# 2.Анализ исходных данных

Перед тем, как начать решение поставленной задачи – 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 задачи – защита 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

(Ниже-возможно исключённый абзац)

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 могут 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.

## 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 так-же будет произведён их сравнительный анализ.

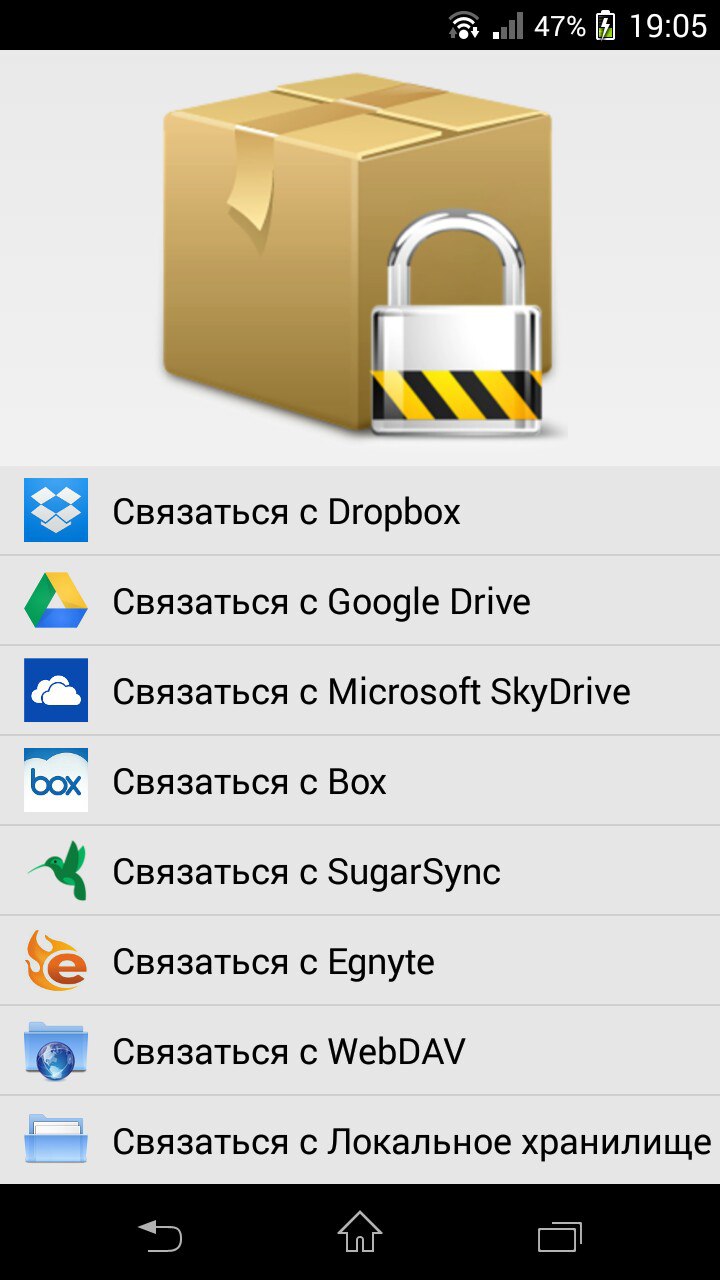
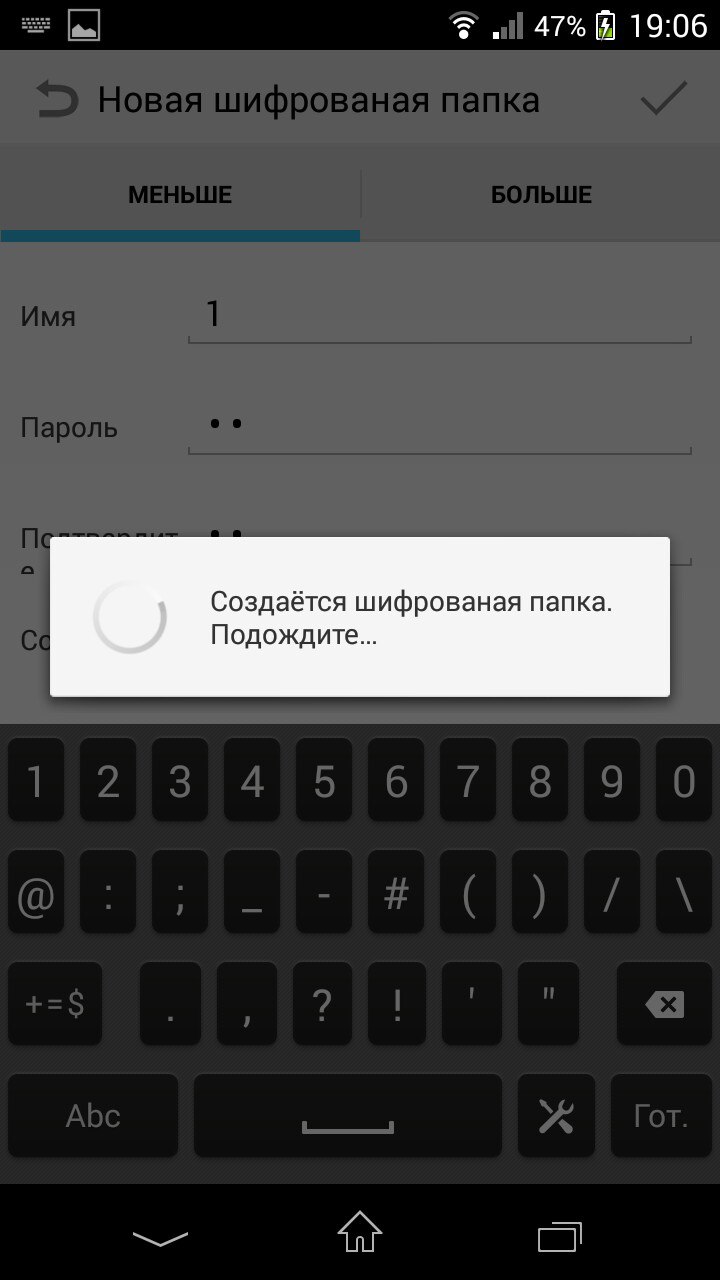
### Boxcryptor Classic

