МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Систем сбора и обработки данных

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

***Ермошенко Павла Андреевича***

(фамилия, имя, отчество автора)

***Разработка клиентского программного обеспечения для обмена данными на основе ассиметричного шифрования.***

|  |  |
| --- | --- |
| Направление подготовки | ***230200 Информационные системы*** |

|  |  |
| --- | --- |
| **Руководитель**  ***Воронов В.В.***  (фамилия, И.О.)  ***Старший преподаватель кафедры ССОД***  (уч. степень, уч. звание)  (подпись, дата) | **Автор**  ***Соснов М.Е.***  (фамилия, И.О.)  ***АВТ,*** ***АТ-03***  (факультет, группа)  (подпись, дата) |

Новосибирск, 2014 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Систем сбора и обработки данных

|  |  |
| --- | --- |
|  | Утверждаю  Зав. кафедрой  (подпись)  Белик Д.В.  (фамилия, инициалы)  « »   г. |

**ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

студенту ***Соснову Максиму Евгеньевичу***

(фамилия, имя, отчество студента)

|  |  |
| --- | --- |
| Направление подготовки | ***230200 Информационные системы*** |

***Факультет автоматики и вычислительной техники***

Тема  ***Разработка клиентского программного обеспечения для обмена данными на основе ассиметричного шифрования.***

Исходные данные (или цель работы):

*Разработка клиентского программного обеспечения, позволяющего производить обмен данными по защищенному протоколу через незащищенный канал. Разработка протокола, основанного на шифровании, обеспечивающего конфиденциальность и невозможность подмены данных.*

Структурные части работы:

1. Изучение предметной области:
   1. Существующие алгоритмы шифрования. Преимущества/недостатки.
   2. Известные атаки на алгоритмы. Способы защиты от известных атак.
2. Выбор параметров будущей системы. Определение основных принципов.
3. Создание протокола с учетом существующих методов шифрования и атак на них.
4. Выбор инструментов для создания клиентского приложения.
5. Создание клиентского приложения. Тестирование приложения и его компонентов.
6. Описание полученной технологии. Создание ВКР.

**План-график выполнения работы**

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование этапа | Планируемые сроки выполнения |
| 1 | Анализ существующих алгоритмов шифрования, их преимуществ/недостатков а также известных атак. Анализ способов защиты от известных атак. | 12.02.2014 – 01.04.2014 |
| 2 | Выбор параметров будущей системы. Определение основных принципов. | 05.04.2014 – 15.05.2014 |
| 3 | Создание протокола с учетом существующих методов шифрования и атак на них. | 20.05.2014 – 26.05.2014 |
| 4 | Выбор инструментов для создания клиентского приложения. | 26.05.2014 – 27.05.2014 |
| 5 | Создание клиентского приложения. Тестирование приложения и его компонентов. | 27.05.2014 – 09.06.2014 |
| 6 | Описание полученной технологии. Создание ВКР. | 10.06.2014 – 20.06.2014 |

Задание согласовано и принято к исполнению.

|  |  |
| --- | --- |
| **Руководитель**  ***Воронов В.В.***  (фамилия, И.О.)  ***кафедры ССОД***  (уч. степень, уч. звание)    ***(***подпись, дата) | **Автор**  ***Соснов М.Е.***  (фамилия, И.О.)  ***АВТ, АТ-03***  (факультет, группа)    ***(***подпись, дата) |

Тема утверждена приказом по НГТУ №***6550/2***  от « ***4*** »***декабря***  ***2013*** г.

(подпись секретаря экзаменационной комиссии по защите ВКР, дата)

(фамилия, инициалы секретаря экзаменационной комиссии по защите ВКР)

# Реферат

Данная выпускная квалификационная работа содержит quint8 n; страниц, quint8 m иллюстраций и quint8 o; таблиц. В процессе создания этой работы были использованы следующие источники:

