**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

**Кафедра информатики и компьютерных систем**

**Разработка безопасного клиент-серверного приложения для обмена информации на предприятии**

Дипломная работа

Данилина Даниила Алексеевича

студента 4 курса,

специальность

«компьютерная безопасность»

Научный руководитель:

старший преподаватель кафедры информатики и компьютерных систем

Ю. А. Бондаренко

Минск, 2022

Оглавление

[ВЕДЕНИЕ 3](#_Toc91615004)

[ГЛАВА 1 НАЗНАЧЕНИЕ И СТРУКТУРА КРИПТОГРАФИЧЕСКИХ МЕТОДОВ ЗАЩИТЫ 5](#_Toc91615005)

[1.1 Основные понятия 5](#_Toc91615006)

[1.2 Обзор криптографических методов 7](#_Toc91615007)

[ГЛАВА 2 АЛГОРИТМЫ ШИФРОВАНИЯ 10](#_Toc91615008)

[2.1 Симметричные алгоритмы 11](#_Toc91615009)

[2.2 Асимметричные алгоритмы 13](#_Toc91615010)

[2.3 Стойкость шифра 13](#_Toc91615011)

[ГЛАВА 3 АЛГОРИТМ AES И RSA. ОПИСАНИЕ ГИБРИДНЫХ СИСТЕМ 15](#_Toc91615012)

[3.1 Алгоритм блочного шифрования Advanced Encryption Standart (AES) 15](#_Toc91615013)

[3.2 Алгоритм шифрования RSA 16](#_Toc91615014)

[3.3 Гибридное шифрование 18](#_Toc91615015)

[3.4 Пример работы гибридного алгоритма 19](#_Toc91615016)

[3.5 Слабости гибридного шифрования 20](#_Toc91615017)

[ГЛАВА 4 РАЗРАБОТКА ПРОГРАММНОГО ПРИЛОЖЕНИЯ, ПОСТРОЕННОГО НА ГИБРИДНЫХ АЛГОРИТМАХ ШИФРОВАНИЯ 21](#_Toc91615018)

[4.1 Выбор технологий и среды разработки 21](#_Toc91615019)

[4.2 Обмен сообщений и описание работы сервера 22](#_Toc91615020)

[4.3 Методы шифрования/ дешифрования 23](#_Toc91615021)

[4.4 Шифрование и дешифрование AES и RSA в C# 24](#_Toc91615022)

[4.5 Установка соединения с сервером 24](#_Toc91615023)

[4.5 Описание работы программы-клиента 25](#_Toc91615024)

[ЗАКЛЮЧЕНИЕ 28](#_Toc91615025)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc91615026)

# ВЕДЕНИЕ

По оценкам специалистов:

* В среднем хакерские атаки в сети происходят каждые 39 секунд - Университет Мэриленда;
* Утечки данных для публичных компаний обходятся в 116 миллионов долларов - Compliance;
* Когда началась пандемия и офисные работы переместились домой, в США ФБР сообщило о 300% -ном росте киберпреступлений - IMC Grupo;
* 95% нарушений кибербезопасности происходят из-за человеческой ошибки - CYBINT.

Сейчас, более чем когда-либо, информационные системы должны использовать защиту данных предприятия и защиту данных потребителей, необходимые для того, чтобы выдержать растущее количество ежедневных нарушений безопасности. Для многих отраслей нормативные требования и цели соответствия направлены на снижение повышенного риска кражи интеллектуальной собственности. Для потребителей утечка данных в виде украденной парольной фразы, кредитной карты или информации даже с мобильных устройств передает их личную и защищенную информацию в руки злоумышленников [1].

Актуальность и важность проблемы обеспечения информационной безопасности обусловлена следующими факторами:

• Современные уровни и темпы развития средств информационной безопасности значительно отстают от уровней и темпов развития информационных технологий.

• Высокие темпы роста парка персональных компьютеров, применяемых в разнообразных сферах человеческой деятельности [2].

*Целью курсовой работы* разработка безопасного клиент-серверного приложение для шифрованного обмена информацией внутри предприятия.

Таким образом в ходе работы ставятся конкретные *задачи*:

1. Рассмотрение принципов функционирования алгоритмов шифрования
2. Выделение особенностей написания клиент-серверных приложений на основе шифрования;
3. Изучение программных средств языка C# для шифрования и дешифрования текстовых данных по алгоритмам AES и RSA, а также отправки и получения TCP пакетов;
4. Разработка приложения для шифрованного обмена данными внутри предприятия с использованием интуитивно понятного интерфейса.