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 рисунках №**(№ РИСУНКОВ)**

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.

(<https://play.google.com/store/apps/details?id=com.boxcryptor.android>)

### 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.

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.

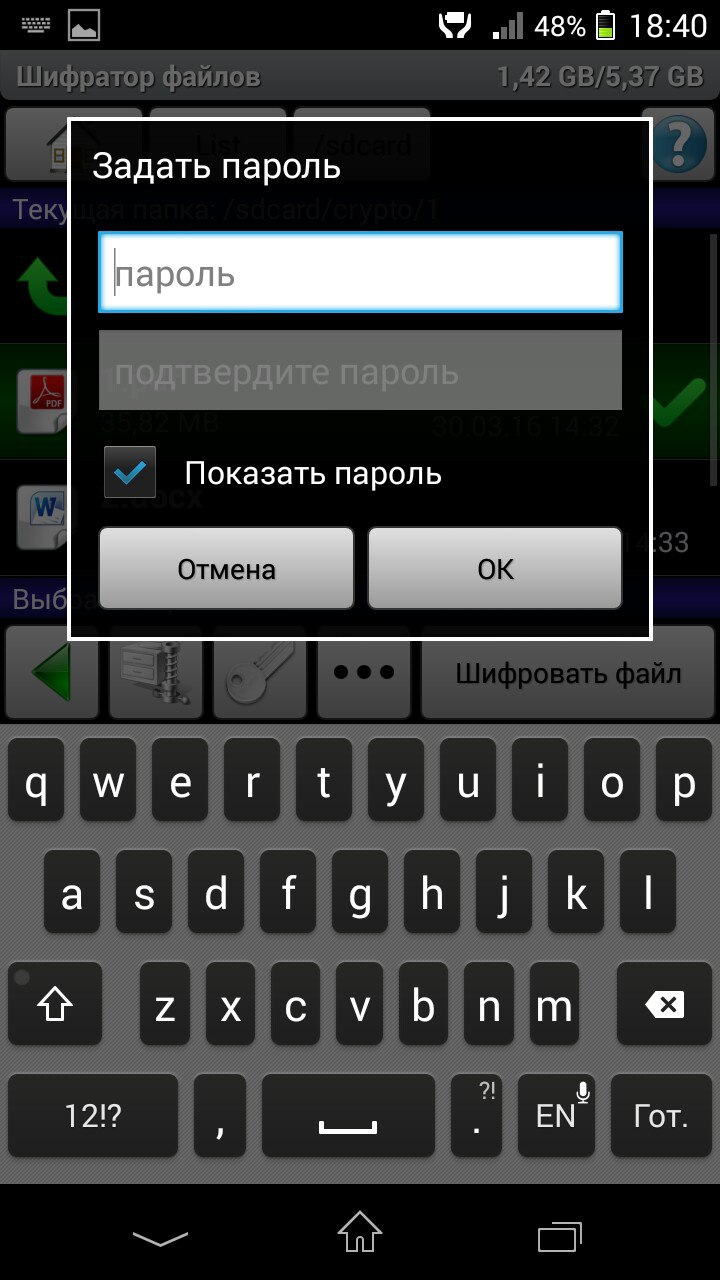
### SSE Шифрование

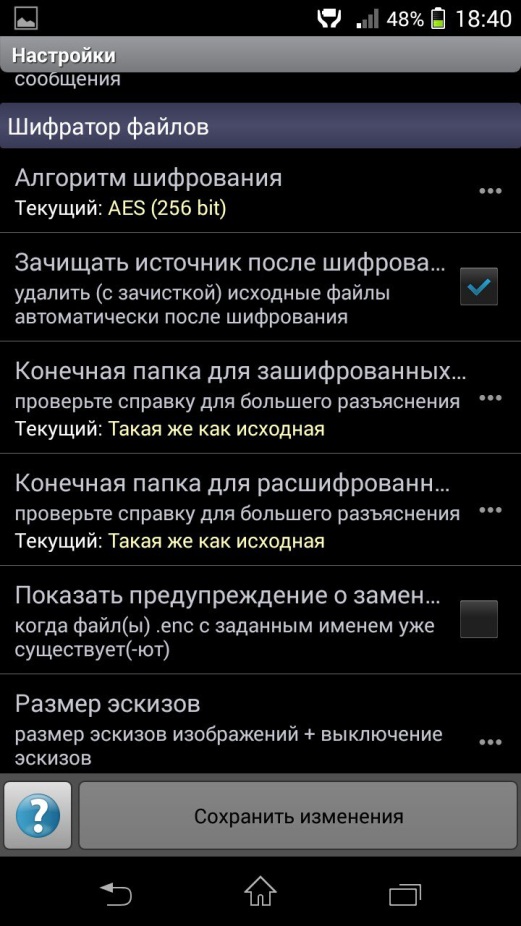
Данное приложение сочетает в себе функции менеджера паролей, шифратора текста и шифратора файлов. В нашем случае целесообразно подробнее разобрать возможностиданного приложения в области шифрования файлов.

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 рисунках №**(№ РИСУНКОВ)**



Плюсы:

* Кроссплатформенное.
* Наличие нескольких алгоритмов шифрования.
* Затирание файлов.
* Много возможностей в одном приложении.
* Частота обновлений.

Минусы:

* Полный функционал доступен только при покупке платной версии.

### Анализ и сравнение существующих решений

В следующей таблице представлены данные о функциональности протестированных приложений.

Таблица 1 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 | 2 | - | + | + | - | Постоянно |
| SSE Шифро-вание | 6 в бесплатной версии,  8 в платной | + | + | + | + | Постоянно |

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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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.

**Алгоритм ГОСТ 28147-89**

**Структура алгоритма:**

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



**Рисунок 2.3.1** 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 **(№ РИСУНКА**):

К0*,* К1, К2, К3,К4, К5, К6*,* К7.



**Рисунок 2 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 (рис. 3.3) 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 3 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 **(№РИСУНКА)**. Первый же блок в данном режиме генерируется полностью аналогично предыдущему.