* Раздел Information Security на Stack Exchange: <http://security.stackexchange.com/>
* Стандарт RFC5114 описывающий выбор параметров открытых ключей для алгоритма Диффи-Хеллмана: <http://www.ietf.org/rfc/rfc5114.txt>
* Спецификация Rijndael (победителя конкурса на Advanced Encryption Standard – AES):  
  <http://en.wikipedia.org/wiki/Advanced_Encryption_Standard>
* Спецификация PKI – Public Key Infrastructure:  
  <http://en.wikipedia.org/wiki/Public_key_infrastructure>
* Описание схемы шифрования RSA:  
  <http://en.wikipedia.org/wiki/RSA_(cryptosystem)>
* Описание схемы обмена ключами Диффи-Хеллмана:  
  <http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange>

Целью этой работы является создание защищенного протокола на основе SSL/TLS. В отличие от TLS (который применяется в основном при передаче данных в Internet – веб страниц), данный протокол предназначен для применения в более узких областях. Небольшая распространенность является его преимуществом по отношению к TLS – не требуется продолжительная поддержка более старых версий (которая обычно влечет к дополнительным уязвимостям).

Объектом исследования являются криптографические алгоритмы и их реализации. Анализ существующих атак и методов защиты от них позволяет повысить устойчивость полученного протокола.

В ходе выполнения этой работы был разработан протокол защищенного обмена данных по незащищенному каналу. Протокол берет свой дизайн из широко известного и популярного протокола TLS (бывший SSL). В его основе лежат такие криптографические алгоритмы, как RSA, Rijndael (AES) и технология обмена ключами Диффи-Хеллмана. После разработки протокола он был реализован в клиентском приложении написанном на C++/QT(возможно развертывание на всех трех основных операционных системах: Windows NT, Linux, OSX). Также в совместном проекте существует совместимое клиентское приложение. Протокол не ограничен “банковской” сферой применения и созданные приложения являются лишь примерами его применения.

Криптографическая защита данных является очень популярной областью исследования (множество людей задействовано по всему миру ввиду ценности информации в современном мире) и любые полученные результаты требуют постоянного обновления, чтобы оставаться достоверными. *Потому* ***блять*** *написание бумажек в 40 листов is a waste of time.*

Созданный протокол не привязан к типу передачи данных и требует лишь наличие механизма, отвечающего за доставку данных через сеть (или другую используемую среду), такого как TCP (Transmission Control Protocol – один из двух основных протоколов транспортного уровня используемых в Internet).

В данную работу не входит создание физического устройства, в связи с чем ее объем несколько ниже среднего.

Ключевые слова: **RSA**, **DH**, **AES**, **TLS**, **криптография**, **шифрование**.

**Содержание**

[Реферат 5](#_Toc390089128)

[Введение 9](#_Toc390089129)

[Глава 1: Описание предметной области 10](#_Toc390089130)

[Глава 2: Описание нового протокола 11](#_Toc390089131)

[Глава 3: Описание разработанного приложения 12](#_Toc390089132)

[Заключение 13](#_Toc390089133)

# Введение

С появлением сети интернет открылась возможность передавать данные с одного конца мира в другой за короткое время. Однако сеть интернет не является защищенным каналом передачи данных исходя из его строения. При передаче данных пакет, следуя от отправителя к получателю, проходит множество узлов, каждый из которых при должном желании может посмотреть содержимое пакета.

Для конфиденциальности передачи данных используются различные алгоритмы, основанные на шифровании данных и электронных подписях. Например, существуют алгоритмы SSL и TLS, использующие асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений. Эти алгоритмы повсеместно используются в различных приложениях, использующих сеть Интернет (Веб-браузеры, приложения для работы с электронной почтой, мессенджеры, IP-телефония и тд). Существует множество бесплатных библиотек с открытым исходным кодом для установления безопасного соединения в открытом канале передачи данных по технологиям SSL/TLS, но, как показывают события последних месяцев библиотеки с открытым исходным кодом не всегда справляются со своей работой. Например, в апреле 2014го года была найдена уязвимость в библиотеке OpenSSL, названной “Heartbleed”, которая позволяла получить стороннему человеку расшифрованные переданные данные. Спустя 2 месяца была найдена еще одна уязвимость. Надо заметить, что эти уязвимости были внесены с функционалом в начале 2012-го года. Таким образом, начиная с 2012го года информация, передаваемая с помощью OpenSSL могла быть получена злоумышленниками, знающими об уязвимости. Например, Агентство Национальной Безопасности США использовало эту уязвимость в своих целях 2 года.

