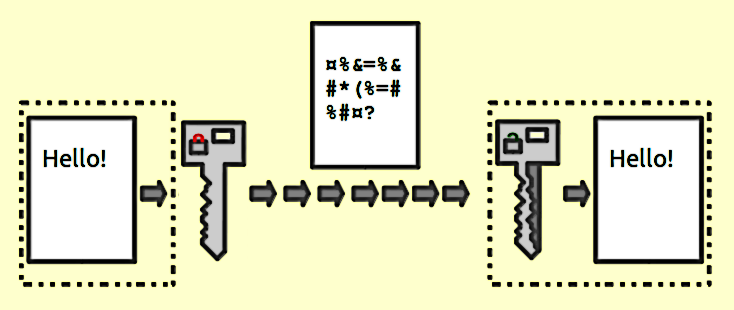
* 1. **Смена цвета акцента.**

По умолчанию для кнопок стоит красный цвет, а для тулбара зелено-синий. Если пользователю это не нравится, то он может изменить цвет кнопок и тулбара. После нажатия на кнопку «Сменить цвет» создаётся единый цвет. Этот цвет заменяется как для кнопок, так и для тулбара. Единый цвет записывается в файл настроек как шестнадцатеричная система счисления(RGB).

Обобщающим элементом настроек является запись состояния каждого пункта в файл. Если перезапустить приложение, то все настройки сохранятся. Таким образом интерфейс подстраивается под пользователя, а благодаря настройкам экономит время так, как пользователю не нужно при каждом запуске заново создавать идеальную цветовую гамму.

* 1. **Обмен ключами шифрования:**

Чтобы общение по сети было анонимным, пользователям нужно встретиться и договориться о ключе шифрования. На данным момент это единственный способ создания приватного ключа.

В следующей версии программы будут добавлены ассиметричные методы шифрования, например «GPG».

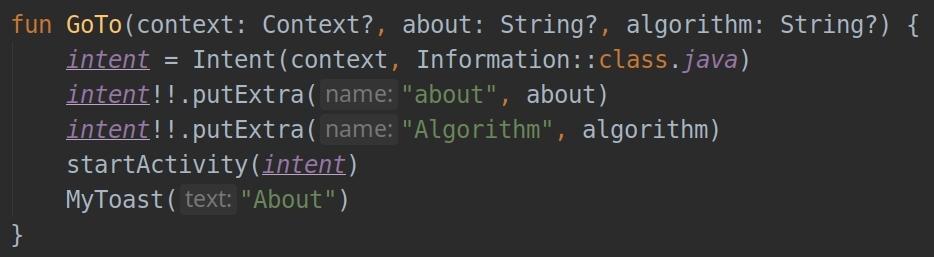
Этот метод позволит пользователям не встречаться. Им не придется договариваться. Так как «GPG» используется 2 ключа. Публичный и приватный. Публичный ключ используется для шифрования, его можно не скрывать от глаз окружающий. А приватный ключ используется для расшифровки, его необходимо хранить в безопасности, никому не показывать.

Благодаря такой схеме расшифровать сообщение может только владелец приватного ключа. Даже тот, кто зашифровывал сообщение, не может произвести обратную операцию.

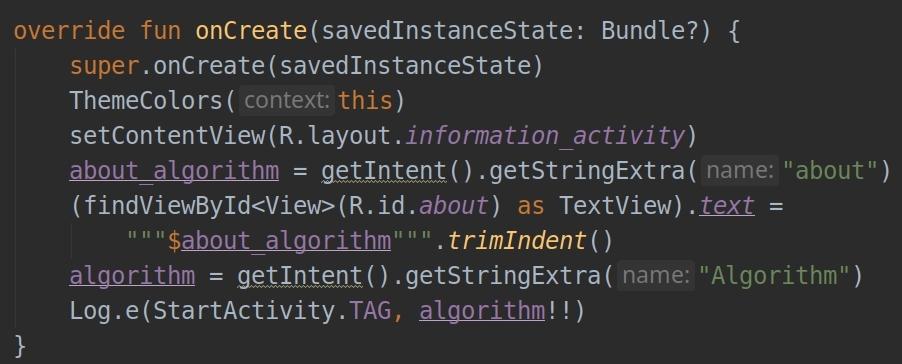
* 1. **Оптимизация приложения**

Из главного класса через методы переходим к разным активностям. (Например, DES, AES)