**Рисунок 4 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 1)приведены краткие сведения по некоторым опубликованным работам, посвящённым криптоатакам на ГОСТ 28147-89.

**Таблица 1**

|  |  |  |  |
| --- | --- | --- | --- |
| 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.

**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 5 Таблица Замен AES)



**Рисунок 5 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:

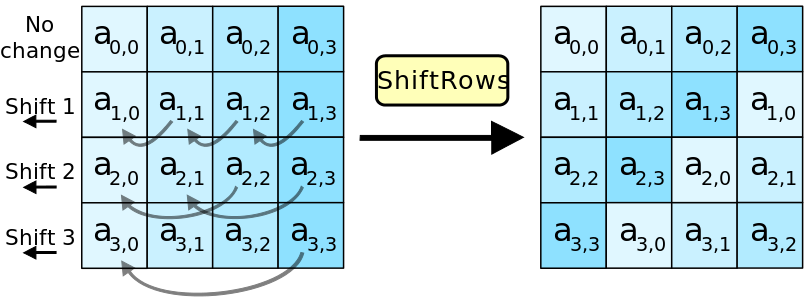
* вычисление мультипликативной обратной величины от входного значения в конечном поле GF(28); обратной величиной от 0 является 0;
* выходное значение b вычисляется следующим образом:

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 6 ShiftRows). Сдвиг i-й строки массива (для i = 1,2,3) производится на i байтов.

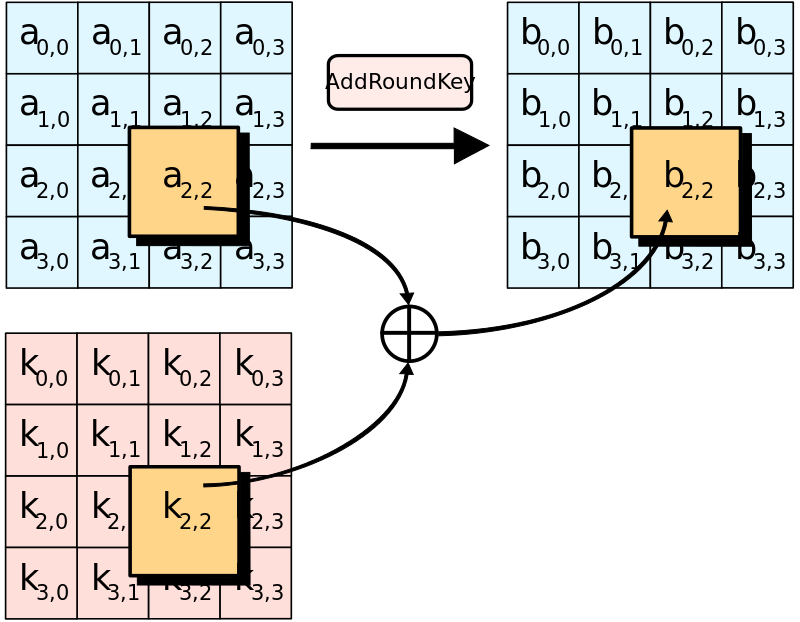


**Рисунок 6 ShiftRows**

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 7 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 7 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 2).

**Таблица 2**

|  |  |
| --- | --- |
| 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 8 ).



**Рисунок 8 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:

* 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.
* 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 3).

**Таблица 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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.

**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.

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:

Матрица MDS представлена в таблице 4 :