Крупным информационным системам, которым нужно обеспечивать конфиденциальность передачи данных, желательно использовать качественное программное обеспечение для этих целей. Например, крупный банк может позволить себе потратиться на разработку своего решения создания защищенного канала передачи данных в среде интернет.

Целями бакалаврской работы являются:

1. Создание протокола передачи данных по защищенному каналу по типу SSL/TLS, используя RSA, AES шифрования и обмен ключами по алгоритму Diffie-Hellman.
2. Создание клиентского приложения, способного установить защищенное соединение с сервером, по созданному протоколу.

# Глава 1: Описание предметной области

## 1.1. Исследование существующих методов шифрования, их достоинств и недостатков

### RSA.

#### Определение

**RSA** (аббревиатура от фамилий Rivest, Shamir и Adleman) — асимметричный криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Криптографические системы с открытым ключом используют так называемые односторонние функции, которые обладают следующим свойством:

Если известно x, то f(x) вычислить относительно просто

Если известно y=f(x), то для вычисления x нет простого (эффективного) пути.

Под односторонностью понимается не теоретическая однонаправленность, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства, за обозримый интервал времени.

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования за разумное время (обратной операции) необходимо уметь вычислять функцию Эйлера от данного большого числа, для чего необходимо знать разложения числа на простые множители.

#### Описание алгоритма

##### Создание ключей

RSA-ключи генерируются следующим образом:

* 1. Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое).
  2. Вычисляется их произведение n=p\cdot q, которое называется *модулем*.
  3. Вычисляется значение функции Эйлера от числа n: \varphi(n) = (p-1)(q-1).
  4. Выбирается целое число e (1 < e < \varphi(n)),взаимно простое со значением функции \varphi(n). Обычно, в качестве e берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые числа Ферма 17, 257 или 65537.
  5. Число e называется открытойэкспонентой (англ. *public exponent*)

Время, необходимое для шифрования с использованием быстрого возведения в степень, пропорционально числу единичных бит в e. Слишком малые значения e, например 3, потенциально могут ослабить безопасность схемы RSA.

* 1. Вычисляется число d, мультипликативно обратное к числу e по модулю \varphi(n), то есть число, удовлетворяющее условию: d\cdot e \equiv 1 \mod {\varphi(n)}.
  2. Число d называется *секретной экспонентой*. Обычно, оно вычисляется при помощи расширенного алгоритма Евклида.

Пара \left\{ e, n \right\} публикуется в качестве *открытого ключа RSA* (англ. *RSA public key*). Пара \left\{ d, n \right\} играет роль *закрытого ключа RSA* (англ. *RSA private key*) и держится в секрете.

##### Шифрование и дешифрование

Шифрование сообщения m открытым ключом:  c \equiv m^e \pmod{n} , при этом *0 ≤ m < n*

Дешифрование сообщения c закрытым ключом:  m \equiv c^d \pmod{n} 

Пример алгоритма:

* 1. Выбрать простые числа p и q
  2. Вычислить n = p \* q
  3. Вычислить m = (p - 1) \* (q - 1)
  4. Выбрать число d взаимно простое с m
  5. Выбрать число e так, чтобы e \* d = 1 (mod m)
  6. Открытый ключ = (n,e). Закрытый ключ = (n,d)
  7. Шифрование: b = ae (mod n)
  8. Дешифровка: a = bd (mod n)

Практический пример:

* 1. Выбрали числа: p=61, q=53
  2. Вычисляем n=3233
  3. Вычисляем m`=3120
  4. Выбираем d=17
  5. Выбираем e=2753
  6. Выбираем сообщение m = 65
  7. Шифруем сообщение m c = 65^{17} \; \operatorname{mod}\; 3233 = 2790 
  8. Дешифруем сообщение m = 2790^{2753} \; \operatorname{mod}\; 3233 = 65 

### Elliptic Curve

#### Определение

Эллиптическая криптография — раздел [криптографии](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F), который изучает асимметричные криптосистемы, основанные на [эллиптических кривых](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F) над конечными полями. Основное преимущество эллиптической криптографии заключается в том, что на сегодняшний день неизвестно существование субэкспоненциальных [алгоритмов](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B) решения задачи [дискретного логарифмирования](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

При использовании алгоритмов на [эллиптических кривых](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F) полагается, что не существует субэкспоненциальных [алгоритмов](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B) для решения задачи [дискретного логарифмирования](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) в группах их точек. При этом [порядок группы](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D1%8B) точек эллиптической кривой определяет сложность задачи. Считается, что для достижения такого же уровня безопасности как и в RSA требуются группы меньших порядков, что уменьшает затраты на хранение и передачу информации.

