Оглавление

[Введение 1](#_Toc453301035)

[Глава 1. 3](#_Toc453301036)

[1.1 Обзор мобильных устройств. 3](#_Toc453301037)

[1.2 Основы и история криптографии 9](#_Toc453301038)

[Глава 2. 12](#_Toc453301039)

[2.1 Предполагаемый характер защищаемой информации 12](#_Toc453301040)

[2.2 Исследование существующих решений 13](#_Toc453301041)

[2.2.1 Boxcryptor Classic 14](#_Toc453301042)

[2.2.2 Encryption Manager 16](#_Toc453301043)

[2.2.3 SSE Шифрование 18](#_Toc453301044)

[2.2.4 Анализ и сравнение существующих решений 20](#_Toc453301045)

[2.3 Сравнение и Выбор Алгоритмов шифрования 24](#_Toc453301046)

[2.3.1 Алгоритм ГОСТ 28147-89 24](#_Toc453301047)

[2.3.2 Алгоритм AES(Rijndael) 32](#_Toc453301048)

[2.3.3 Алгоритм TwoFish 42](#_Toc453301049)

[2.3.4 Сравнение Алгоритмов. 48](#_Toc453301050)

[1.4 Исследование правого аспекта. 51](#_Toc453301051)

[Глава 3 54](#_Toc453301052)

[3.1. Предполагаемый функционал. 54](#_Toc453301053)

[3.2 Выбор средств реализации. 54](#_Toc453301054)

[3.3 Описание программной реализации. Интерфейс. 56](#_Toc453301055)

[3.4 Описание программной реализации. Функционал. 59](#_Toc453301056)

[3.5 Руководство пользователя. 61](#_Toc453301057)

[3.5.1 Осуществление шифрования. 61](#_Toc453301058)

[3.5.2 Осуществление расшифровывания. 61](#_Toc453301059)

[3.5.3 Генерация Пароля. 62](#_Toc453301060)

# Введение

Прежде чем говорить об актуальности темы необходимо разобраться с самим понятием мобильных устройств, например смартфона - мобильного телефона, дополненного функциональностью карманного персонального компьютера.

При расширении функциональности мобильных устройств увеличился объём и тип хранимой на мобильных устройствах информации, вследствие чего появилась проблема защиты информации, находящейся на мобильном устройстве. Современные операционные системы на смартфонах только начали встраивать поддержку шифрования пользовательских данных, однако не все мобильные устройства поддерживают переход на новые версии ОС со встроенной защитой данных.

Создание же программного обеспечения, предоставляющего возможность защиты данных посредством шифрования, позволит пользователям самостоятельно осуществлять защиту информации, находящейся на мобильном устройстве.

Целью данной работы является разработка программного обеспечения осуществляющего шифрование данных для мобильных устройств под управлением мобильной ОС Android с возможностью портирования приложения ну другие ОС.

Задачами работы являются:

1. Обзор мобильных устройств с точки зрения разработчика ПО.

2. Раскрытие алгоритмического процесса шифрования.

3. Рассмотреть существующие решения в области защиты данных на мобильных устройствах.

4. Выработка требований к создаваемому ПО на основе анализа существующих решений и анализа алгоритмов шифрования.

5. Разработка кроссплатформенного приложения.

Дипломная работа состоит из введения, основной части (три главы), заключения, библиографического списка и приложения. В библиографический список включены наименования N работ, среди которых M интернет ресурсы.

# Глава 1.

## 1.1 Обзор мобильных устройств.

Современный мир невозможно представить без компьютеров. Мощные вычислительные устройства позволили человечеству качественно продвинуться вперёд в своём развитии, помогая производить сложные расчёты с считанные секунды, а создание и развитие сети Internet и сети World Wide Web(WWW), которая работает на базе сети Internet, и в большинстве случаев, под сетью Internet люди понимают именно WWW, позволило распространять и получать данные независимо от того, насколько далеко мы находимся от источника или получателя этих данных. Сеть WWW совместно с веб-технологиями построения сайтов позволила большинству людей пользоваться достижениями современных технологий без необходимости изучения основ работы всемирной сети.

Согласно данным, указанным в книге [1], в период с 2012 по 2015 год количество пользователей сети Internet увеличилось на 800 миллионов человек, достигнув 3,3 миллиардов. Столь взрывной рост был вызван в том числе тем, что в данный период началось широкое распространение Мобильный устройств, поддерживающих стандарт связи 3G и уменьшение стоимости мобильного интернета в мире.

Для начала следует разобраться с самим понятием мобильного устройства, а также тем, что из себя представляют мобильные устройства на 2016 год.

Во первых, обратим внимание на само значение термина «Мобильные устройства». Согласно определению с сайта SecurityLab.ru:

Мобильные устройства – ряд устройств, который включает в себя смартфоны, планшеты, электронные книги, телефоны, КПК и нетбуки, главной особенностью которых является размер, а также количество выполняемых ими функций

Таким образом к мобильным устройствам относятся:

* смартфоны;
* планшеты;
* телефоны;
* нетбуки;
* КПК;
* электронные книги.

Учитывая ситуацию на 2016 год следует отметить, что современные нетбуки по функциональности приближаются к ноутбукам, отличаясь лишь размерами, КПК были вытеснены с рынка смартфонами, а современные телефоны, как правило, не имеют возможности устанавливать ПО ввиду своей простоты. Электронные книги же бывают в двух вариантах:

1. Электронная книга без избыточной функциональности. Такие книги, как правило используют закрытую ОС на базе ядра Linux, и их функциональность ограничена чтением электронных книг.
2. Электронная книга с дополнительной функциональностью. На данный момент, в качестве ОС для подобных книг в подавляющем большинстве случае использует мобильная ОС Android c частично урезанной функциональностью, что позволяет устанавливать на электронную книгу приложения от сторонних разработчиков, например, таковым устройством является электронная книга SonyReader.

Таким образом, в данной работе для нас представляют интерес 2 вида устройств:

1. Смартфоны.
2. Планшеты.

Смартфон - мобильный телефон, дополненный функциональностью карманного персонального компьютера. Смартфоны имеют достаточно развитую операционную систему, под которую возможно создание приложений сторонними разработчиками. Наличие полнофункциональной операционной системы делает смартфоны более привлекательными в глазах большинства пользователей. Современные смартфоны прекрасно справляются со многими задачами, выходящими за рамки функциональности, которая ожидают от телефона: работа с электронной почтой, просмотр текстовых документов и электронных таблиц, обработка изображений. Некоторые модели телефонов достаточно производительны для произведения сложных вычислений при установке соответствующего программного обеспечения. Важно отметить, что программы, написанные специально для операционной системы смартфона, являются полноценными скомпилированными в двоичный код последовательностями низкоуровневых микропроцессорных команд.

Наиболее распространёнными мобильными операционными системами на смартфонах являются (по данным 2015 года, Таблица 1) :

Таблица 1

|  |  |
| --- | --- |
| Название ОС | Количество устройств, % |
| Android | 79,9 |
| iOS | 13,6 |
| Windows 10 Mobile и Windows phone | 5,3 |

Планшетные Компьютеры – в общем смысле – это собирательное понятие, включающее в себя различные виды переносных мобильных устройств оснащённых сенсорным экраном, основной способ взаимодействия с планшетным компьютером – при помощи рук или стилуса, в зависимости от типа сенсора. Под планшетными компьютерами могут подразумеваться большое количество устройств, и основные из них:

1. Планшетный персональный компьютер – IBM PC – совместимый компьютер. Например, ноутбуки-трансформеры являются планшетными ПК, так как являются IBM совместимыми и при этом имеют возможность полноценного взаимодействия с пользователем посредством сенсорного экрана.
2. Интернет-планшет – Подвид планшетных компьютеров, которые не являются IBM PC-совместимыми, и предназначены в основном для веб-серфинга, работы с веб-приложениями, а также – запуска лишь специально написанных для них приложений. Для управления интернет-планшетом используется сенсорный экран, взаимодействие с которым реализуется при помощи пальцев или специального стилуса, без использования аппаратной клавиатуры или мыши. Современные планшеты поддерживают мультитач – касание несколькими пальцами (до 10 касаний) , и управлением планшетом специальными жестами.

В планшетных компьютерах используются Операционные системы, аналогичные операционным системам смартфонов. Соответственно, основные ОС на планшетах представлены в Таблице 2 :

Таблица 2

|  |  |
| --- | --- |
| Название ОС | Количество устройств, % |
| Android | 75 |
| iOS | 14,3 |
| Windows RT и Windows 10 | 9,1 |

Здесь следует отметить, что планшеты под управлением Windows 10 могут являться IBM PC – совместимыми компьютерами, в зависимости от аппаратной части.

Как можно заметить из указанных выше данных – семейство Android является доминирующим на рынке мобильных ОС. Более того, в абсолютных числах – в 2014 году количество активных Android устройств перешагнуло рубеж в 1 миллиард устройств, и по оценкам аналитиков, в 2016-2017 году количество активных Android устройств может достичь 2 миллиардов.

Итак, исходя из популярности данной системы, целесообразно будет начать разработку приложения для данной ОС – по причине большего охвата пользовательской аудитории.

Android – это мобильная операционная система, основанная на монолитном модифицированном ядре Linux. Изначально разрабатывалась компанией Android Inc., которую затем купила Google. Впоследствии Google инициировала создание альянса Open Handset Alliance , который сейчас занимается поддержкой и развитием платформы. Android позволяет создавать приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать библиотеки и компоненты приложений, написанные на C, С++ и других языках. Приложения под операционную систему Android являются программами в нестандартном байт-коде для виртуальной машины Dalvik, для них был разработан формат установочных пакетов .APK. Начиная с версии Android 5.0 виртуальная машина Dalvik была заменена системой, под названием Android Runtime (ART) , которая, в отличии от Dalvik, использующего Just-In-Time компиляцию, производил компиляцию в момент установки. При этом ART использует тот-же байт код, что и Dalvik, для обеспечения совместимости приложений.

Последняя выпущенная версия Android на данный момент – Android 6.0 Marshmallow, базируется на ядре Linux 3.10. Однако, большинство устройств до сих пор используют предыдущие версии ОС. На настоящий момент выпущено 12 версий системы. Распределение версий ОС показано в Таблице 3 на основе статистики на 4 мая 2016 года.[&&]

Таблица 3 Распределение версий ОС Android

|  |  |
| --- | --- |
| Версия ОС | Количество устройств, % |
| 2.2.x (Froyo) | 0,1 |
| 2.3.x (Gingerbread) | 2,2 |
| 4.0.x (Ice Cream Sandwich) | 2,0 |
| 4.1.x (Jelly Bean) | 7,2 |
| 4.2.x (Jelly Bean) | 10,0 |
| 4.3.x (Jelly Bean) | 2,9 |
| 4.4.x (KitKat) | 32,5 |
| 5.0.x (Lollipop) | 16,2 |
| 5.1.x (Lollipop) | 19,4 |
| 6.0.x (Marshmallow) | 7,5 |

На основе этой статистики можно сделать вывод о том, что большинство пользователей ОС Android используют 4.х версию, при этом абсолютное большинство пользователей используют на своих устройствах версию Android 4.4. Таким образом, приложение будет разработано с использованием средств, которые поддерживаются данной версией ОС, с целью охвата большого количества мобильных устройств и их пользователей.

## 1.2 Основы и история криптографии

Криптография - наука о методах обеспечения конфиденциальности, целостности данных и аутентификации . Она изучает методы преобразования информации, направленные на сокрытие ее содержания.

Дэвид Кон (David Kohn) в книге о криптоанализе «Выдающиеся взломщики шифров» привел данные о том, что криптография использовалась в Египте в 1900 г. до н.э. В Древней Греции и Древнем Риме криптография уже широко использовалась в разных областях деятельности, особенно в государственной сфере. (Юлий Цезарь «Записки о галльской войне», 1 век до н.э.)

Период развития криптографических наук с древних времен до 1940-х годов принято называть эрой донаучной криптографии, так как изыскания тех времен основаны на интуиции и не подкреплены доказательствами.

Вторая мировая война явилась поворотным пунктом в истории криптографии: если до войны криптография представляла собой достаточно узкую область, то после войны она стала широким полем деятельности.

В 70-х годах 20-ого века произошло два события, серьезно повлиявших на дальнейшее развитие криптографии. Первое - был принят первый стандарт шифрования данных (DES). Второе – на основе работы американских математиков У.Диффи и М.Хеллмана был разработан новый раздел криптографии - криптография с открытым ключом. Эти события были закономерным итогом того, что развивающиеся компьютерные сети и информация циркулирующая в них требовали всё большей надёжности и защищённости. Криптография стала широко востребованной не только в военной, дипломатической, государственной сферах, но также в коммерческой, банковской и т.д..

Основной инструмент современной криптографии – шифрование. Шифрование – это обратимое преобразование информации с использованием одного из алгоритмов, осуществляемое с целью сокрытия информации от неавторизованных лиц, и в то же время - с предоставлением доступа к информации доверенным лицам.