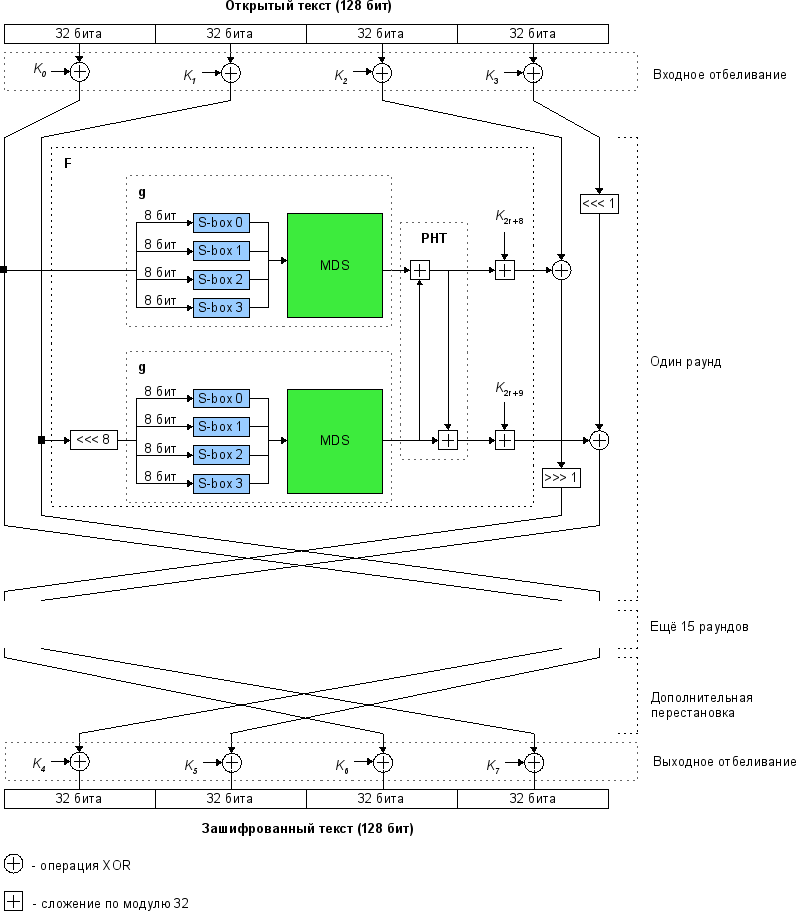
**Таблица 4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 9



**Рисунок 9 Алгоритм TwoFish**

**Процедура расширения ключа**

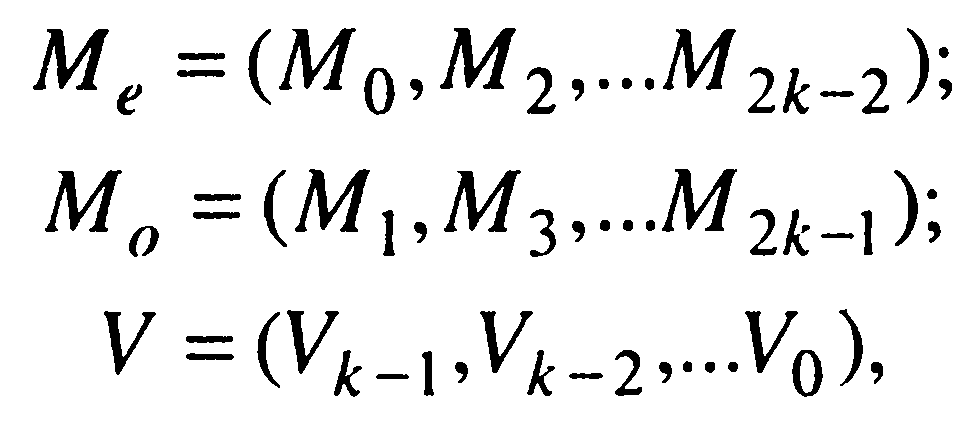
Twofish, так-же, как и Rijndael, может работать с ключами длиной 128, 192 и 256 бит. Из исходного ключа генерируется 40 4 байтных подключей, первые 8 из них используются только в операциях входного и выходного отбеливания, а остальные 32 — в раундах шифрования, по два ключа на раунд. Отличием Twofish 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:

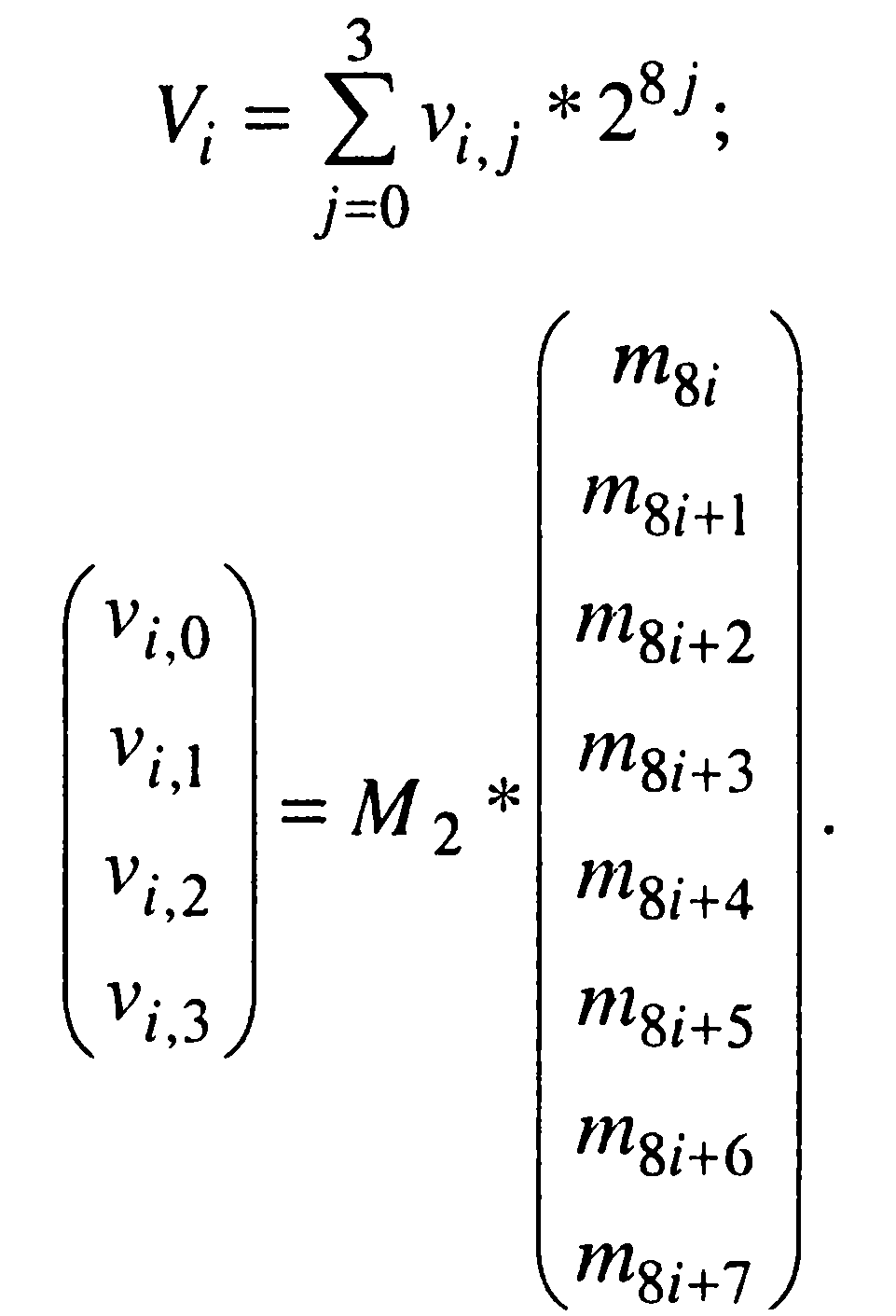
Сначала производится предварительная обработка ключа, включающая в се­бя следующие шаги:

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:



Где V вычисляется следующим образом:



Матрица представлена в Таблице 5

**Таблица 5**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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:

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

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 выполения:

1. 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 .

2. 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.*

3. и 4. 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.

5. Как и на предыдущих шагах, применяются замены q0 и q1, после чего выполняется следующее преобразование:

;

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.

**Сравнение Алгоритмов.**

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

* 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.
* 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
* 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 Таблице 4.

**Таблица 6**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 № 6, следует также сравнить параметры, являющиеся субъективным мнением экспертов и пользователей представленных алгоритмов, а также параметры, которые нельзя оценить с высокой степенью точности (Таблица 7).

**Таблица 7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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(КГБ) |
| 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 таже 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.