#### Алгоритм

##### Параметры:

Пусть P - точка эллиптической кривой E над полем G(p), имеющая порядок n. Тогда циклическая подгруппа E порожденная точкой P будет состоять из точек {O, P,2P,3P, . . .,(n?1)P}. Характеристика поля p, уравнение эллиптической кривой E, точка P и ее порядок n являются параметрами кривой. Секретным ключом является число d, которое выбирается случайно из интервала [1, n-1], а открытым ключом является точка Q = dP. Задача вычисления d по известным параметрам кривой и точке Q называется проблемой дискретного логарифма в группе точек эллиптической кривой (ECDLP).

##### Генерация ключей:

1. Вход: Параметры кривой (p, E, P, n).
2. Выход: Открытый ключ Q и секретный ключ d.
3. Алгоритм:
   1. Выбрать d из [1,n-1].
   2. Вычислить Q = dP.
   3. Вернуть: (Q,d)

##### Шифрование и дешифрование

Открытый текст m представляется в виде точки *M*, а затем шифруется путем сложения с точкой *kQ*. Отправитель передает точку*C1 = kP* и *C2 = M + kQ* получателю, который использует свой секретный ключ *d* для вычисления *dC1 = d(kP) = kQ* и затем восстанавливает *M = C2 - k*q. Злоумышленник, желающий восстановить *M*, должен вычислить *kQ*. Проблема вычисления *kQ* при знании параметров кривой, *Q* и *C1 = kP*, является аналогом проблемы Диффи-Хеллмана для эллиптических кривых. Ниже предоставленалгоритм шифрования.

###### Шифрование

1. Вход: Параметры кривой (p, E, P,n), открытый ключ Q, текст m.
2. Вывод: Криптограмма (C1,C2).
3. Алгоритм:
   1. Представить сообщение m в виде точки M кривой E(Fp).
   2. Выбрать k из R [1,n - 1].
   3. . Вычислить C1 = kP.
   4. . Вычислить C2 = M +kQ.
   5. Вернуть: (C1,C2).

###### Дешифрование

1. **Вход**: Параметры кривой *(p, E, P,n)*, секретный ключ *d*, криптограмма *(C1,C2)*.
2. **Выход**: Исходный текст *m*.
3. **Алгоритм**:
   1. Вычислить *M = C2 - dC1*,
   2. Вычислить *m* из *M*.
4. **Вернуть**: (*m*).

### AES

#### Описание

**Advanced Encryption Standard** (**AES**), также известный как Rijndael (произносится [rɛindaːl] (Рэндал) ) — симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурс. Этот алгоритм хорошо проанализирован и сейчас широко используется. Национальный институт стандартов и технологий США (англ. National Institute of Standards and Technology, NIST) опубликовал спецификацию AES 26 ноября 2001 года после пятилетнего периода, в ходе которого были созданы и оценены 15 кандидатур. 26 мая 2002 года AES был объявлен стандартом шифрования. По состоянию на 2009 год AES является одним из самых распространённых алгоритмов симметричного шифрования. Поддержка AES (и только его) введена фирмой Intel в семейство процессоров x86 начиная с Intel Core i7-980X Extreme Edition, а затем на процессорах Sandy Bridge.