Основные алгоритмы шифрования, применяемые в наше время, можно разделить на 2 группы:

1. Симметричные алгоритмы шифрования.
2. Ассиметричные алгоритмы шифрования.

Разница между этими двумя видами алгоритмов состоит в следующем –в симметричных алгоритмах шифрования для шифрования и расшифровывания информации используется один ключ шифрования. Такие системы дешевле и проще в реализации, они быстрее работают и обладают большей гибкостью, но при этом, в случае компрометации секретного ключа, система становится полностью открыта для злоумышленника, а в случае симметричного алгоритма – точек, где используется секретный ключ больше, что даёт в итоге больше мест потенциальных утечек ключа.

В ассиметричных алгоритмах шифрования используется два ключа. Секретный ключ используется для расшифровывания сообщения. Открытый ключ генерируется с использованием секретного ключа и используется для шифрования сообщения. Ассиметричные системы дороже в применении, они дольше обрабатывают информацию, и обладают меньшей гибкостью, однако их сложнее взломать. Некоторые виды ассиметричных алгоритмов практически невозможно взломать на доступных человечеству вычислительных мощностях.

В соответствии с требованиями, предъявляемыми к работе – осуществлении шифрования данных на мобильных устройствах – целесообразно будет рассматривать симметричные алгоритмы шифрования, так как они менее требовательны к мощностям, необходимым для осуществления операций шифрования и расшифровывания.

# Глава 2.

Перед тем, как начать разрабатывать решение поставленной цели – защиты информации находящейся на мобильном устройстве от несанкционированного доступа, следует выбрать методы защиты от НСД, проанализировать уже существующие на рынке решения и на основании полученных в результате анализа данных – выработать основные технические требования (ТЗ) к разрабатываемому приложению, а также изучить правовые аспекты защиты информации на мобильном устройстве.

## 2.1 Предполагаемый характер защищаемой информации

Исходя из поставленной цели – защиты информации на мобильных устройствах – следует предполагать, что защищаемая информация является персональными данными владельца устройства.

Например:

1. Фотографии, на которых запечатлён владелец устройства или его имущество.
2. Список и контакты друзей и родственников владельца.
3. Информация о религиозных и политических предпочтениях владельца устройства.
4. Информация о прочих предпочтениях и интересах.
5. Ежедневный график.
6. Информация о медицинском обслуживании и состоянии здоровья владельца устройства
7. Прочая информация, которую владелец устройства не желает предавать огласке.

Данная информация защищается Федеральным законом № 152 "О персональных данных", и требования к защите Персональных данных, предъявляемых законом, будут рассмотрены в соответствующей части данной главы.

В тоже время следует учитывать, что не исключена возможность хранения на устройстве информации, относящейся к коммерческой тайне предприятия, сотрудником которой в том или ином виде является владелец устройства. Такой информацией может являться например :

1. Информация о работах, выполненных компанией.
2. Информация, о договорах, заключаемых данной компанией.
3. Информация о структуре компании.
4. Информация об обязанностях владельца в компании.
5. Прочая информация, утечка которой может нанести ущерб компании.

Информация, утечка которой несёт убытки для компании защищается законами:

1. Федеральный закон N 149-ФЗ "Об информации, информационных технологиях и о защите информации".
2. Федеральный закон N 98-ФЗ "О коммерческой тайне".

Требования и ограничения, накладываемые на приложение данными законами, разобраны и представлены в пункте 4 данной главы.

## 2.2 Исследование существующих решений

Прежде чем начинать разработку приложения, следует изучить уже существующие решения. В Play Market от Google были выбраны 3 популярных приложения, обладающих возможностью шифрования пользовательских файлов:

1. Boxcryptor Classic
2. Encryption Manager
3. SSE Шифрование

Представленные 3 приложения далее будут рассмотрены с точки зрения пользователей, а также будет произведён их сравнительный анализ.

### 2.2.1 Boxcryptor Classic

Бесплатная программа для Android. Разработчики позиционируют ПО как приложение, оптимизированное для работы с удалёнными облачными хранилищами, такими как Google drive, Dropbox, Microsoft SkyDrive и.т.д.

В Boxcryptor Classicиспользуется симметричное шифрование по стандарту AES-256, так-же доступны варианты AES-192 и AES-128, Все операции шифрования и дешифрования производятся непосредственно на устройстве, таким образом, секретный ключ не подвергается опасности перехвата при передаче по открытым каналам связи.

Само приложение на телефоне блокируется ПИН-кодом, таким образом не зная пин-кода доступа будет невозможно запустить приложение.

Интерфейс программы можно увидеть на рисунке 1 и рисунке 2**.**

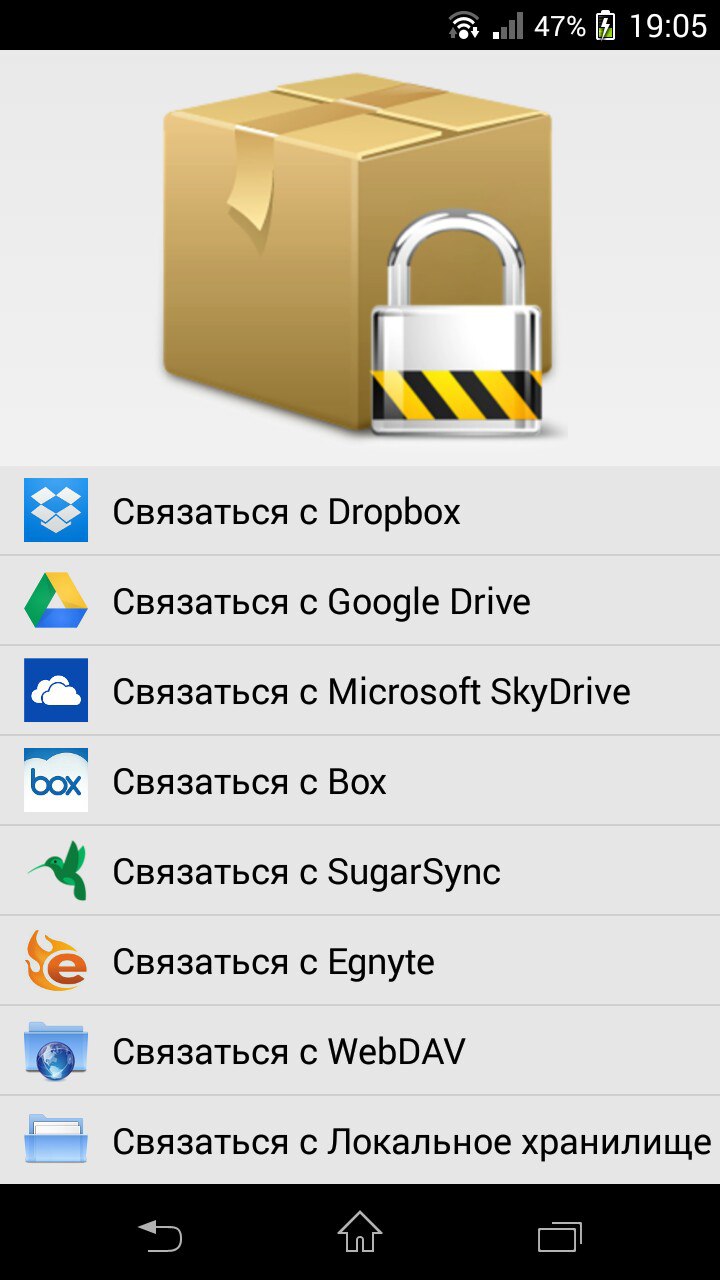


Рисунок 1

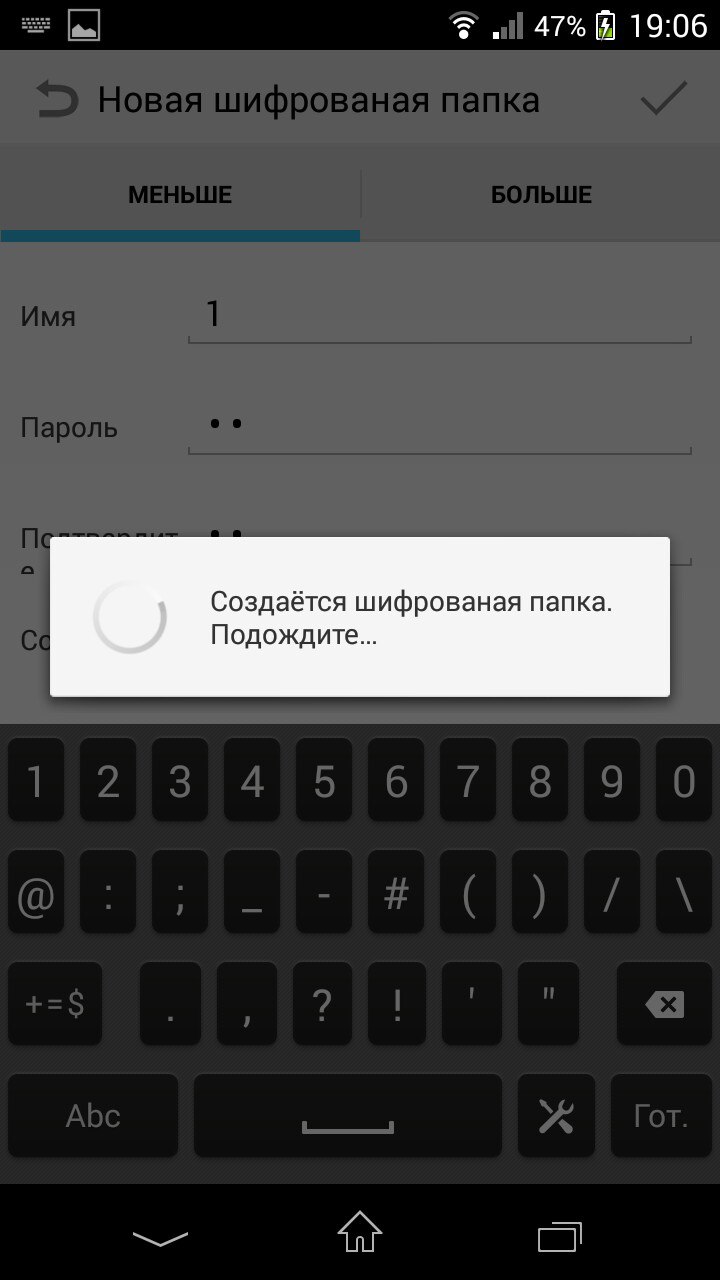


Рисунок 2

Способ взаимодействия с файловой системой: Создание зашифрованного хранилища. Все файлы, помещённые в хранилище, будут зашифрованы с применением выбранного алгоритма.

Плюсы:

* встроенная поддержка AES-256 шифрования;
* интеграция с популярными Облачными хранилищами;
* не требует ROOT доступа.

Минусы:

* не удаляет оригинал файла после шифрования;
* не ведёт историю операций;
* не очищает кэш приложения;
* частота Обновлений.

Из минусов особенно хотелось бы отменить отсутствие очищения кэша приложения – таким образом, дешифрованные файлы какое-то время хранятся на устройстве в незащищённом виде.

### 2.2.2 Encryption Manager

Приложение являющееся файловым менеджером с возможностью шифрования файлов. Защита осуществляется с помощью алгоритмов AES или Twofish шифрования.Для доступа к приложению необходимо знать мастер-пароль, который задаётся при первом запуске. Ключи шифрования для каждого файла генерируются отдельно, без участия пользователя, однако сами ключи шифруются при помощи Мастер-Пароля, и именно в этом проявляется главное узкое место программы.

Интерфейс программы можно увидеть на рисунке 3 и рисунке 4.

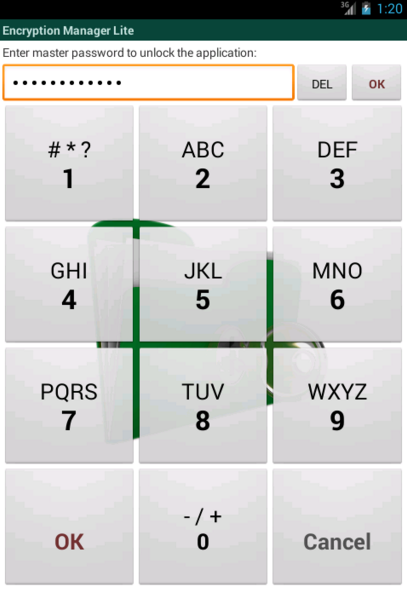


Рисунок 3

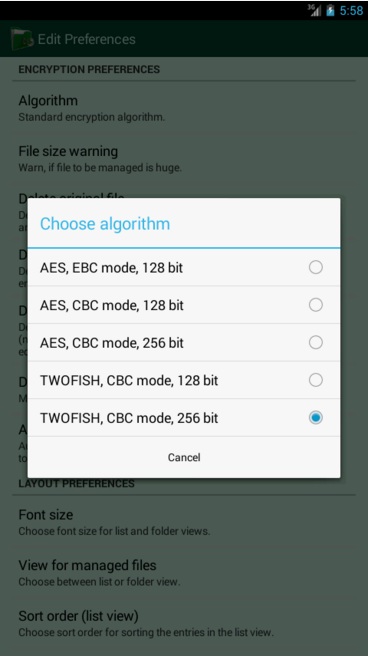


Рисунок 4

Плюсы:

* низкие системные требования;
* проста в использовании;
* поддержка двух различных алгоритмов шифрования;
* частота Обновлений.