*Практическая значимость.* Разработка система защиты информации позволяет передавать конфиденциальную информацию. Однако впоследствии можно шифровать информацию с целью её хранения в надёжных источниках.

# ГЛАВА 1 НАЗНАЧЕНИЕ И СТРУКТУРА КРИПТОГРАФИЧЕСКИХ МЕТОДОВ ЗАЩИТЫ

Криптографические методы защиты информации основаны на использовании криптографических систем, или шифров. Криптосистемы позволяют с высокой степенью надежности защитить информацию путем ее специального преобразования. В криптопреобразовании используется один или несколько секретных парамет­ров, неизвестных злоумышленнику, на чем и основана стойкость криптосистем.

Основным достоинством криптографических методов является то, что они обеспечивают высокую гарантированную стойкость защиты, которую можно рассчитать и выразить в числовой форме (средним числом операций или временем, необходимым для раскрытия зашифрованной информации или вычисления ключей) [1].

К числу основных недостатков криптографических методов следует отнести:

* значительные затраты ресурсов (времени, производительности процессоров) на выполнение криптографических преобразований информации;
* трудности совместного использования зашифрованной (подписанной) информации, связанные с управлением ключами (генерация, распределение и т.д.);
* высокие требования к сохранности секретных ключей и защиты открытых ключей от подмены.

### 1.1 Основные понятия

Традиционной задачей криптографии является проблема обеспечения конфиденциальности информации при передаче сообщений по каналу связи. В простейшем случае эта задача описывается взаимодействием трех сторон. На рис. 1.1 представлен пример передачи информации [3].

*Алиса и Боб* — имена, обычно использующиеся в качестве условных обозначений законных взаимодействующих сторон. Используются для удобства объяснения работы: фраза «Алиса посылает Бобу сообщение, зашифрованное его открытым ключом» гораздо легче воспринимается, чем «сторона А посылает стороне Б сообщение, зашифрованное открытым ключом стороны Б». Со временем сформировались традиции, какими именами обозначать каких участников процесса. Наиболее часто Алиса хочет послать сообщение Бобу. Эти имена были использованы Роном Ривестом (Ron Rivest) в 1978 году в журнале «Communications of the ACM» в статье «A Method for Obtaining Digital Signatures and Public-Key Cryptosystems».

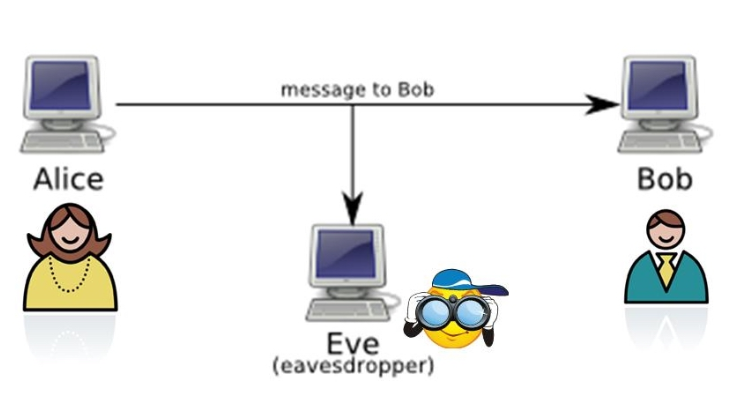


Рис. 1.1 Передача шифрованной информации

*Ева (Eve)* – пассивный злоумышленник, от англ. eavesdropper (подслушивающий). Она может прослушивать сообщения между Алисой и Бобом, но она не может влиять на них. Пассивные атаки связаны с прослушиванием, анализом трафика, перехватом, записью передаваемых шифрованных сообщений, дешифрованием, т.е. попытками «взломать» защиту с целью овладения информацией.

*Мэллори (Mallory, от malicious)* – активный злоумышленник; в отличие от Евы, Мэллори может изменять сообщения, воспроизводить старые сообщения, подменять сообщения и так далее.

Под *шифрованием* обычно понимается семейство обратимых преобразований. Алиса и Боб хотят общаться друг с другом, но к сожалению канал не безопасен. Все сообщения которые Алиса и Боб отсылают друг другу, проходят через Еву. Чтобы Ева не смогла читать сообщения не предназначенные ей, Алиса и Боб должны иметь общий секрет. Так как Ева не знает секрета, она не сможет прочитать это сообщение [2].