#### Алгоритм шифрования

##### Определения

1. State — промежуточный результат шифрования, который может быть представлен как прямоугольный массив байтов имеющий 4 строки и Nb колонок. Каждая ячейка State содержит значение размером в 1 байт
2. Nb — число столбцов (32-х битных слов), составляющих State. Для стандарта регламентировано Nb = 4
3. Nk — длина ключа в 32-х битных словах. Для AES, Nk = 4, 6, 8.
4. Nr — количество раундов шифрования. В зависимости от длины ключа, Nr = 10, 12 или 14

##### Схема

Алгоритм имеет четыре трансформации, каждая из которых своим образом влияет на состояние State и в конечном итоге приводит к результату: *SubBytes(), ShiftRows(), MixColumns()* и *AddRoundKey()*. Общую схему шифрования можно представить как:



Рисунок 1.1Схема шифрования AES

##### Подготовка данных

В начале заполняется массив State входными значениями по формуле State[r][c] = input[r + 4c], r = 0,1...4; c = 0,1..Nb. То есть по колонкам. За раз шифруется блок размером 16 байт.



Рис.2. Заполнение State.

Алгоритм оперирует байтами, считая их элементами конечного поля или поля Галуа GF(28). Число в скобках — это количество элементов поля или его мощность. Элементами поля GF(28) являются многочлены степени не более 7, которые могут быть заданы строкой своих коэффициентов. Байт очень легко представить в виде многочлена. Например, байту {1,1,1,0,0,0,1,1} соответствует элемент поля 1x7 + 1x6 + 1x5 + 0x4 + 0x3 + 0x2 + 1x1 + 1x0 = 1x7 + 1x6 + 1x5 + x +1. То, что мы работаем с элементами поля, очень важно потому, что это меняет правила операций сложения и умножения. Этого мы коснемся немного позже.

##### SubButes()

Преобразование представляет собой замену каждого байта из State на соответствующий ему из константной таблицы Sbox. Значения элементов Sbox представлены в шестнадцатеричной системе счисления. Сама же таблица получена посредством преобразований уже известного нам поля GF(28)

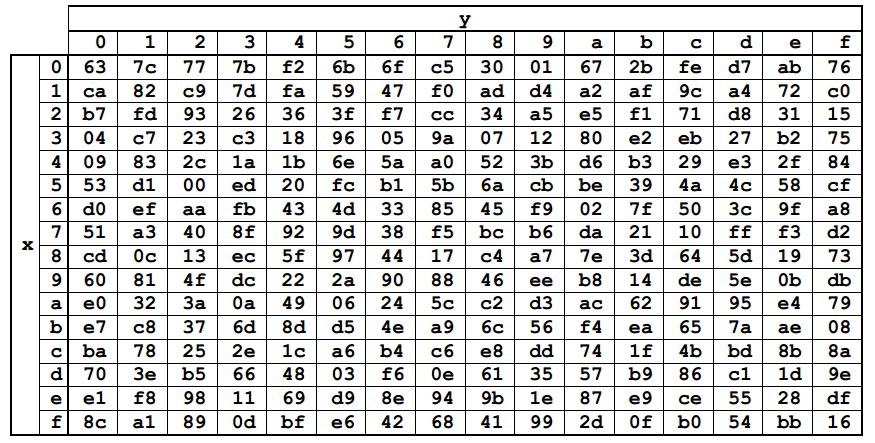


Рис.3. Sbox AES.

Каждый байт из State можно представить как {xy} в шестнадцатеричной системе счисления. Тогда следует заменять его на элемент, стоящий на пересечении строки x и столбца y. Например, {6е} заменится на {9f}, а {15} на {59}

##### ShiftRows()

Простая трансформация. Она выполняет циклический сдвиг влево на 1 элемент для первой строки, на 2 для второй и на 3 для третьей. Нулевая строка не сдвигается.

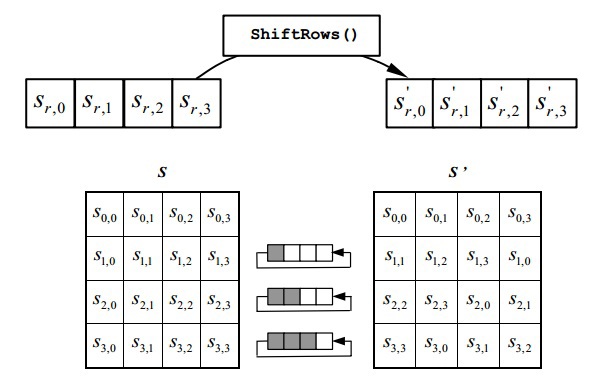


Рис.4. Shift Rows AES.