Минусы:

* использование Мастер-пароля;
* невозможность передать зашифрованные файлы по каналу связи без сообщения мастер пароля получателю;
* в бесплатной версии возможно зашифровать не более 5 файлов.

### 2.2.3 SSE Шифрование

Данное приложение сочетает в себе функции менеджера паролей, шифратора текста и шифратора файлов. В нашем случае целесообразно подробнее разобрать возможностиданного приложения в области шифрования файлов.

SSE использует алгоритмы шифрования: AES (Rijndael) 256bit, RC6 256bit, Serpent 256bit, Blowfish 448bit, Twofish 256bit, GOST 256bit. Так же в случае приобретения платной версии программы есть возможность использовать алгоритмы Threefish 1024bit, SHACAL-2 512bit .

Приложение не требует ROOT доступа на устройстве. Имеется функция затирания исходного файла после его Шифрования. SSE – кроссплатформенное решение, имеются версии для десктопных операционных систем семейств Windows, Linux, Mac OS X.

Интерфейс программы можно увидеть на рисунке 5, рисунке 6 и рисунке 7.



Рисунок 5

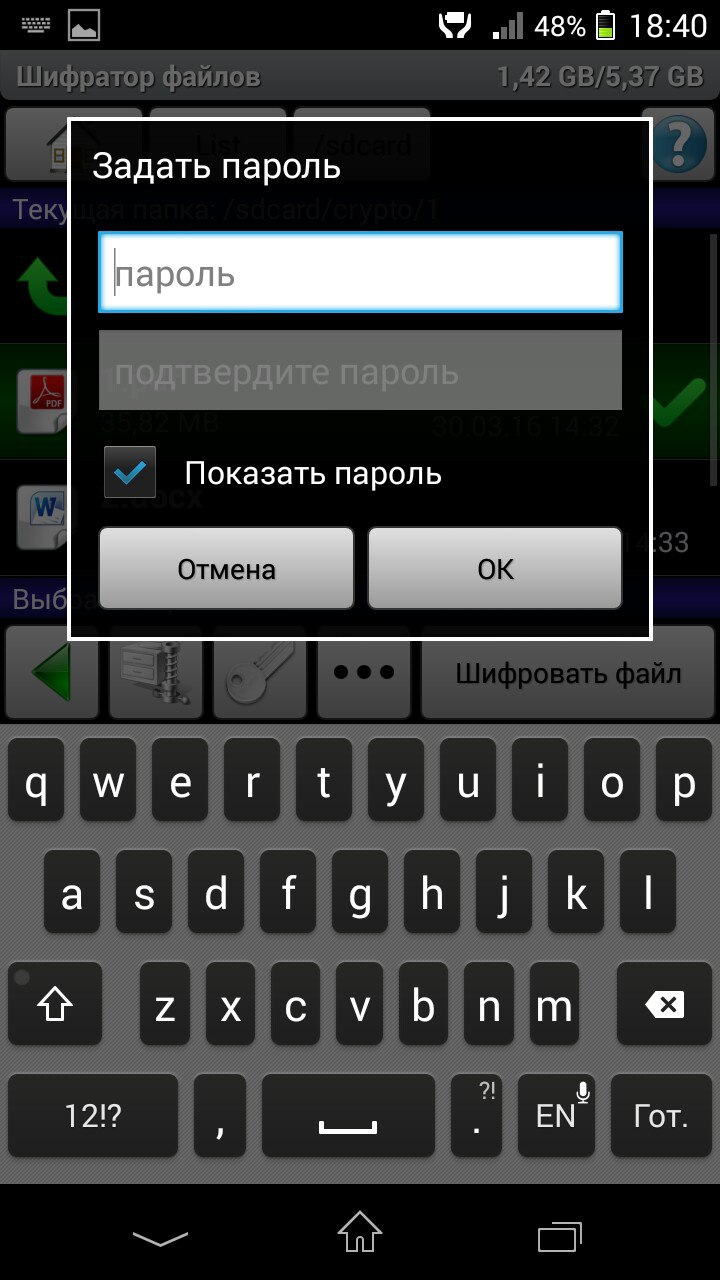


Рисунок 6

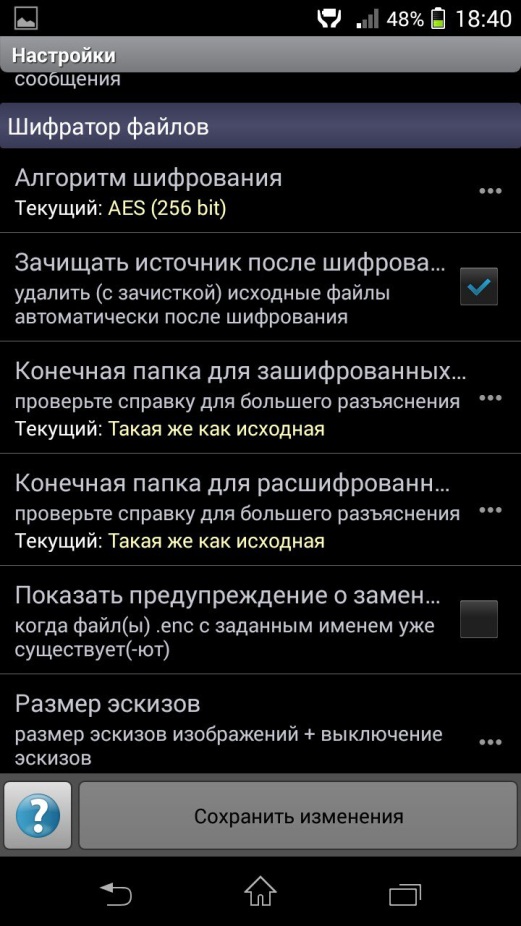


Рисунок 7

Плюсы:

* кроссплатформенное;
* наличие нескольких алгоритмов шифрования;
* затирание файлов;
* много возможностей в одном приложении;
* частота обновлений.

Минусы:

* полный функционал доступен только при покупке платной версии.

### 2.2.4 Анализ и сравнение существующих решений

Сравнение - процесс количественного или качественного сопоставления разных свойств. Производить анализ сущностей удобнее основываясь на табличных сопоставлениях характеристик приложений. Для удобства сравнение разделим характеристики на качественные (Таблица 4 и Таблица 5) и количественные (Таблица 6).

В следующей таблице(№4) представлены данные о характеристиках протестированных приложений.

Таблица 4 Характеристики Приложений

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Количество доступных алгоритмов шифрования | Дополнительный функционал | Наличие Платной версии | User Friendly интерфейс | Кросс-Платфор-менность | Частота Обновления |
| Boxcryptor Classic | 3 | - | - | - | - | Очень редко |
| Encryption Manager | 5 | - | + | + | - | Постоянно |
| SSE Шифро-вание | 6 в бесплатной версии,  8 в платной | + | + | + | + | Постоянно |

Но таблица с характеристиками не позволит нам достаточно ясно оценивать полезность приложений, так как она не учитывает практическую применимость и функциональность решения. Сопоставление функционала приложений представлено в Таблице 5.

Таблица 5 Функционал Приложений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Шифрование | Генератор Паролей | Шифратор Текста | Уборщик Буфера Обмена | Хранилище Паролей |
| Boxcryptor Classic | + | - | - | - | - |
| Encryption Manager | + | +-(Генерирует ключи шифрования, но не предоставляет их пользователю) | - | - | + |
| SSE Шифро-вание | + | + | + | + | + |

После первоначальной оценки приложений была произведена проверка работоспособности и быстродействия приложений. В качестве Android-устройства выступал смартфон Sony Xperia SP - модель C5303 с основными характеристиками:

* SoC Qualcomm Snapdragon S4 Pro (MSM8960T), 1,7 ГГц, 2 ядра Krait (ARMv7);
* GPU Adreno 320;
* оперативная память (RAM) 1 ГБ;
* внутренняя память 8 ГБ;
* операционная система Android 4.3 (Jelly Bean).

Тестирование приложений производилось на трёх различных файлах:

* pdf файл размером 37 560 320 байт;
* docx файл размером 6 770 688 байт;
* png файл размером 2 494 464 байт.

Шифрование осуществлялось при использовании алгоритма AES-256, присутствующего во всех представленных приложениях. Результаты произведённых испытаний указаны Таблице 6.

Таблица 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Время, секунды, Pdf | Время, секунды, Docx | Время, секунды, Png | Время, Затирание, PDF |
| Boxcryptor Classic | 54.350 | 5.300 | 3.000 | - |
| Encryption Manager | 40.500 | 4.100 | 1.950 | - |
| SSE Шифро-вание | 14.350 | 2.350 | 1.100 | 17.400 |

#### Заключение из Анализа.

Таким образом, проанализировав существующие на платформе Android приложения, осуществляющие шифрование данных, были выявлены следующие требования и критерии, которые необходимо учитывать при разработке приложения.

При проектировании и создании приложения необходимо учитывать:

* скорость работы;
* удобство интерфейса для конечного пользователя;
* наличие встроенного файлового менеджера;
* отсутствие критически узких мест, например - использования 1-го Мастер пароля для последующего шифрования всех данных, без возможности альтернативных вариантов защиты;
* возможность Передачи защищённых файлов по открытым каналам, и возможность использовать эти файлы различных устройствах.

## 2.3 Сравнение и Выбор Алгоритмов шифрования

При реализации приложения, реализующего шифрования, самой важной частью создаваемого программного средства является выбранный алгоритм шифрования данных. От алгоритма зависит степень защищённости файлов, скорость работы, выбор средств реализации ПО. Различные алгоритмы используют различные длины ключей, шифруют данные разными, по размеру, частями, используют при шифровании различные битовые операции в ходе шифрования. Перед выбором алгоритма необходимо произвести всесторонний анализ возможных алгоритмов шифрования, оценить степень их исследованности и запас криптостойкости.

Так как предполагается достаточно большой объём шифруемых данных, для реализации в ПО предполагается использовать Симметричный алгоритм шифрования, который выполняет преобразования над блоками данных. Три наиболее популярных алгоритма рассмотрены и проанализированы ниже.

### 2.3.1 Алгоритм ГОСТ 28147-89

#### *Структура алгоритма:*

Схема алгоритма ГОСТ 28147-89 показана на Рисунке №8. Схема этого алгоритма реализована на основе сети Фейстеля, и является достаточно простой, что однозначно упрощает его программную или аппаратную реализацию.



Рисунок 8 Схема работы алгоритма ГОСТ 28147-89

Алгоритм ГОСТ 28147-89 шифрует информацию блоками по 64 бита, которые разбиваются на два подблока по 32 бита (N1 и N2). подблок N1 определенным образом обрабатывается, после чего его значение складывается со значением подблока N2 (сложение выполняется по модулю 2), затем подблоки меняются местами. Такое преобразование выполняется определенное количество раундов: 16 или 32 в зависимости от режима работы алгоритма (описаны далее). В каждом раунде выполняются следующие операции:

1. Наложение ключа. Содержимое подблока N1 складывается по модулю 232 с частью ключа К*x*.

Ключ шифрования алгоритма ГОСТ 28147-89 имеет размерность 256 битов, а К*x* — это его 32-битная часть, т. е. 256-битный ключ шифрования представляется в виде конкатенации 32-битных подключей (Рисунок №9 ):



**Рисунок 9 Ключ Шифрования ГОСТ-28147-89**

В процессе шифрования используется один из этих подключей — в зависимости от номера раунда и режима работы алгоритма.

1. Табличная замена используется таким образом: на вход подается блок данных определенной размерности (в этом случае — 4-битный), числовое представление которого определяет номер выходного значения. Например, имеем S-box следующего вида:

4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1.

Пусть на вход пришел 4-битный блок «0100», т. е. значение 4. Согласно таблице, выходное значение будет равно 15, т. е. «1111» (0 заменяется на 4, 1 — на 11, значение 2 не изменяется и т. д.).

Как видно, схема алгоритма весьма проста, что означает, что наибольшая нагрузка по шифрованию данных ложится на таблицы замен. К сожалению, алгоритм обладает тем свойством, что существуют «слабые» таблицы замен, при использовании которых алгоритм может быть раскрыт криптоаналитическими методами. К числу слабых относится, например, таблица, в которой выход равен входу, то есть:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.

1. Побитовый циклический сдвиг влево на 11 битов.

#### *Режимы работы алгоритма*

Алгоритм ГОСТ 28147-89 имеет 4 режима работы:

* режим простой замены;
* режим гаммирования;
* режим гаммирования с обратной связью;
* режим выработки имитоприставок.