*Ключ* - это важнейший компонент шифра. Чтобы Алисе и Бобу получить общий секрет им следует о нем договориться. Так как они не могут использовать для этого небезопасный канал, который прослушивает Ева, им придется воспользоваться другим вариантом. Вариант который здесь будет рассмотрен называется криптография с открытым ключом. Для того чтобы Алисе и Бобу безопасно обменяться ключами, Бобу нужно сгенерировать два ключа, приватный и публичный. Далее Боб отсылает публичный ключ Алисе, не опасаясь того, что Ева его тоже может получить, иначе почему бы ему называться публичным. После того, как Алиса получила публичный ключ Боба, она может зашифровать им свой секрет и отправить Бобу. Сообщение зашифрованное публичным ключом можно расшифровать только зная парный секретный ключ. Когда Боб получит зашифрованное сообщение он может воспользоваться своим секретным ключом для его расшифровки, после этого Алиса и Боб будут иметь общий секрет. Так, например, работает самый популярный алгоритм с открытым ключом – RSA. Но этот метод с 2015 года считается историческим по ряду причин. Например, если секретный ключ RSA когда-нибудь будет скомпрометирован, все предыдущие сообщения, которые получала Ева, можно будет расшифровать [4].

### 1.2 Обзор криптографических методов

Криптографические методы могут быть классифицированы различным образом, но наиболее часто они подразделяются в зависимости от количества ключей, используемых в соответствующих криптоалгоритмах (рис. 1.2) [4]:

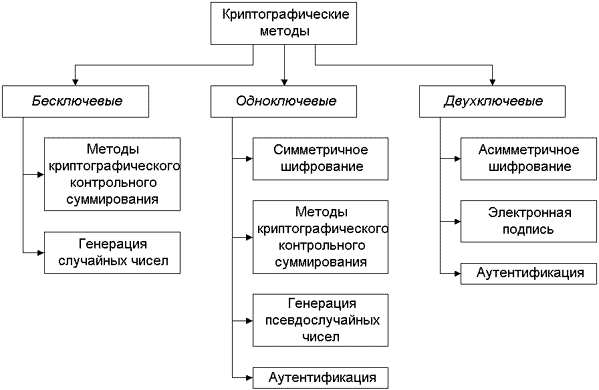
1. Бесключевые, в которых не используются какие-либо ключи.
2. Одноключевые - в них используется некий дополнительный ключевой параметр - обычно это секретный ключ.
3. Двухключевые, использующие в своих вычислениях два ключа: секретный и открытый.

Рис. 1.2 Криптографические методы

Шифрование является основным методом защиты; рассмотрим его подробно далее.

Стоит сказать несколько слов и об остальных криптографических методах:

1. Электронная подпись используется для подтверждения целостности и авторства данных. Целостность данных означает, что данные не были случайно или преднамеренно изменены при их хранении или передаче.  
   Алгоритмы электронной подписи используют два вида ключей:
   * секретный ключ используется для вычисления электронной подписи;
   * открытый ключ используется для ее проверки.

При использовании криптографически сильного алгоритма электронной подписи и при грамотном хранении и использовании секретного ключа (то есть при невозможности использования ключа никем, кроме его владельца) никто другой не в состоянии вычислить верную электронную подпись какого-либо электронного документа [6].

1. Аутентификация позволяет проверить, что пользователь (или удаленный компьютер) действительно является тем, за кого он себя выдает. Простейшей схемой аутентификации является парольная - в качестве секретного элемента в ней используется пароль, который предъявляется пользователем при его проверке. Такая схема доказано является слабой, если для ее усиления не применяются специальные административно-технические меры. А на основе шифрования или хэширования (см. ниже) можно построить действительно сильные схемы аутентификации пользователей [5].
2. Существуют различные методы криптографического контрольного суммирования:
   * ключевое и бесключевое хэширование;
   * вычисление имитоприставок;
   * использование кодов аутентификации сообщений.

Фактически, все эти методы различным образом из данных произвольного размера с использованием секретного ключа или без него вычисляют некую контрольную сумму фиксированного размера, однозначно соответствующую исходным данным.  
Такое криптографическое контрольное суммирование широко используется в различных методах защиты информации, например:

* + для подтверждения целостности любых данных в тех случаях, когда использование электронной подписи невозможно (например, из-за большой ресурсоемкости) или является избыточным;
  + в самих схемах электронной подписи - "подписывается" обычно хэш данных, а не все данные целиком;
  + в различных схемах аутентификации пользователей [3].