##### MixColumns()

В рамках этой трансформации каждая колонка в State представляется в виде многочлена и перемножается в поле GF(28) по модулю x4 + 1 с фиксированным многочленом 3x3 + x2 + x + 2. Звучит вроде просто, но малопонятно, как это сделать. Картина становится проще, если посмотреть на эквивалентную матричную запись, предоставленную в официальном документе стандарта:



Рис. 5. Mix Columns AES.

При умножении матриц, значение аij получается как сумма произведений соответствующих элементов i-ой строки первой матрицы и j-ого столбца второй, т. е.

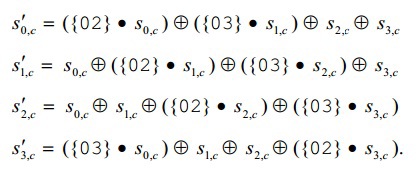


Рис. 6. Mix Columns AES.

Здесь нужно вспомнить о неработоспособности обычных правил сложения и умножения.   
Новые правила:

1. Сложение в поле GF(28) эквивалентно операции XOR
2. Умножение на {01} не меняет умножаемое
3. Умножение на {02} производится по правилу: если умножаемое значение меньше {80}, оно сдвигается влево на 1 бит. Если же умножаемое значение больше или равно {80}, оно сначала сдвигается влево на 1 бит, а затем к результату сдвига применяется операция XOR со значением {1b}. Результат может перескочить за значение {ff}, то есть за границы одного байта. В этом случае нужно вернуть остаток от деления результата на {100}.
4. Умножение на другие константы можно выразить через предыдущие

Естественно, это не общие правила арифметики в конечном поле, но в рамках алгоритма придется умножать на три константы при шифровании и на четыре при дешифровке. MixColumns() вместе с ShiftRows() добавляют диффузию в шифр.

##### AddRoundKey()

Трансформация производит побитовый XOR каждого элемента из State с соответствующим элементом из RoundKey. RoundKey — массив такого же размера, как и State, который строится для каждого раунда на основе секретного ключа функцией KeyExpansion(), которую и рассмотрим далее.

##### KeyExpansion()

Эта вспомогательная трансформация формирует набор раундовых ключей — KeySchedule. KeySchedule представляет собой длинную таблицу, состоящую из Nb\*(Nr + 1) столбцов или (Nr + 1) блоков, каждый из которых равен по размеру State. Первый раундовый ключ заполняется на основе секретного ключа, который вы придумаете, по формуле KeySchedule[r][c] = SecretKey[r + 4c], r = 0,1...4; c = 0,1..Nk.



Рис. 6. KeySchedule для AES 128.

На рисунке 6 изображен макет KeySchedule для AES-128: 11 блоков по 4 колонки. Для других вариаций алгоритма будет соответственно (Nr + 1) блоков по Nb колонок. Теперь нам предстоит заполнить пустые места. Для преобразований опять определена константная таблица — Rcon — значения которой в шестнадцатеричной системе счисления.

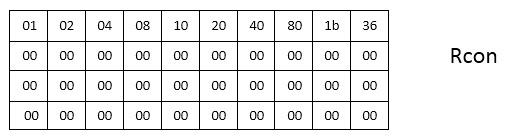


Рис. 7. Rcon AES.

##### Алгоритм дозаполнения KeySchedule:

На каждой итерации работаем с колонкой таблицы. Колонки 0,..,(Nk — 1) уже предварительно заполнены значениями из секретного слова. Начинаем с колонки под номером Nk (в нашем случае с четвертой)

Если номер Wi колонки кратен Nk (в нашем случае каждая четвертая), то берем колонку Wi-1, выполняем над ней циклический сдвиг влево на один элемент, затем все байты колонки заменяем соответствующими из таблицы Sbox, как делали это в SubBytes(). Далее выполняем операцию XOR между колонкой Wi-Nk, измененной Wi-1 и колонкой Rconi/Nk-1. Результат записывается в колонку Wi. Чтобы было немного понагляднее, иллюстрация для i = 4.