Эти режимы несколько отличаются от общепринятых (описанных в paid. 1.4), поэтому стоит рассмотреть их подробнее.

Данные режимы имеют различное назначение, но используют одно и то же описанное выше шифрующее преобразование.

##### Режим простой замены

В режиме простой замены для зашифровывания каждого 64-битного блока информации просто выполняются 32 описанных выше раунда. 32-битные подключи используются в следующей последовательности:

* К0, К1, К2, К3, К4, К5, К6, К7, K0… в раундах с 1 по 24.
* K7, K6, K5, K4, K3, K2, K1, K0 в раундах с 25 по 32.

Дешифровывание в режиме простой замены производится совершенно так же, но с несколько другой последовательностью применения подключей:

* K7, K6, K5, K4, K3, K2, K1, K0 в раундах с 1 по 8.
* К0, К1, К2, К3, К4, К5, К6, К7, K0… в раундах с 9 по 32.

Аналогично стандартному режиму ЕСВ. по причине раздельного шифрования блоков режим простой замены категорически не рекомендуется использовать для шифрования собственно данных; он должен использоваться только для шифрования других ключей шифрования в много-ключевых схемах.

##### Режим гаммирования

В режиме гаммирования (Рисунок 10) каждый блок открытого текста побитно складывается по модулю 2 с блоком гаммы шифра размером 64 бита. Гамма шифра — это специальная последовательность, которая вырабатывается с помощью описанных выше преобразований следующим образом:

1. В регистры N1 и N2 записывается их начальное заполнение— 64-битная величина, называемая «синхропосылкой» (синхропосылка, практически, является аналогом вектора инициализации).



**Рисунок 10 Режим Гаммирования**

1. Выполняется зашифровывание содержимого регистров N1 и N2 (в данном случае — синхропосылки) в режиме простой замены.
2. Содержимое N1 складывается по модулю (232 - 1 ) с константой Cl = 224 +216 +28 +4, результат сложения записывается в регистр N1.
3. Содержимое N2 складывается по модулю 232 с константой C2 = 224 +216 +28 + 1 , результат сложения записывается в регистр N2.
4. Содержимое регистров N1 и N2 подается на выход в качестве 64-битного блока гаммы шифра (т. е. в данном случае N1 и N2 образуют первый блок гаммы).
5. Если необходим следующий блок гаммы (т. е. необходимо продолжить зашифровывание или дешифровывание), выполняется возврат к шагу 2.

Для расшифровывания аналогичным образом выполняется выработка гаммы, затем снова применяется операция XOR к битам зашифрованного текста и гаммы.

Для выработки той же самой гаммы шифра у пользователя, расшифровывающего криптограмму, должен быть тот же самый ключ и то же значение синхропосылки, которые применялись при зашифровывании информации.

В противном случае получить исходный текст из зашифрованного не удастся.

В большинстве реализаций алгоритма ГОСТ 28147-89 синхропосылка не является секретным элементом, однако синхропосылка может быть так-же секретна, как и ключ шифрования. В этом случае можно считать, что эффективная длина ключа алгоритма (256 битов) увеличивается еще на 64 бита синхропосылки, которую можно рассматривать как дополнительный ключевой элемент.

##### Режим гаммирования с обратной связью

В режиме гаммирования с обратной связью в качестве заполнения регистров N1 и N2, начиная со 2-го блока, используется не предыдущий блок гаммы, а результат зашифровывания предыдущего блока открытого текста **(**Рисунок 11**)**. Первый же блок в данном режиме генерируется полностью аналогично предыдущему.



**Рисунок 11 Выработка гаммы шифра в режиме гаммирования с обратной связью**

##### Режим выработки имитоприставки

Имитоприставка — это криптографическая контрольная сумма, вычисляемая с использованием ключа шифрования и предназначенная для проверки целостности сообщений.

Генерация имитоприставки в рамках алгоритма ГОСТ 28147-89 выполняется следующим образом:

1. Первый 64-битный блок информации, для которой вычисляется имитоприставка, записывается в регистры N1 и N2 и зашифровывается в сокращенном режиме простой замены, в котором выполняются первые 16 раундов из 32.
2. Полученный результат суммируется по модулю 2 со следующим блоком информации с сохранением результата в N1 и N2.
3. N1 и N2 снова зашифровываются в сокращенном режиме простой замены и т. д. до последнего блока информации.

Имитоприставкой считается 64-битное результирующее содержимое регистров N1 и N2 или его часть. Чаще всего используется 32-битная имитоприставка. Этого достаточно, поскольку, как и любая контрольная сумма, имитоприставка предназначена, прежде всего, для защиты от случайных искажений информации.

Имитоприставка используется следующим образом:

1. При зашифровывании какой-либо информации вычисляется имитоприставка открытого текста и посылается вместе с шифротекстом.
2. После расшифровывания имитоприставка снова вычисляется и сравнивается с присланной.
3. Если вычисленная и присланная имитоприставки не совпадают— шифр- текст был искажен при передаче или использовались неверные ключи при расшифровывании.

Имитоприставка особенно полезна для проверки правильности расшифровывания ключевой информации при использовании многоключевых схем.

#### Криптоанализ ГОСТа 28147-89

С момента опубликования ГОСТа на нём стоял ограничительный гриф «для служебного пользования», и формально шифр был объявлен «полностью открытым» только в мае 1994 года. История создания шифра и критерии разработчиков по состоянию на 2016 год не обнародованы.

Высокая стойкость алгоритма ГОСТ 28147-89 дости­гается за счет следующих факторов:

* большой длины ключа— 256 битов, и вместе с секретной синхропосылкой эффективная длина ключа увеличивается до 320 битов;
* 32 раундов преобразований; уже после 8 раундов достигается полный эф­фект рассеивания входных данных: изменение одного бита блока откры­того текста повлияет на все биты блока шифртекста, и наоборот, т. е. су­ществует многократный запас стойкости.

Однако в алгоритме ГОСТ 28147-89 существуют и уязвимости. Например, секретные таблицы замен могут быть вычислены с помощью следующей атаки:

1. Устанавливается нулевой ключ и выполняется поиск «нулевого вектора», т. е. значения где — функция раунда алгоритма. Этот этап занимает порядка 232 операций шифрования.
2. С помощью нулевого вектора вычисляются значения таблиц замен, что занимает не более 211 операций.

Указанный выше способ – не единственная возможная атака на ГОСТ 28147-89. В различное время были опубликованы следующие работы, посвящённые криптоатакам на алгоритм ГОСТ 28147-89. Далее в таблице( Таблица 7)приведены краткие сведения по некоторым опубликованным работам, посвящённым криптоатакам на ГОСТ 28147-89.

**Таблица 7**

|  |  |  |  |
| --- | --- | --- | --- |
| Название Работы | Количество раундов | Количество Текста  (ОТ-открытый Текст, СПТ-специально подобранный текст | Вычислительная Сложность |
| 1R атака на связанных ключах | 32 | 235 СПТ | 236 |
| Метод рационального продолжения | 32 | В среднем 10-100 СПТ | 1010 |
| Атака методом бумеранга | 32 | 210  пар открытого/зашифрованного текста (СПТ) | 226 |
| Атака Исобе | 32 | 232 ОТ | 2224 |
| Атака на шифросистему с 12 связанными ключам | 8 | От 250 до 2106 СПТ | 260 |
| Улучшенная атака скольжения | 24 | 264 ОТ | 263 |

На основе представленных выше данных и в заключении всего сказанного можно сделать вывод о том, что ГОСТ 28147-89 обладает высокой криптостойкостью. В пользу этого утверждения говорит так-же то, что алгоритм шифрования, описанный в ГОСТ 28147-89 ныне вместе с новым алгоритмом «Кузнечик» был опубликован как часть стандарта ГОСТ Р 34.12-2015.

### 2.3.2 Алгоритм AES(Rijndael)

#### Структура алгоpитма

Алгоритм AES представляет блок данных в виде двумерного байтового массива размером 4x4. Все операции производятся над отдельными байтами массива, а также над независимыми столбцами и строками.

В каждом раунде алгоритма выполняются следующие преобразования :

1. SubBytes.
2. ShiftRows.
3. MixColumns.
4. AddRoundKey.

1. Операция SubBytes, представляющая собой табличную замену каждого байта массива данных согласно Таблице замен алгоритма AES (Рисунок 12 )



**Рисунок 12 Таблица Замен AES**

Таблица меняет входное значение 0 на 63 (шестнадцатеричное значение), 1 — на 7С и т. д.

Вместо данной табличной замены можно выполнить эквивалентную ей комбинацию двух операций:

* вычисление мультипликативной обратной величины от входного значения в конечном поле GF(28); обратной величиной от 0 является 0;
* выходное значение b вычисляется следующим образом:

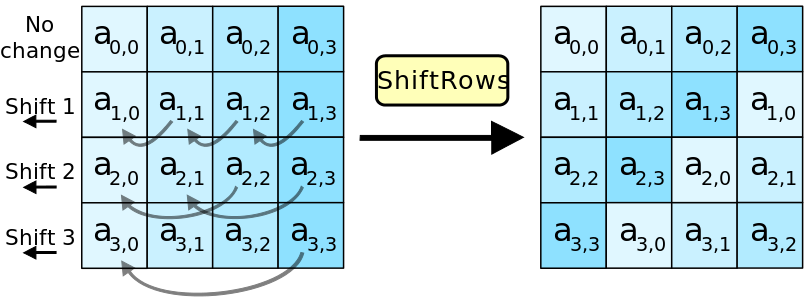
(1)

Где:

- есть i-ый бит b,

  — i-ый бит константы c=01100011;

2. Операция ShiftRows, которая выполняет циклический сдвиг влево всех строк массива данных, за исключением нулевой (Рисунок 13 ShiftRows). Сдвиг i-й строки массива (для i = 1,2,3) производится на i байтов.

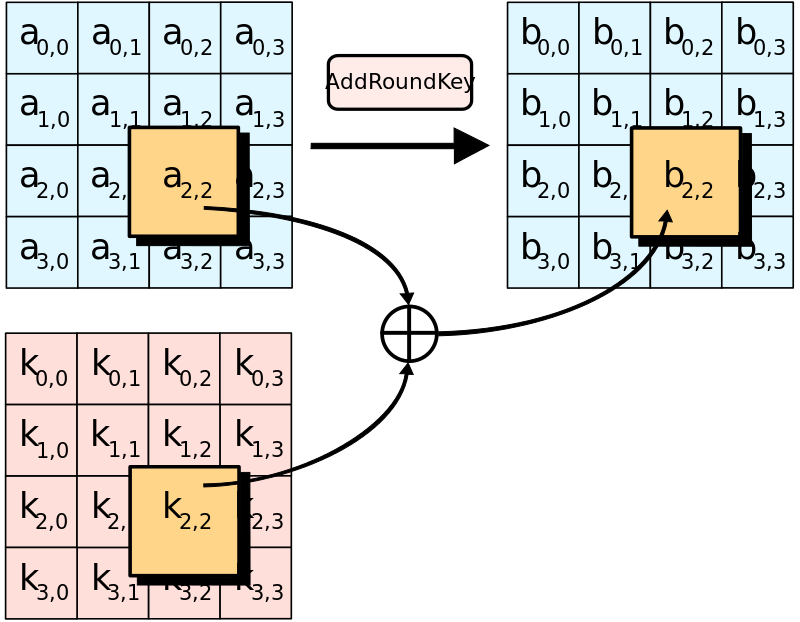


**Рисунок 13 ShiftRows**

3.Операция MixColumns. Выполняет умножение каждого столбца массива данных, который рассматривается как полином в конечном поле GF( 28), на фиксированный полином а(х):

Умножение выполняется по модулю .

4.Операция AddRoundKey (Рисунок 14 Add Round Key). Выполняет наложение на массив данных материала ключа. А именно, на i-й столбец массива данных (i = 0...3) побитовой логической операцией «исключающее или» (XOR) накладывается определенное слово расширенного ключа *WArVt,*, где r— номер текущего раунда алгоритма, начиная с 1.



**Рисунок 14 Add Round Key**

Алгоритм AES имеет несколько стандартных вариаций, отличающихся размером ключа и, соответственно, количеством раундом шифрования. Возможная длина ключей шифрования для алгоритма AES - 128, 192 и 256 битов. Зависимость количества раундов от длины ключа представлена в таблице (Таблица 8).

**Таблица 8**

|  |  |
| --- | --- |
| Длина ключа, бит | Количество раундов |
| 128 | 10 |
| 192 | 12 |
| 256 | 14 |

Перед первым раундом алгоритма выполняется предварительная модификация ключа с помощью функции AddRoundKey.А в последний раунд Алгоритма над данными не выполняется функция MixColumns.

**Процесс обработки ключа**

Алгоритм обработки ключа состоит из двух процедур:

* процедура расширения ключа;
* процедура выбора раундового ключа (ключа итерации).