1. Генераторы случайных и псевдослучайных чисел позволяют создавать последовательности случайных чисел, которые широко используются в криптографии, в частности:
   * случайные числа необходимы для генерации секретных ключей, которые, в идеале, должны быть абсолютно случайными;
   * случайные числа применяются во многих алгоритмах электронной подписи;
   * случайные числа используются во многих схемах аутентификации.

Не всегда возможно получение абсолютно случайных чисел - для этого необходимо наличие качественных аппаратных генераторов. Однако, на основе алгоритмов симметричного шифрования можно построить качественные генераторы псевдослучайных чисел [8].

# ГЛАВА 2 АЛГОРИТМЫ ШИФРОВАНИЯ

Шифрование — это преобразование данных в нечитабельную форму, используя ключи шифрования-расшифровки. Шифрование является наиболее широко используемым криптографическим методом сохранения конфиденциальности информации, он защищает данные от несанкционированного ознакомления с ними [2].

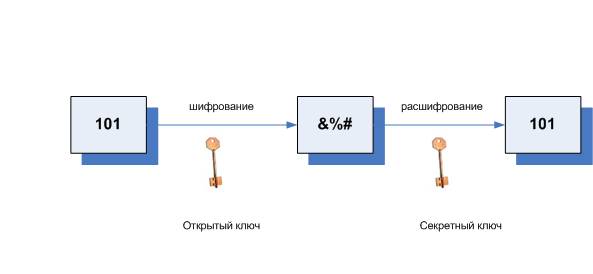
Алгоритм шифрования данных с использованием открытого ключа приведена на рис. 2.1 и состоит из двух этапов. На первом из них производится обмен по несекретному каналу открытыми ключами. При этом необходимо обеспечить подлинность передачи ключевой информации. На втором этапе, собственно, реализуется шифрование сообщений, при котором отправитель зашифровывает сообщение открытым ключом получателя.

Рис. 2.1 Алгоритм шифрования данных

Алгоритмы шифрования можно разделить на две категории:

1. Алгоритмы симметричного шифрования.
2. Алгоритмы асимметричного шифрования.

В алгоритмах *симметричного шифрования* для расшифрования обычно используется тот же самый ключ, что и для шифрования, или ключ, связанный с ним каким-либо простым соотношением. Последнее встречается существенно реже, особенно в современных алгоритмах шифрования. Такой ключ (общий для зашифрования и расшифрования) обычно называется просто *ключом шифрования [9]*.

В *асимметричном шифровании* ключ шифрования *k1* легко вычисляется из ключа *k2* таким образом, что обратное вычисление невозможно. Например, соотношение ключей может быть таким:

где a и p - параметры алгоритма шифрования, имеющие достаточно большую размерность.

Такое соотношение ключей используется и в алгоритмах электронной подписи [10].

### 2.1 Симметричные алгоритмы

Симметричный алгоритм прекрасно подходит при передаче больших объёмов зашифрованных данных. Симметричное шифрование — способ шифрования, в котором для шифрования и расшифровывания применяется один и тот же криптографический ключ [9]. Ассиметричный в этом случае будет работать медленнее. Кроме того, при организации обмена информацией по ассиметричному алгоритму оба ключа должны быть известны обеим сторонам либо пар должно быть две (по одной на каждую сторону) [10].

Ниже таблице 2.1 приведен обзор некоторых алгоритмов симметричного шифрования:

Таблица 2.1 — Алгоритмы симметричного шифрования

|  |  |
| --- | --- |
| **Тип** | **Описание** |
| DES (Data Encryption  Standard) | Популярный алгоритм шифрования, используемый как стандарт шифрования данных правительством США. Шифруется блок из 64 бит, используется 64-битовый ключ (требуется только 56 бит), 16 проходов.  Может работать в 4 режимах:   * Электронная кодовая книга (ECB-Electronic Code Book) - обычный DES, использует два различных алгоритма. * Цепочечный режим (CBC-Cipher Block Chaining), в котором шифрование блока данных зависит от результатов шифрования предыдущих блоков данных. * Обратная связь по выходу (OFB-Output Feedback), используется как генератор случайных чисел.   Обратная связь по шифратору (CFB-Cipher Feedback), используется для получения кодов аутентификации сообщений. |
| 3-DES или  тройной DES | 64-битный блочный шифратор, использует DES 3 раза с тремя различными 56-битными ключами. Достаточно стоек ко всем атакам |
| Каскадный 3-DES | Стандартный тройной DES, к которому добавлен механизм обратной связи, такой как CBC, OFB или CFB. Очень стоек ко всем атакам. |
| FEAL (быстрый  алгоритм шифрования) | Блочный шифратор, используемый как альтернатива DES. Вскрыт, хотя после этого были предложены новые версии. |
| IDEA (международный  алгоритм шифрования) | 64-битный блочный шифратор, 128-битовый ключ, 8 проходов. Предложен недавно; хотя до сих пор не прошел полной проверки, чтобы считаться надежным, считается более лучшим, чем DES |
| Rijndael | Алгоритм разработан в Бельгии. Работает с ключами длиной 128, 192 и 256 бит. На данный момент к нему нет претензий у специалистов по криптографии |
| RC2 | 64-битный блочный шифратор, ключ переменного размера. Приблизительно в 2 раза быстрее, чем DES. Может использоваться в тех же режимах, что и DES, включая тройное шифрование. Конфиденциальный алгоритм, владельцем которого является RSA Data Security |
| RC4 | Потоковый шифр, байт-ориентированный, с ключом переменного размера. Приблизительно в 10 раз быстрее DES. Конфиденциальный алгоритм, которым владеет RSA Data Security |
| RC5 | Имеет размер блока 32, 64 или 128 бит, ключ с длиной от 0 до 2048 бит, от 0 до 255 проходов. Быстрый блочный шифр. Алгоритм, которым владеет RSA Data Security |
| CAST | 64-битный блочный шифратор, ключи длиной от 40 до 64 бит, 8 проходов. Неизвестно способов вскрыть его иначе как путем прямого перебора. |
| Blowfish. | 64-битный блочный шифратор, ключ переменного размера до 448 бит, 16 проходов, на каждом проходе выполняются перестановки, зависящие от ключа, и подстановки, зависящие от ключа и данных. Быстрее, чем DES. Разработан для 32-битных машин |
| ГОСТ Р 34.10-2012 [20] | Данный алгоритм основан на эллиптических кривых. Считается, что задача дискретного логарифма на эллиптической кривой является более трудной по сравнению с задачей дискретного логарифма в конечных полях. |
| Поточные шифры | Быстрые алгоритмы симметричного шифрования, обычно оперирующие битами (а не блоками бит). Разработаны как аналог устройства с одноразовыми ключами, и хотя не являются такими же безопасными, как оно, по крайней мере практичны. |

### 2.2 Асимметричные алгоритмы

Ассиметричное шифрование позволяет дать старт безопасному соединению без каких-либо усилий со стороны пользователя. Если говорить о симметричном шифровании, то пользователю нужно знать пароль. Однако не стоит думать, что ассиметричный подход безопасен на 100 %. К примеру, он подвержен атакам «человек посередине». Это когда между сервером и вами размещается компьютер, который вам отсылает свой открытый ключ, а при передаче информации с вашей стороны, использует открытый ключ сервера. В итоге происходит перехват конфиденциальных данных [11].

Ниже таблице 2.2 приведен обзор некоторых алгоритмов асимметричного шифрования:

Таблице 2.2 - Алгоритмы асимметричного шифрования

|  |  |
| --- | --- |
| **Тип** | **Описание** |
| RSA | Популярный алгоритм асимметричного шифрования, стойкость которого зависит от сложности факторизации больших целых чисел. |
| ECC (криптосистема  на основе  эллиптических кривых) | Использует алгебраическую систему, которая описывается в терминах точек эллиптических кривых, для реализации асимметричного алгоритма шифрования. Является конкурентом по отношению к другим асимметричным алгоритмам шифрования, так как при эквивалентной стойкости использует ключи меньшей длины и имеет большую производительность. Современные его реализации показывают, что эта система гораздо более эффективна, чем другие системы с открытыми ключами. Его производительность приблизительно на порядок выше, чем производительность RSA, Диффи-Хеллмана и DSA. |
| Эль-Гамаль. | Вариант Диффи-Хеллмана, который может быть использован как для шифрования, так и для электронной подписи. |

### 2.3 Стойкость шифра

Способность шифра противостоять всевозможным атакам на него называют стойкостью шифра. Под атакой на шифр понимают попытку вскрытия этого шифра. Понятие стойкости шифра является центральным для криптографии. Хотя качественно понять его довольно легко, но получение строгих доказуемых оценок стойкости для каждого конкретного шифра - проблема нерешенная. Это объясняется тем, что до сих пор нет необходимых для решения такой проблемы математических результатов. Поэтому стойкость конкретного шифра оценивается только путем всевозможных попыток его вскрытия и зависит от квалификации криптоаналитиков, атакующих шифр [15].