Рис. 8. Алгоритм дозаполнения KeySchedule.

Для остальных колонок выполняем XOR между Wi-Nk и Wi-1. Результат записываем в Wi

Эта вспомогательная трансформация является самой объемной по написанию и, наверное, самой сложной, после осознания математики в MixColumns(), в алгоритме. Шифроключ обязан состоять из 4\*Nk элементов (в нашем случае 16). Но так как мы делаем все это для домашнего применения, то вполне вероятно, что придумывать ключ в 16 символов и запоминать его не каждый будет. Поэтому, если на вход поступила строка длиной менее 16, я в KeySchedule дозаношу значения {01} до нормы.

Как было сказано ранее, KeySchedule используется в трансформации AddRoundKey(). В раунде инициализации раундовым ключом будут колонки с номерами 0,..,3, в первом — с номерами 4,..,7 и тд. Вся суть AddRoundKey() — произвести XOR State и раундового ключа.

Это, собственно, все, что касается процесса шифрования. Выходной массив зашифрованных байтов составляется из State по формуле **output[r + 4c] = State[r][c], r = 0,1...4; c = 0,1..Nb**.

#### Алгоритм дешифрования

##### Схема

Идея здесь проста: если с тем же ключевым словом выполнить последовательность трансформаций, инверсных трансформациям шифрования, то получится исходное сообщение. Такими инверсными трансформациями являются InvSubBytes(), InvShiftRows(), InvMixColumns() и AddRoundKey(). Общая схема алгоритма расшифровки:



Рис. 9. Схема дешифрования AES.

Стоит отметить, что последовательность добавления раундовых ключей в AddRoundKey() должна быть обратной: от Nr + 1 до 0. Изначально, как и при шифровании, из массива входных байтов формируется таблица State. Затем над ней в каждом раунде производятся преобразования, в конце которых должно получиться расшифрованный файл. Порядок трансформаций немного изменился. Что будет первым, InvSubBytes() или InvShiftRows(), на самом деле не важно, потому что одна из них работает со значениями байтов, а вторая переставляет байты, этих самых значений не меняя, но я придерживался последовательности преобразований в псевдокоде стандарта.

##### InvSubBytes()

Работает точно так же, как и SubBytes(), за исключением того, что замены делаются из константной таблицы InvSbox.

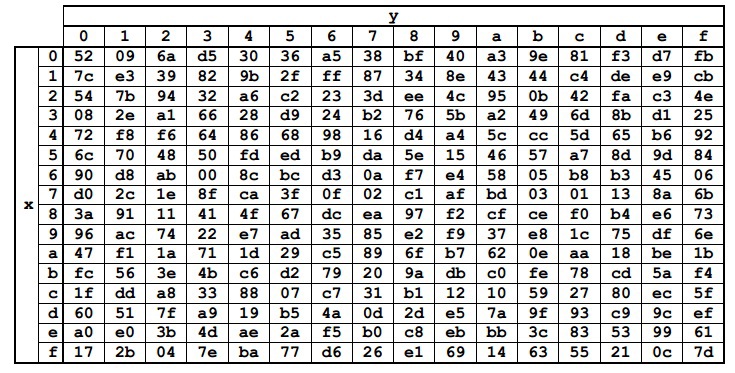


Рис. 10. InvSbox AES.

Оставшиеся обратные трансформации тоже будут очень похожи на свои прямые аналоги, поэтому в коде не выделяем под них отдельных функций. Каждая функция, описывающая трансформацию, будет иметь входную переменную inv. Если она равна False, то функция будет работать в обычном или прямом режиме(шифрование), если True — в инверсном(дешифровка).

##### InvShiftRows()

Трансформация производит циклический сдвиг вправо на 1 элемент для первой строки State, на 2 для второй и на 3 для третьей. Нулевая строка не поворачивается.