**Процедура расширения ключа**

Алгоритм AES использует ключи шифрования трех фиксированных размеров: 128, 192 и 256 битов. Задача процедуры расширения ключа состоит в формировании нужного количества слов(4-байтный фрагмент) расширенного ключа для их использования в операции AddRoundKey . Таким образом, в процессе расширения ключа формируется 4 \* (R + 1) слов, где R – Количество раундов шифрования.

AES алгоритм, используя процедуру KeyExpansion() и подавая в неё Cipher Key – Исходный криптографический ключ, K, получает ключи для всех раундов. Всего получается Nb\*(Nr + 1) слов: изначально для алгоритма требуется набор из Nb слов, и каждому из Nr раундов требуется Nb ключевых набора данных. Полученный массив ключей для раундов обозначается как .

Первые Nk (Nk — размер исходного ключа шифрования К в словах) слов Wj , т. е. i = 0...(Nk -1), формируются их последовательным заполнением байтами ключа, как можно увидеть исходя из иллюстрации (Рисунок 15 ).



**Рисунок 15 Инициализация первых Nk слов**

В каждое последующее слово, , кладётся значение полученное при операции XOR между словами и . Для слов, позиция которых кратна Nk, перед операией XOR к w[i-1] применяется трансформация, которая состоит из операций RotWord, и затем операцией SubWord над результатом, полученным в RotWord, после чего производят операцию XOR между результатом операции SubWord и константой раунда Rcon[i].

Rcon[i] представляет собой слова, в которых все байты, кроме первого, являются нулевыми, а первый байт имеет значение mod;

Операция RotWord побайтно сдвигает входное слово на 1 байт влево.

Операция SubWord производит в слове побайтовую нелинейную замену в ключе, аналогичную замене, производимой в операции SubBytes.

**Алгоритм выбора раундового ключа**

На каждой итерации раундовый ключ для операции AddRoundKey выбирается из массива начиная с элемента до , где Nb – число столбцов в шифруемом блоке, и в случае с AES алгоритмом Nb всегда равно 4.

**Расшифровывание**

Расшифровывание выполняется применением обратных операций в обратной последовательности.В ходе расшифровывания данных используются следующие операции:

1. AddRoundKey – является обратной по отношению к самой себе, что обеспечивает простоту практической реализации алгоритма AES.
2. InvMixColumns - операция, которая является обратной по отношению к MixColumns.
3. InvShiftRows- операция, которая является обратной по отношению к ShiftRows.
4. InvSubBytes - операция, которая является обратной по отношению к SubBytes.

Перед первым раундом расшифровывания выполняется операция AddRoundKey . Затем выполняется R раундов расшифровывания, в каждый из которых осуществляет указанные преобразования:

1. Операция InvShiftRows производит циклический сдвиг вправо трех последних строк массива данных на то же количество байтов, на которое выполнялся сдвиг операцией ShiftRows при зашифровывании.
2. Операция InvSubBytes производит побайтно замену, обратную замене,произведённой в операции SubBytes.
3. Операция AddRoundKey. Нумерация раундов при расшифровывании производится в обратную сторону — от (R -1) до 0.
4. Операция InvMixColumns выполняет умножение каждого столбца массива данных аналогично прямой операции MixColumns, однако, умножение производится на полином , определенный следующим образом:

При этом, последний раунд расшифровывания не содержит операцию InvMixColumns.

#### Исследования криптостойкости Алгоритма AES.

Согласно оценкам авторов, AES(Rijndael) не подвержен следующим видам криптоаналитических атак:

* у алгоритма отсутствуют слабые ключи, а также возможности его вскрытия с помощью атак на связанных ключах;
* к алгоритму не применим дифференциальный криптоанализ;
* алгоритм не атакуем с помощью линейного криптоанализа и усеченных дифференциалов;
* square-атака также не применима к алгоритму AES(Rijndael);
* алгоритм не вскрывается методом интерполяции.

В ходе оценки криптостойкости алгоритма в рамках конкурса были предложены различные виды криптоаналитических атак на алгоритм AES(Rijndael) с сокращённым количеством раундов. Одной из команд криптологов была предложена атака на связанных ключах на 9- раундовую версию AES(Rijndael) с 256-битным ключом, которой необходимо 277 выбранных открытых текстов, зашифрованных на 256 связанных ключах, и 2224 операций шифрования. И, хотя существует негласное правило, гласящее, что следует модифицировать криптографический алгоритм путём двукратного увеличения количества раундов, по сравнению с максимально атакуемым, не все атаки на AES(Rijndael) являются осуществимыми на практике. Опираясь на эти сведения, а так же учитывая, что в алгоритме AES(Rijndael) выполняется, как минимум, 10 раундов, запас криптостойкости алгоритма экспертами был признан адекватным.

В результате, учитывая заключение экпертов о достаточном запасе криптостойкости AES(Rijndael), а так же простоту реализации данного алгоритма, алгоритм, названный создателями Rijndael был выбран в качестве победителя конкурса AES.

После того, как в июне 2003 года Агентство национальной безопасности США постановило, что шифр AES является достаточно надёжным, чтобы использовать его для защиты сведений, составляющих государственную тайну, а алгоритм стал известен широкой общественности в качестве стандарта, количество попыток взлома алгоритма AES увеличилось. Таким образом, в процессе эксплуатации алгоритма были предложены и более сильные атаки, которые, однако, до сих пор не привели к вскрытию алгоритма, что позволяет сомневаться в их практической реализуемости. Например была предложена атака на алгоритм AES-192 с использованием комбинации методов бумеранга и связанных ключей, при котором 9-раундовый AES-192 атакуется при наличии выбранных откры­тых текстов, каждый из которых шифруется на 256 связанных ключах, выполнением операций шифрования. Позднее эта атака была усилена, в частности предлагается атака на 10-раундовый алгоритм AES-192; для новой атаки требуется выбранных открытых текстов (на 256 связанных ключах) и операций.

Кроме атак, нацеленных непосредственно на алгоритмическую структуру AES, алгоритм так-же подвергался атакам по сторонним каналам. Атаки по сторонним каналам используют определённые особенности реализации систем, использующих данный шифр, с целью раскрыть частично или полностью секретные данные, в том числе ключ. Известно несколько подобных атак на системы, использовавшие алгоритм AES:

1. В апреле 2005 года Дэниел Бернштейн опубликовал работу с описанием атаки, использующей для взлома информацию о времени выполнения каждой операции шифрования. Данная атака потребовала более 200 миллионов выбранных и заранее подготовленных шифротекстов для нахождения ключа.
2. В декабре 2009 года была опубликована работа, в которой использование дифференциального анализа ошибок, искусственно создаваемых в матрице состояния на 8-м раунде шифрования, позволило восстановить ключ за операций.

Так же следует привести краткие данные, по некоторым работам, посвящённым криптоанализу AES(Таблица 9).

**Таблица 9**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название Работы | Тип алгоритма | Количество раундов | Количество Текста | Количество Операций |
| Improved cryptoanalysis of Rijndael | AES | 7 | От до СП |  |
| Improved cryptoanalysis of Rijndael | AES-256 | 8 | От до СП | От до |
| Improved cryptoanalysis of Rijndael | AES-256 | 9 | текстов зашифрованных на 256 связанных ключах |  |
| Cache-timing attacks on AES | AES | Не указанно | 200 миллионов выбранных шифротекстов | 2\* |
| A Diagonal Fault Attack on the Advanced Encryption Standar | AES | 8 | Не указанно |  |
| Related-Key Boomerang and Rectangle attack | AES-192 | 9 | СП зашифрованных на 256 связанных ключах |  |
| Combined Differential, Linear and Related-Key Attacks on Block-Ciphers and MAC Algorithms | AES-192 | 10 | СП зашифрованных на 256 связанных ключах |  |

В заключение разбора алгоритма AES хочется сказать, что несмотря на то, что некоторые версии Алгоритма находятся под угрозой взлома, например AES-192, версия с 256-битным ключом до сих пор является в достаточной степени криптостойкой, о чём свидетельствует то, что этот вариант алгоритма AES до сих пор является стандартом в США, а так же тот факт, что большая часть теоретически описанных атак на алгоритм AES является практически нереализуемой. Таким образом, учитывая лёгкость реализации, высокую криптостойкость и повсеместную распространённость, AES является одним из наиболее надёжных и предпочтительных алгоритмов для реализации шифрования в программных и аппаратных системах.

### 2.3.3 Алгоритм TwoFish

Алгоритм Twofish наравне с алгоритмом Rijndael был одним из финалистов конкурса AES, однако, согласно мнению экспертов, по ряду параметров Twofish проигрывал Rijndael. Отличия в алгоритмах, приведшие к выбору в качестве стандарта алгоритм Rijndael будут рассмотрены в ходе сравнения алгоритмов.

#### Структура Алгоритма

Twofish разрабатывался с учетом требований, предъявляемых к алгоритмам, участвовавшим в конкурсе AES :

* размер шифруемого блока - 128 бит;
* различные по длине ключи - 128, 192 и 256 бит;
* отсутствие слабых ключей;
* возможность эффективной программной и аппаратной реализации;
* гибкость;
* простота алгоритма — для возможности его эффективного анализа.

Twofish разбивает шифруемые данные на четыре 32-битных подблока (обозначим их А, В, С, D), над которыми производится 16 раундов шифрования, в каждом из которых выполняются следующие операции:

1. Подблок В циклически сдвигается влево на 8 битов.

2. Подблок А обрабатывается операцией g().

3. Подблок В также обрабатывается операцией g().

4. Подблок В накладывается на A с помощью сложения по модулю 232, после чего аналогичным образом выполняется наложение подблока А на подблок В.

5. Фрагмент расширенного ключа K*2r+i* (r - номер текущего раунда) складывается с подблоком А по модулю 232.

6. Аналогично предыдущему шагу, К*2г+с*, накладывается на подблок В.

7. Подблок А накладывается на С операцией XOR.

8. Содержимое подблока D циклически сдвигается влево на 1 бит.

9. Подблок В накладывается на D операцией XOR.

10. Содержимое подблока С циклически сдвигается вправо на 1 бит.

Функция g — основа алгоритма Twofish. На вход функции подается 32-битный блок X, далее происходят следующие преобразования:

1. Блок делится на 4 части по 1 байту каждая.
2. Фрагменты поступают в S-блоки, S0...S3. Таблицы замен вычисляются динамически и зависят от ключа шифрования. Каждый из получившихся байтов пропускается через свой S-блок.
3. Результаты замены из S-box-ов умножается на фиксированную матрицу MDS. Умножение выполняется в конечном поле GF(28), с образующим полиномом вида:

(2)

Матрица MDS представлена в Таблице 10:

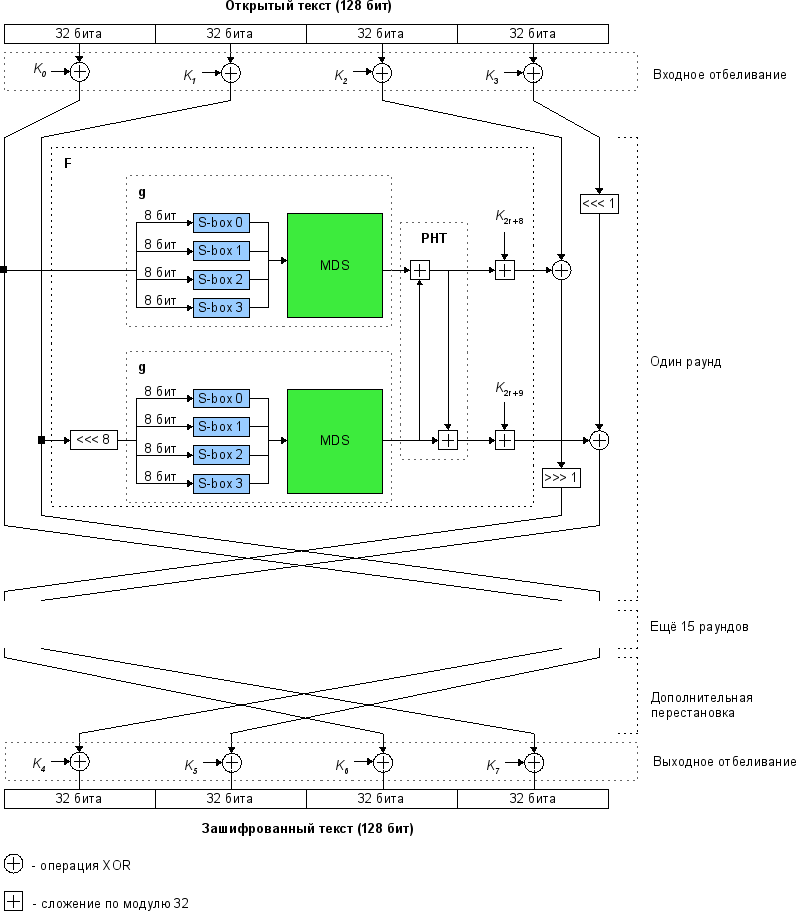
**Таблица 10**

|  |  |  |  |
| --- | --- | --- | --- |
| 01 | EF | 5B | 5B |
| 5B | EF | EF | 01 |
| EF | 5B | 01 | EF |
| EF | 01 | EF | 5B |

Перед началом шифрующих преобразований алгоритм производит входное Отбеливание. Также отбеливание производится после последнего раунда шифрования. Процедура отбеливания заключается в XOR данных с подключами перед первым раундом и после последнего раунда.