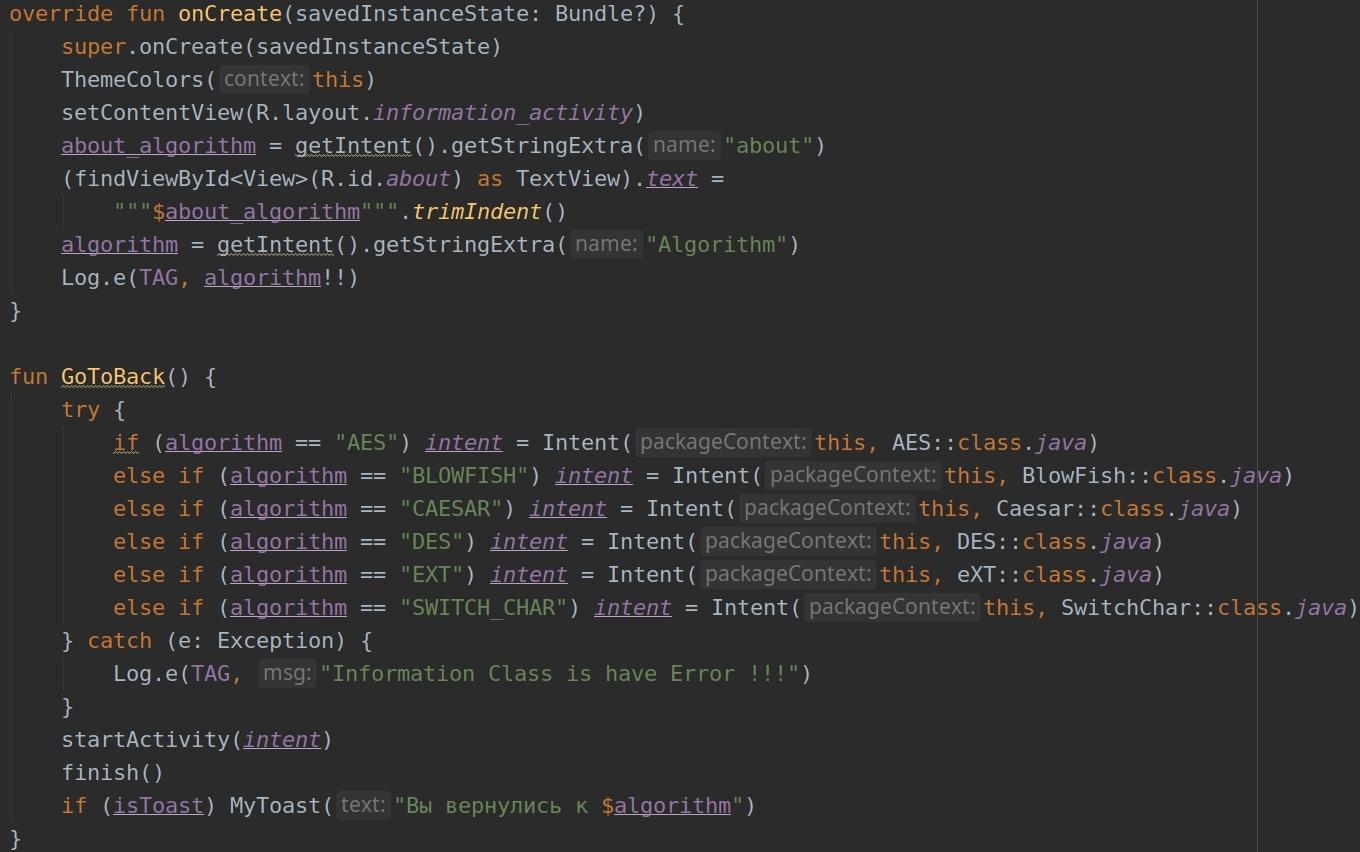
Из DES/AES с помощью «intent» в информационный класс передаем R.string.DES\_About и метод шифрования.

Когда мы открываем информационный класс ставится разметка того алгоритма, который мы передаём.



Из информационного класса, чтобы вернуться назад, мы проверяем какой алгоритм мы сейчас читаем, и через «intent» возвращаемся обратно. Благодаря Intent получилось оптимизировать программу. Всего 6 классов, значит должно быть 6 разметок под каждый класс. Но получилось так, что классов может быть много, а разметка всего одна.



1. **Заключение**

За всё время создания приложения я научился:

* Программировать на языках Java/Kotlin, XML(расширяемый язык разметки)
* Научился создавать различного рода разметки - XML
* Работать с циклами, условиями, исключениями.
* Создавать активности, фрагменты и работать с ними.
* Создавать функции, классы и их наследников.
* Работать с разметкой и взаимодействием разметки с классом активности/фрагмента.
* Работать с кнопками и обработкой событий.

Благодаря этим знаниям у меня получилось достичь своей цели.