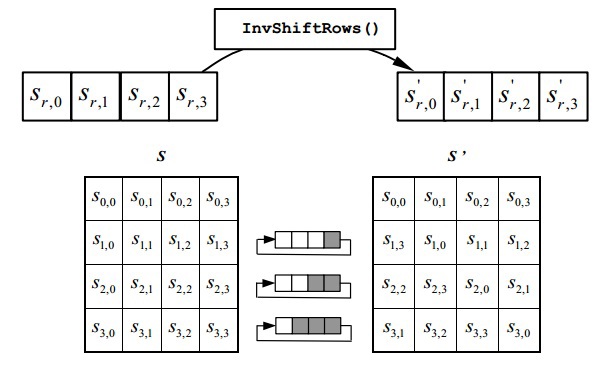


Рис. 11. InvShiftRows AES.

##### InvMixColumns()

Операции те же, но каждая колонка State перемножается с другим многочленом {0b}x3 + {0d}x2 + {09}x + {0e}. В матричной форме это выглядит так:



Рис. 12. InvMixColumns AES.

Или готовые формулы. Все значения в шестнадцатеричной системе счисления.

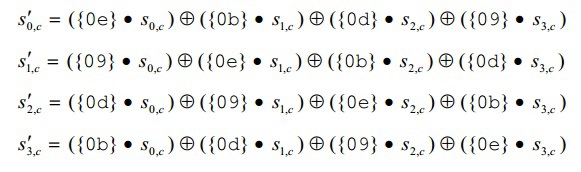


Рис. 13. InvMixColumns AES

##### AddRoundKey()

Эта трансформация обратна сама себе в силу свойства операции XOR: *(a XOR b) XOR b = a*

Поэтому никаких изменений в нее вносить не нужно. Набор раундовых ключей формируется таким же образом, как и для шифрования с помощью функции KeyExpansion(), но раундовые ключи необходимо подставлять в обратном порядке.

### XOR

XOR шифрование основано на свойстве исключающего ИЛИ. Алгоритм:

1. a XOR 0 = a
2. a XOR a = 0
3. a XOR b = b XOR a
4. (a XOR b) XOR b = a

Таким образом, мы получаем самое быстрое симметричное шифрование. При этом в данной записи a – сообщение, b –ключ.

### Обмен ключами Diffie-Hellman

**Протокол Ди́ффи — Хе́ллмана** (англ. *Diffie-Hellman*, *DH*) — криптографический протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя незащищенный от прослушивания канал связи. Полученный ключ используется для шифрования дальнейшего обмена с помощью алгоритмов симметричного шифрования.

#### Алгоритм

##### Схема



Рис. 14. Схема обмена ключами по протоколу Diffie-Hellman.

##### Пример:

Пусть Алена и Борис хотят обменяться ключами.

1. Алена и Борис соглашаются на том, чтобы использовать p = 23 и g = 5
2. Алена выбирает секретное число a=6 и посылает Борису *A* = *g****a*** mod *p*
   1. *A* = 5**6** mod 23 = **15,625** mod 23= 8
3. Борис выбирает секретное число b=15 и посылает Алене *B* = *g****b*** mod *p*
   1. *B* = 5**15** mod 23 = **30,517,578,125** mod 23 = 19
4. Алена вычисляет общий секретный ключ ***s*** = *B****a*** mod *p*
   1. ***s*** = 19**6** mod 23 = **47,045,881** mod 23= **2**
5. Борис вычисляет общий секретный ключ ***s*** = *A****b*** mod *p*
   1. ***s*** = 8**15** mod 23 = **35,184,372,088,832** mod 23 = **2**

В результате получаем общий секрет S. который неизвестен никому, кроме Алены и Бориса.

Примечание:

1. p - простое число
2. a,b - натуральные числа, такие, что a<p, b<p
3. g - первообразный корень по модулю p

**1.2. Исследование существующих решений.**

## 1.3. Выводы по результатам проработки предметной области

# Глава 2: Описание нового протокола

## 2.1. Описание новых методов шифрования и защиты данных

## 2.2. Создание нового протокола

## 2.3. Анализ нового протокола с учетом известных атак на существующие системы

# Глава 3: Разработка серверного приложения

## 3.1. Описание будущего проекта

## 3.2. Создание серверного приложения, описание дизайна

## 3.3. Оценка получившегося решения

# Заключение

## 1. Общая оценка работы

## 2. Полнота решения поставленных задач 3. Экономическая и научная значимость работы

# Список использованных источников

# Приложения