В конце каждого раунда, за исключением последнего, 32- х битный подблок А (до преобразования) и С меняются местами, подблоки В (до преобразования) и D также меняются местами.

Схематичное изображение алгоритма представлено на Рисунке 16



**Рисунок 16 Алгоритм TwoFish**

**Процедура расширения ключа**

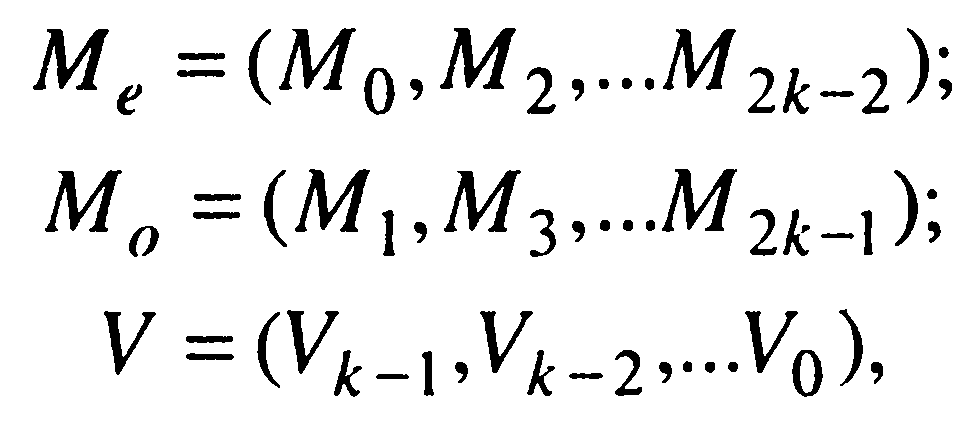
Twofish, так-же, как и Rijndael, может работать с ключами длиной 128, 192 и 256 бит. Из исходного ключа генерируется 40 4 байтных подключей, первые 8 из них используются только в операциях входного и выходного отбеливания, а остальные 32 — в раундах шифрования, по два ключа на раунд. Отличием Twofish является то, что исходный ключ шифрования применяется также и для модификации самого алгоритма, так как используемые в функции g() таблицы замен не фиксированы, а генерируются в зависимости от ключа.

В Книге **[№ книги в спискелитературы**] Алгоритм расширения ключа описывается так:

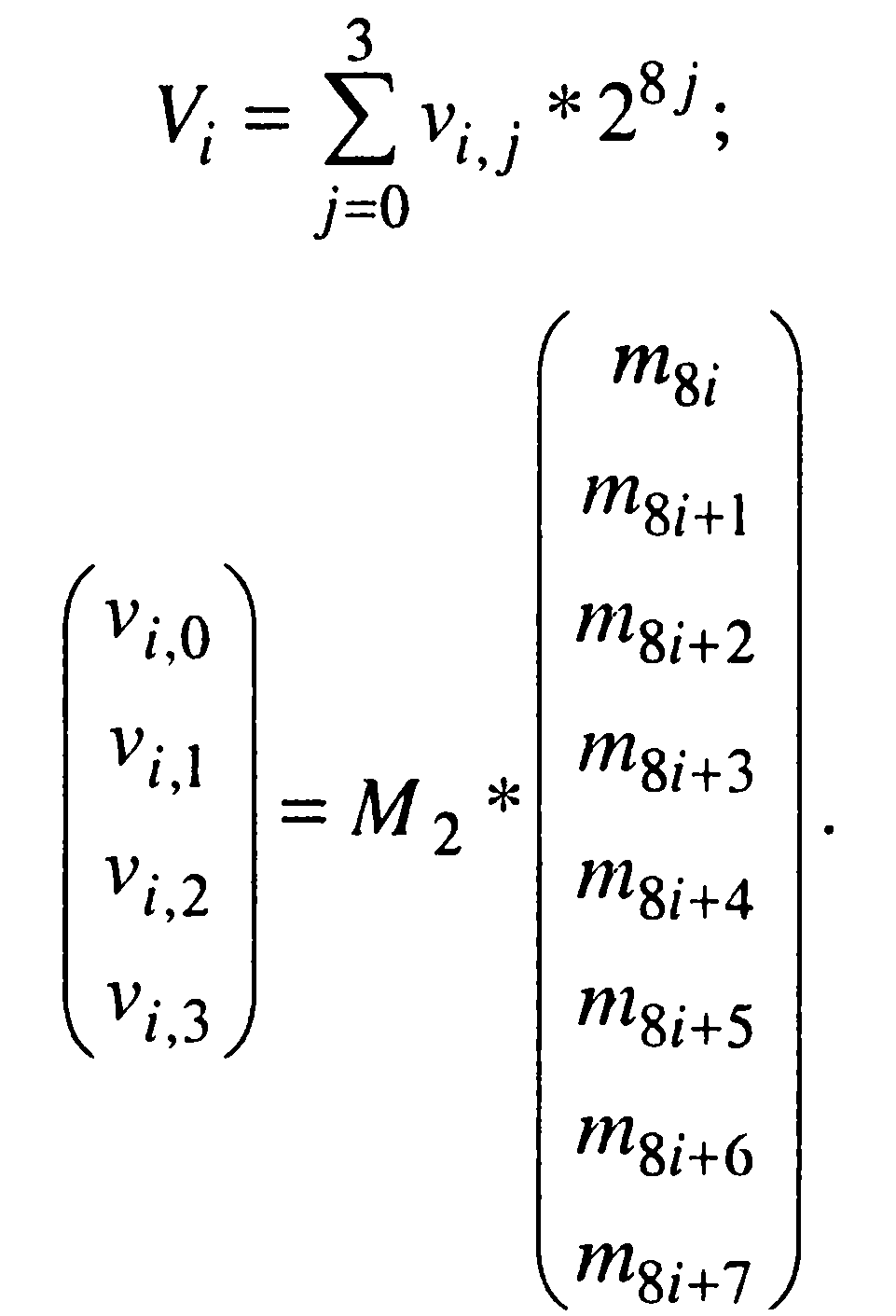
Сначала производится предварительная обработка ключа, включающая следующие шаги:

1. Выполняется инициализация переменных, участвующих в дальнейших расчетах: *к = N/64*, где N *—* размер дополненного ключа шифрования в битах, т. е. к принимает значение 2, 3 или 4. Ключ шифрования представляется в виде 8к байтов, обозначаемых, или в виде 2к 32-битных слов, обозначаемых как

2. Формируются 3 массива, каждый из которых состоит из *к* 32-битных слов:

 (3)

Где V вычисляется следующим образом:

 (4)

Матрица представлена в Таблице 11.

**Таблица 11**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 01 | A4 | 55 | 87 | 5A | 58 | DB | 9E |
| A4 | 56 | 82 | F3 | 1E | C6 | 68 | E5 |
| 02 | A1 | FC | C1 | 47 | AE | 3D | 19 |
| A4 | 55 | 87 | 5A | 58 | DB | 9E | 03 |

Генерация подключей к0...к39 производится на основе вычисленных на предварительном этапе массивов *Ме* и *М0* следующим образом:

(5)

где i= 0...19, а A и B, —промежуточные величины, вычисляемые так :

(6)

(7)

Константа р определена следующим образом:

р = 224 +216 + 28 +1,

Далее следует описание функции *()*.

Она выполняется в не­сколько шагов, состав которых зависит от размера дополненного ключа в 64- битных фрагментах, т. е. от описанного выше значения*.* В качестве параметров функция *h()* принимает 32-битное слово и массив 32-битных слов размерностью *.*

Порядок выполнения выглядит так – во первых, входное слово делится на 4 8-битных фрагмента, которые «прогоня­ются» через специфические операции замены q0 и q1. Результат замены объединяется в 32-битное слово, ко­торое складывается с третьим словом входного массива Данный шаг не выполняется, если .

Далее результат предыдущего шага или входное слово обрабатывается ана­логичным образом, но с другой последовательностью применения q0 и q1,которая различна для каждого шага, и склады­вается со вторым словом входного массива. Не выполняется, если  *= 2.*

Третий и четвёртый шаги выполняются всегда и включают в себя обработку результата предыдущего шага (или входного слова — для шага 3 и = 2), аналогичную предыдущим шагам с использованием, соответственно, первого и нулевого слов входного массива.

На пятом шаге, как и на предыдущих шагах, применяются замены q0 и q1, после чего выполняется следующее преобразование:

;

**(8)**

Где

– Байты результата выполнения замен на шаге 5

Матрица – Описана выше

H - Выходное значение Функции *h()*

Операции *q0* и *qx* представляют собой не табличные замены 8x8 битов, а вычисляют выходные значения с использованием нескольких таблиц замен 4x4, алгоритм генерации которых подробно описан в спецификации алгоритма Twofish.

#### Криптоанализ Twofish

В ходе конкурса AES экспертами были отмечены следующие недостатки алгоритма:

* низкая скорость расширения ключа при программной реализации алгоритма;
* относительно высокую сложность аппаратной реализации алгоритма;
* сложность криптоанализа алгоритма, использующего запутанную схему шифрования.

По окончанию конкурса алгоритм Twofish не получил настолько-же широкого распространения, как алгоритм AES, и потому, работы по проведению криптоанализа Twofish весьма редки. Однако некоторые работы по криптоанализу Twofish заслуживают Упоминания:

* изучение Twofish с сокращенными числом раундов показало, что алгоритм обладает большим запасом прочности, и, по сравнению с остальными финалистами конкурса AES, он оказался самым стойким. Однако его необычное строение и относительная сложность породили некоторые сомнения в качестве этой прочности;
* криптографы Fauzan Mirza и Sean Murphy предположили, что разделение исходного ключа на две половины при формировании раундовых подключей дает возможность организовать атаку по принципу «разделяй и властвуй». Однако реально подобную атаку провести не удалось;
* на 2016 год лучшим вариантом криптоанализа Twofish был вариант усечённого дифференциального криптоанализа, который был опубликован Shiho Moriai и Yiqun Lisa Yin в Японии в 2000 году. Они показали, что для нахождения необходимых дифференциалов требуется 251 подобранных открытых текстов. Тем не менее исследования носили теоретический характер, никакой реальной атаки проведено не было.

### 2.3.4 Сравнение Алгоритмов.

Все рассмотренные выше алгоритмы, так или иначе разрабатывались с целью применения в качестве стандарта шифрования:

1. Алгоритм ГОСТ 28147-89 , включённый в стандарт ГОСТ Р 34.12-2015 на данный момент является одним из алгоритмов, которыми в РФ допускается шифрование информации, содержащей Государственную Тайну.
2. Алгоритм AES(Rijndael) – так-же является стандартом шифрования в США, принят под названием FIPS 197, и Агентство национальной безопасности США утвердило использование алгоритма AES для защиты сведений, составляющих государственную тайну
3. Алгоритм TwoFish – Участвовал в конкурсе AES, и согласно мнению экспертов конкурса обладал большей криптостойкостью, чем алгоритм Rijndael, однако ввиду сложности алгоритма, было практически невозможно достоверно подтвердить заявленный уровень криптостойкости.

Для начала анализа следует выделить основные характеристики, по которым мы будем оценивать криптографические Алгоритмы:

1. Тип алгоритма.
2. Размер Блока.
3. Размер ключа.
4. Количество раундов.
5. Быстродействие.

Сводные данные по рассмотренным алгоритмам представлены в Таблица 12.

**Таблица 12**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Дата Публи-кации | Тип Алгоритма | Размер Блока,  Бит | Размер Ключа,  Бит | Количество  Раундов |
| ГОСТ 28147-89 | 30.06.1990 | Сеть Фейстеля | 64 | 256 бит | 16 / 32 |
| AES (Rijndael) | 26.11.2001 | SP-сеть | 128 | 128/  192/  256 | 10 / 12 / 14 |
| Twofish | 1998 | Модифиц-ированная Сеть Фейстеля | 128 | 256 | 16 |

Кроме объективных различий в алгоритмах, рассмотренных в Таблице 12 , следует также сравнить параметры, являющиеся субъективным мнением экспертов и пользователей представленных алгоритмов, а также параметры, которые нельзя оценить с высокой степенью точности (Таблица 13).

**Таблица 13**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Распространённость | Количество работ по Криптоанализу | Быстродействие | Количество котиков у создателей алгоритма |
| ГОСТ 28147-89 | Широко распространён в Государственных структурах и некоторой части коммерческих предприятий на территории РФ и дружественных госсударств | Среднее | Средняя скорость | Неизвестно |
| AES (Rijndael) | Высокая степень распространённости по всему миру. | Высокое | Высокая скорость | 5 |
| Twofish | Средняя степень распространённости.  Используется в некоторых СУБД в качестве альтернативного алгоритма шифрования | Низкое | Средняя Скрость | 2-3 |

Таким образом, исходя из данных, представленных в таблицах № 6 и 7, а также опираясь на описание алгоритмов, представленное выше, можно сделать следующий выбор:

1. На основании оценок экспертов относительно быстродействия алгоритма, низкой распространённости алгоритма, а так-же ввиду отсутствия возможности однозначного математического доказательства стойкости алгоритма – исключить из списка вариантов алгоритм Twofish.
2. Учитывая приблизительно равную степень криптостойкости алгоритмов AES и ГОСТ 28147-89, выбираем AES в качестве основного алгоритма Шифрования в разрабатываемом ПО, ввиду большего размера шифруемого блока и большей степени (по сравнению с ГОСТ 28147-89) распространённости данного алгоритма.

Таким образом, подводя итог, в качестве алгоритма шифрования, реализация которого планируется в первую очередь, был выбран алгоритм AES(Rijndael).

## 2.4 Исследование правого аспекта.

В случае обработки персональных данных владельца устройства создаваемым приложением, на создаваемое ПО накладывает ограничение Федеральный закон РФ № 152 "О персональных данных" []. В частности, нас интересуют следующие положения:

“ Статья 7. - Конфиденциальность персональных данных. Операторы и иные лица, получившие доступ к персональным данным, обязаны не раскрывать третьим лицам и не распространять персональные данные без согласия субъекта персональных данных, если иное не предусмотрено федеральным законом.”

Несмотря на тот факт, что как-таковые персональные данные в случае шифрования при помощи ПО, с точки зрения разработчика ничем не отличаются от прочих возможных данных, данная статья накладывает на нас требование соблюдать конфиденциальность обрабатываемых данных.

В случае же, учитывающем возможность шифрования данных, составляющих коммерческую тайну предприятия, нам следует опираться на Федеральный закон N 98-ФЗ "О коммерческой тайне" [] , а в частности на Статью 6.1, пункт 2, который гласит: -

“2. Обладатель информации, составляющей коммерческую тайну, имеет право:

1) устанавливать, изменять, отменять в письменной форме режим коммерческой тайны в соответствии с настоящим Федеральным законом и гражданско-правовым договором;

2) использовать информацию, составляющую коммерческую тайну, для собственных нужд в порядке, не противоречащем законодательству Российской Федерации;

3) разрешать или запрещать доступ к информации, составляющей коммерческую тайну, определять порядок и условия доступа к этой информации;

4) требовать от юридических лиц, физических лиц, получивших доступ к информации, составляющей коммерческую тайну, органов государственной власти, иных государственных органов, органов местного самоуправления, которым предоставлена информация, составляющая коммерческую тайну, соблюдения обязанностей по охране ее конфиденциальности;

5) требовать от лиц, получивших доступ к информации, составляющей коммерческую тайну, в результате действий, совершенных случайно или по ошибке, охраны конфиденциальности этой информации;

6) защищать в установленном законом порядке свои права в случае разглашения, незаконного получения или незаконного использования третьими лицами информации, составляющей коммерческую тайну, в том числе требовать возмещения убытков, причиненных в связи с нарушением его прав”.

Таким образом целесообразно будет в ТЗ к разрабатываемому программному средству добавить требование по обработке(шифрованию) данных исключительно средствами мобильного устройства, на котором находятся шифруемые данные, дабы исключить возможность перехвата и нанесения вреда защищаемой информации.

# Глава 3

## 3.1. Предполагаемый функционал.

Итак, исходя из требований, сформулированных в главе 2 наше приложений должно отвечать следующим требованиям:

* алгоритм шифрования - AES;
* все операции должны осуществляться исключительно на устройстве, без передачи данных по сети;
* приложение должно обладать User-Friendly интерфейсом;
* приложение должно иметь встроенный способ работы с файловой системой устройства, независимый от наличия других приложений;
* целевая операционная система для приложения – Android;
* разрабатываемое Приложение, по возможности, должно иметь возможность переноса на другие ОС с максимальным повторным использованием кода.

Также, для удобства конечного пользователя предполагается внедрить следующие функции:

* иметь возможность генерации паролей заданной длина;
* иметь возможность хранить пароли в хранилище паролей защищённым мастер-паролем.

## 3.2 Выбор средств реализации.

С учётом сформулированных в части 3.1 требований, мною были выбраны следующие средства реализации – Microsoft Visual Studio Community Edition включающую в себя средства для разработки на ООП языке C#, с установленным дополнением Xamarin и пакетом разработки Xamarin.Android (Рисунок 17).

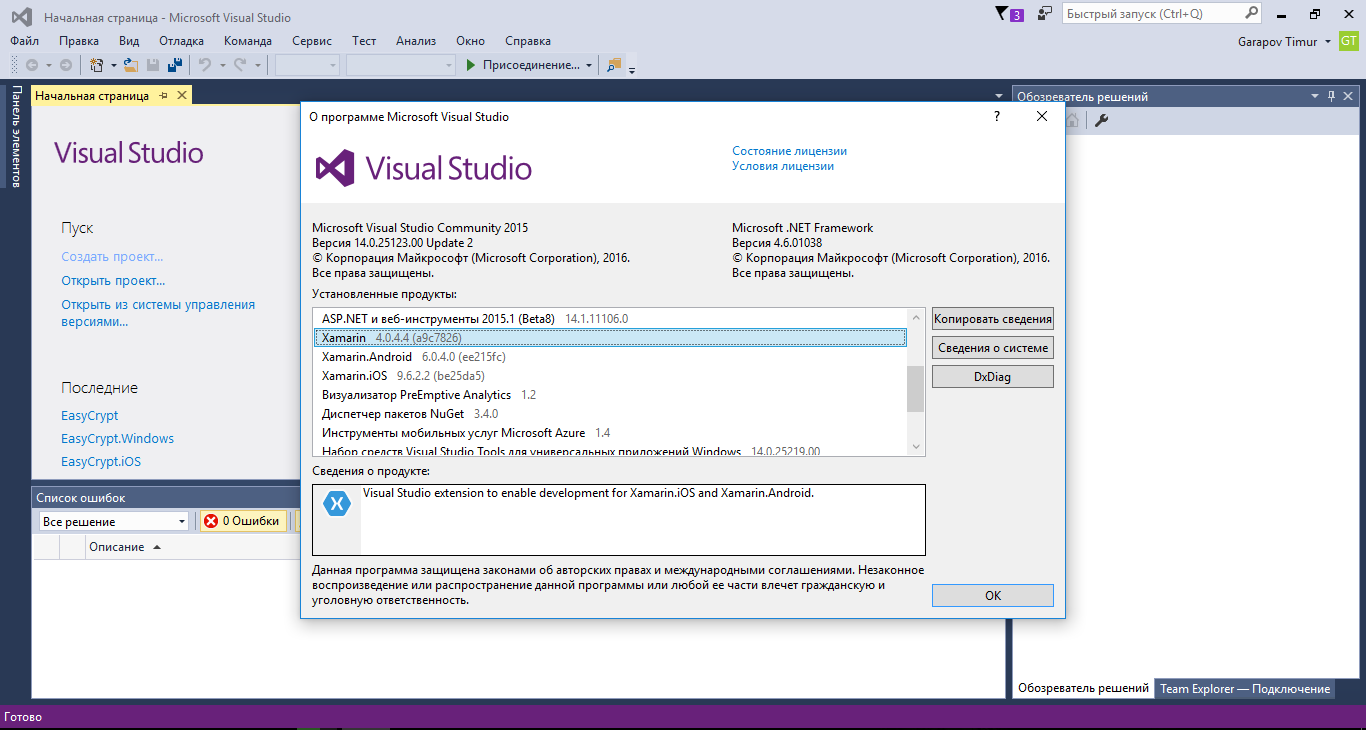


Рисунок 17

Microsoft Visual Studio – это IDE разработанная компанией Microsoft, и поддерживающая множество языков программирования. Однако в данной работе потребуется лишь язык C#, а большую важность для осуществления работы представляет пакет технологий Xamarin, в частности – технология под названием Xamarin Forms.

Следующие факторы послужили причиной выбора связки из языка C# и технологии Xamarin для разработки приложения :

1. Кроссплатформенность.

Xamarin Forms – это технология, позволяющая писать программный код, который возможно скомпилировать сразу под множество платформ. Используя данную технологию можно достичь высокого процента повторного использования кода, так как лишь некоторые специфичные функции, требуют индивидуального написания под конкретную платформу. Пример такого функционала - взаимодействие с файловой системой, которое реализуется разными методами на различных ОС.

1. Скорость разработки.

Использование высокоуровнего языка С# в сочетании с готовыми библиотеками функций, предоставляемых технологиями .NET и Xamarin позволяют быстро реализовывать сложные приложения, используя обширную базу готовых решений.

1. Скорость работы.

Использование функций, предоставляемых используемыми библиотеками, позволяет взаимодействовать с мобильными ОС оптимальными методами.

1. Строгая Типизация.

Язык C# является языком со строгой типизацией переменных. Строгая типизация переменных упрощается процесс отладки кода, так как в отличии от динамической типизации исключаются ошибки, которые могут возникнуть при некорректном обращении к переменным.

1. Инструменты тестирования.

Среда разработки Microsoft Visual Studio имеет встроенные средства тестирования приложений, и кроме прочего, для языка C# существует библиотека NUnit, предназначенная для произведения тестирования приложений.

## 3.3 Описание программной реализации. Интерфейс.

Технология Xamarin.Forms позволяет реализовывать общий интерфейс для всех платформ. Для описания интерфейса используется расширяемый язык разметки для приложений XAML. XAML – это основанный на XML язык разметки разработанный Microsoft.

Любой проект, использующий технологию Xamarin.Forms имеет в своём составе два и более под-проекта. Один проект – основной, и в нём находится весь программный код, который будет являться общим для всех производных проектов. Именно в этом проекте находится логика приложения, реализация которой не зависит от конкретной платформы. На рисунке №18 можно увидеть разбиение проекта на под-проекты в текущем решении:

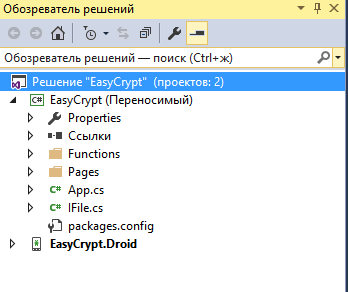


Рисунок 18

Основной проект имеет метку “(Переносимый)”. Основой этого проекта является файл App.cs, именно он является входной точкой приложения.

Весь код проекта, описывающий интерфейс страниц приложения находится в папке Pages. Каждая страница состоит из связки файлов вида [Название страницы].xaml, описывающего внешний вид страницы, и сопутствующего файла [Название страницы].xaml.cs, описывающего логику страницы на языке C#.

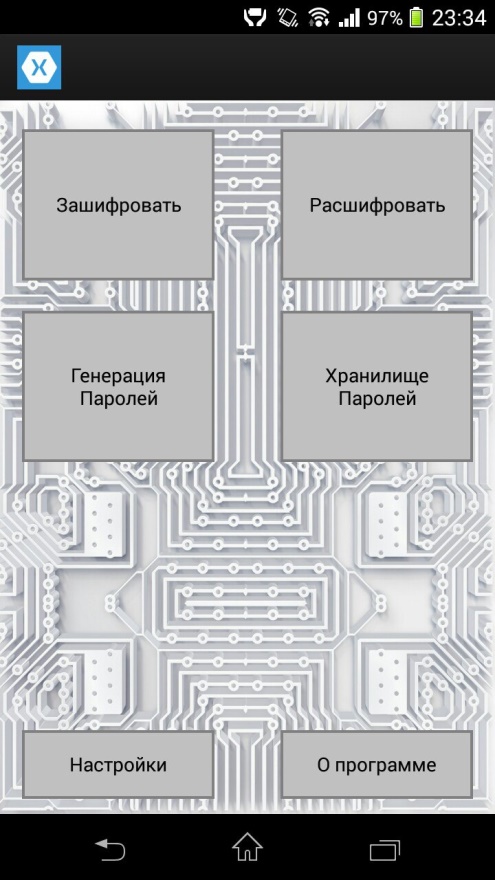
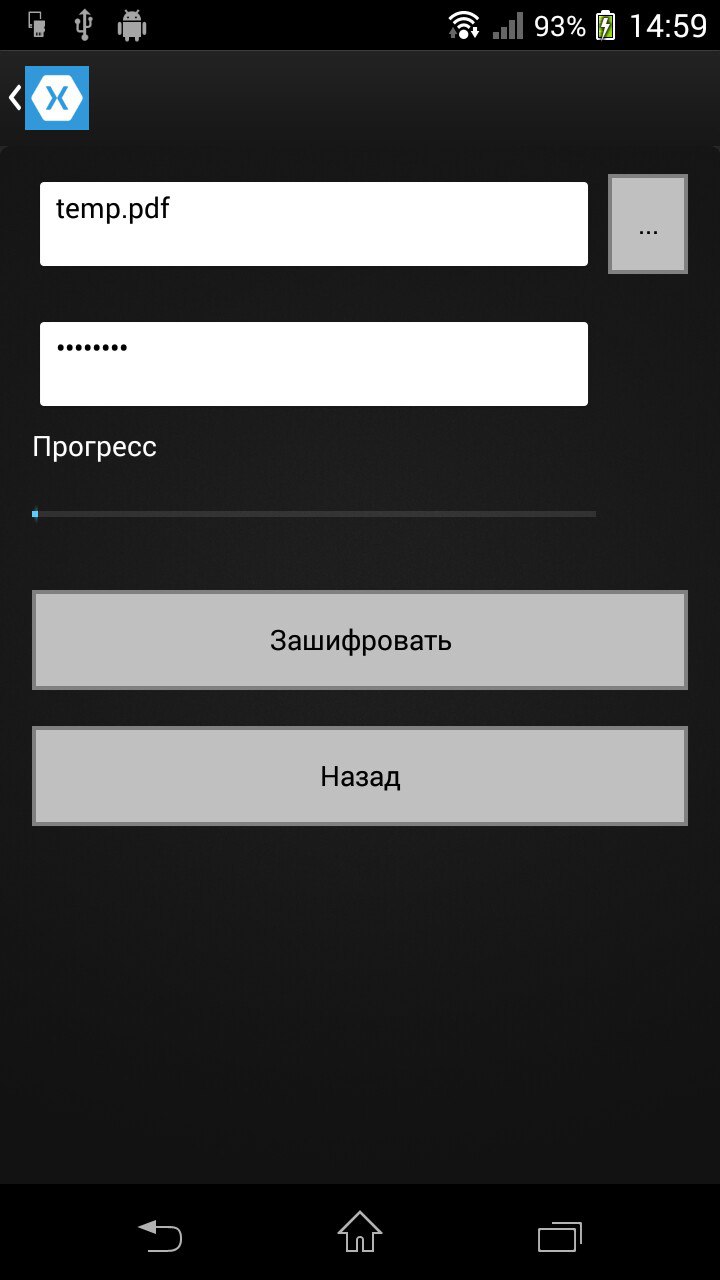
Страница может состоять из различных элементов, расположенных на ней. В приложении для упорядочивания создаваемых элементов на странице используется инструмент с названием Grid – сетка. Сетка позволяет быстро описывать интерфейс, не теряя при этом в качестве.

Пример использования языка Xaml для описания элементов на главной странице, под названием MainPage:

|  |
| --- |
| <Button x:Name="button01" Text="Зашифровать" BorderWidth="2" BorderRadius="1" BorderColor="Gray" BackgroundColor="Silver" TextColor="Black" Clicked="Button\_Click01" Grid.Column="1" Grid.Row="1" />  <Button x:Name="button02" Text="Генерация Паролей" BorderWidth="2" BorderRadius="1" BorderColor="Gray" BackgroundColor="Silver" TextColor="Black" Clicked="Button\_Click02" Grid.Column="1" Grid.Row="3" />  <Button x:Name="button11" Text="Расшифровать" BorderWidth="2" BorderRadius="1" BorderColor="Gray" BackgroundColor="Silver" TextColor="Black" Clicked="Button\_Click11" Grid.Column="3" Grid.Row="1" />  <Button x:Name="button12" Text="Хранилище Паролей" BorderWidth="2" BorderRadius="1" BorderColor="Gray" BackgroundColor="Silver" TextColor="Black" Clicked="Button\_Click12" Grid.Column="3" Grid.Row="3" />  <Button x:Name="button21" Text="Настройки" BorderWidth="2" BorderRadius="1" BorderColor="Gray" BackgroundColor="Silver" TextColor="Black" Grid.Column="1" Grid.Row="6" />  <Button x:Name="button22" Text="О программе" BorderWidth="2" BorderRadius="1" BorderColor="Gray" BackgroundColor="Silver" TextColor="Black" Grid.Column="3" Grid.Row="6" /> |

Фрагмент кода 1

Данная часть кода описывает инициализацию кнопок на главной странице, связывает их нажатие с соответствующими функциями. Пример интерфейса продемонстрирован на рисунке №19 и рисунке №20.

**Рисунок 19**  **Рисунок 20**

На рисунке №19 изображена главная страница приложения, а на рисунке №20 показан внешний вид страницы, на которой происходит выбор файла и осуществляется его шифрование.

Полный код интерфейса будет представлен в рамках кода всего созданного программного обеспечения в Приложении А.

## 3.4 Описание программной реализации. Функционал.

Функционал приложений описывается в соответствующих файлах, являющихся либо описанием классов объектов, используемых в приложении, либо эти функции описаны в файле Functions.cs, описывающих класс, предоставляющий объектам доступ к статическим методам, обеспечивающим функциональность приложения.

Список некоторых методов, используемых в приложении:

* DoAes – функция, принимающая на вход адрес файла, путь к выходной папке, куда следует записать результирующий файл, и пароль, используемый для генерации ключа шифрования;
* UndoAes – функция, обратная предыдущей. Получает на вход адрес зашифрованного файла, путь к результирующей папке, и пароль, используемый для генерации ключа;
* GetPath – функция, вызывающая окно выбора файла и возвращающая упорядоченный адрес результирующего файла;
* GetPassword – функция, принимающая на вход длину требуемого пароля и осуществляющая его генерацию. Возвращает строку, представляющую собой сгенерированный пароль;
* GetMainPage – Функция, при вызове возвращающая новый объект типа Page, используется при старте приложения для получения стартовой страницы.

Подробнее расскажу о функции DoAes(). Эта функция работает следующим образом:

1. Функция при вызове получает адрес файла и обращается к нему, инициируя побайтовый доступ к файлу.
2. Создаётся переменная, в которую записывается хэш от введённого пароля, который будет использоваться в качестве ключа шифрования. Хэш генерируется при помощи алгоритма SHA-256, используя встроенную в .Net реализация данного алгоритма.
3. Исходный файл разбивается на части длиной в 128 бит(16 байт).
4. Создаётся объект класса AesManaged встроенного в .Net . Данный объект инициализируется при помощи созданного хэша.
5. На вход в данный объект, используя потоки данных MemoryStream и CryptoStream, подаются данные блоками по 128 бит.
6. Полученные в предыдущей функции блоки записываются в новый файл, в том-же порядке, в каком они были полученны из исходного файла.
7. Функция DoAes() по окончании возвращает true в случае, если шифрование прошло успешно.

Для навигации и перемещения между страницами была создана группа функций, вызываемая при нажатии соответствующих кнопок. Название функций имеет вид Goto[название страницы]. Данные функции работают используя функции навигации:

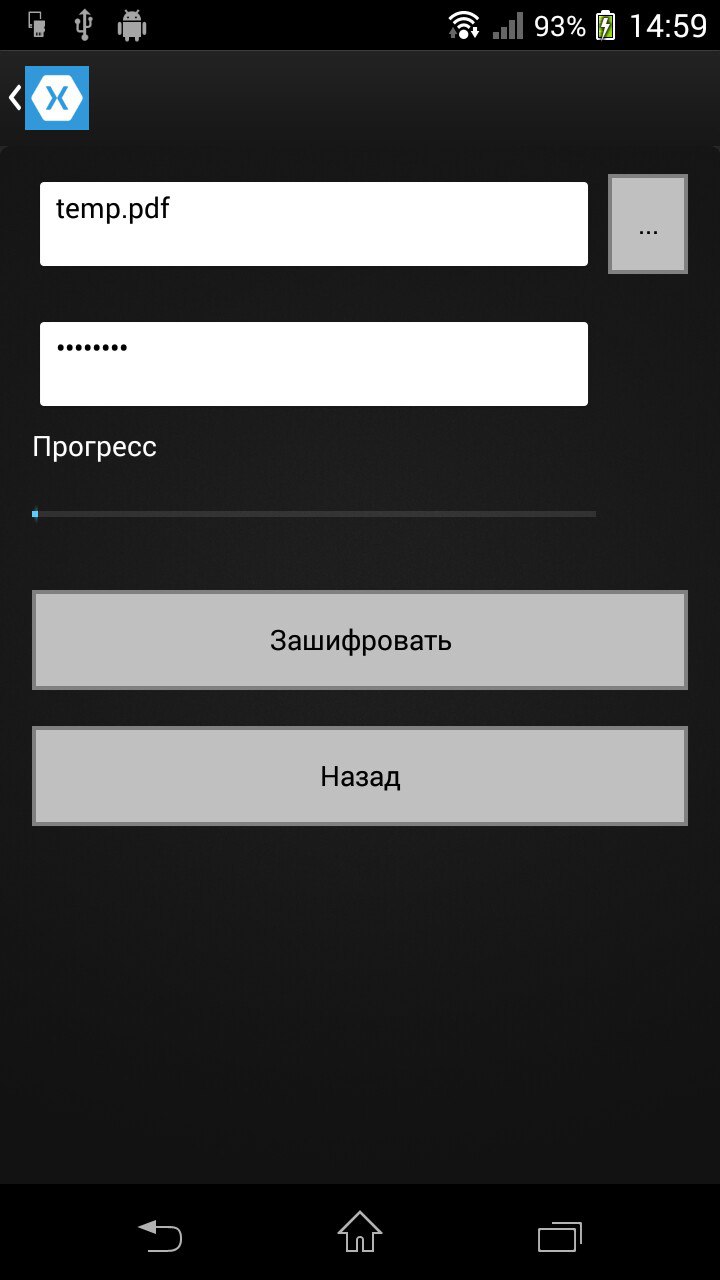
* Navigation.PushAsync();
* Navigation.PopAsync().

Данные функции предоставляют доступ к работе с навигационным стеком, и позволяют записывать или изымать страницы из этого стека.

## 3.5 Визуальное представление приложения и примеры работы.

### 3.5.1 Осуществление шифрования.

1. Запускаем приложение.
2. Выбираем пункт меню – шифрование.
3. Нажимаем на кнопку “[…]” и выбираем файл, который требуется зашифровать.
4. Вводим пароль.
5. Нажимаем кнопку “Зашифровать” (Рисунок 21).



Рисунок

1. Ждём сообщения об успешном окончании операции или сообщения об ошибке.

### 3.5.2 Осуществление расшифровывания.

1. Запускаем приложение.
2. Выбираем пункт меню – расшифровать.
3. Нажимаем кнопку “[…]”.
4. Нажимаем кнопку “Расшифровать”.
5. Вводим пароль, использованный для зашифровывания.
6. Ждём сообщения об успешном окончании операции или .сообщения об ошибке.

### 3.5.3 Генерация Пароля.

1. Запускаем приложение.
2. Выбираем пункт меню – Генерация Паролей.
3. Вводим требуемое количество символов, по умолчанию – 8.
4. Нажимаем кнопку “Генерировать”
5. Получаем пароль.
6. ЗАКЛЮЧЕНИЕ

За последние годы распространение мобильных устройств происходило лавинообразными темпами. На территории РФ у каждого третьего жителя страны на руках имеется хотя-бы одно мобильное устройство. Количество мобильных устройств в мире и активность их использования говорит об актуальности выбранной темы.

В рамках обзора предметной области были проанализированы существующие мобильные операционные системы, приведены данные о частоте их использования, выбраны приоритетные для разработки платформы.

В ходе разработки требований к программному обеспечению были рассмотрены и проанализированы потенциальные сценарии использования создаваемого приложения, произведена оценка требований, накладываемых законодательством РФ в рамках моделируемых сценариев применения создаваемого ПО.

В ходе проектирования создаваемого приложения были рассмотрены особенности практического применения шифрования. Был произведён анализ популярных алгоритмов шифрования, изучены исследования экспертов, посвящённые рассмотренным алгоритмам. На основании мнений экспертов и данных об алгоритмах, представленных в их документации, были собраны и сопоставлены количественные и качественные характеристики алгоритмов, данные об их практическом применении. Путём сравнения характеристик был выбран алгоритм, являющийся лучшим по соотношению характеристик - AES.

В данной работе было проведено сравнение существующих решений по защите данных на мобильных устройствах. Решения были оценены с точки зрения удобства пользователей, широты функционала и обеспечения безопасности защищаемых данных. Всесторонне исследовав представленные приложения были скорректированы требования, предъявляемые создаваемому приложению, позволяя избежать типовых ошибок разработчиков.

В процессе практической реализации программного обеспечения были получены навыки практического применения криптографии, закреплено умение реализации сложных алгоритмов на языке программирования высокого уровня. Применение технологических новинок и сторонних библиотек позволило реализовать приложение в короткие сроки, не потеряв при этом в его функциональности. Использование язык XAML для описания пользовательского интерфейса позволило добиться удобства интерфейса без значительных временных затрат на его проектирование. Созданное ПО выполняет свои функции, при этом большая часть написанного кода может использоваться при создании приложений для других операционных систем, так как программа написана с использованием кросс-платформенных технологий.

Таким образом цель и задачи данной дипломной работы выполнены — программное обеспечение для шифрования пользовательских данных разработано и может быть использовано для защиты конфиденциальной информации.