Такую процедуру иногда называют проверкой стойкости. Важным подготовительным этапом для проверки стойкости шифра является продумывание различных предполагаемых возможностей, с помощью которых противник может атаковать шифр [1]. Появление таких возможностей у противника обычно не зависит от криптографии, это является некоторой внешней подсказкой и существенно влияет на стойкость шифра. Поэтому оценки стойкости шифра всегда содержат те предположения о целях и возможностях противника, в условиях которых эти оценки получены .

Из более специфических приведу три примера возможностей противника:

1. противник может перехватывать все шифрованные сообщения, но не имеет соответствующих им открытых текстов;
2. противник может перехватывать все шифрованный сообщения и добывать соответствующие им открытые тексты;
3. противник имеет доступ к шифру (но не к ключам!) и поэтому может зашифровывать и дешифровывать любую информацию[12];

# ГЛАВА 3 АЛГОРИТМ AES И RSA. ОПИСАНИЕ ГИБРИДНЫХ СИСТЕМ

Глава содержит разбор работы криптографических алгоритмов шифрования AES и RSA. А также обзор гибридных криптосистем и почему они получили такое широкое распространение.

### 3.1 Алгоритм блочного шифрования Advanced Encryption Standart (AES)

Advanced Encryption Standard – симметричный алгоритм блочного шифрования, принятый правительством США в качестве стандарта в результате конкурса, проведенного между технологическими институтами. Он заменил устаревший Data Encryption Standard, который больше не соответствовал требованиям сетевой безопасности, усложнившимся в XXI веке [14].

Благодаря разработкам компании Intel инструкции для реализации алгоритма AES вшиты в сами процессоры, что не только упрощает работу с алгоритмом, но и значительно ускоряет его [13].

Всего имеется шесть инструкций, использование которых и позволяет шифровать данные. Под инструкциями подразумевается набор команд, которые выполняются над входными данными, после ее вызова.

Одна из инструкций — генерирование раундовых ключей — производит «расширение» имеющегося ключа.

В зависимости от его длины (128, 192 или 256 бит), процедуры шифрования и расшифрования происходят за 10, 12 или 14 раундов — в каждом раунде повторяется применение соответствующих инструкций, после выполнения которых мы и получаем верно зашифрованный текст. В таблице 3.1 представлено соответствие между длинной ключа, размером блока данных и числом раундов.

Таблице 3.1 - Cоответствие между длинной ключа, размером блока данных и числом раундов

|  |  |  |  |
| --- | --- | --- | --- |
| Стандарт | Длина ключа  (Nk слов) | Размер блока данных  (Nb слов) | Число раундов |
| AES-128 | 4 | 4 | 10 |
| AES-192 | 6 | 4 | 12 |
| AES-256 | 8 | 4 | 14 |

Из одного ключа с помощью инструкции расширения получается 10, 12 или 14 раундовых ключей длины 128 бит. Необходимо уточнить, что полученные раундовые ключи могут использоваться только для шифрования данных. Для получения ключей расшифрования на помощь приходит еще одна инструкция (инверсии), которая применяется к уже имеющимся ключам шифрования. Так, мы имеем два набора ключей в количестве 10, 12 или 14 штук [3].

После получения нужного количества ключей можно приступать к процедуре шифрования или дешифрования. Для этого используются четыре оставшиеся инструкции. Две из них являются инструкциям шифрования, две другие реализуют обратные преобразования.

На первом шаге исходный открытый текст складывается с исходным ключом (или его частью, если его размер больше 128 бит).

Побитовое сложение подразумевает под собой следующее: две последовательности нулей и единиц записываются одна под другой, символ первой строки складывается с символом второй по правилу «исключающего или» (операция XOR), где 1 + 1 = 0, 0 + 0 = 0, 0 + 1 = 1 + 0 = 1.

После сложения текста и ключа мы приступаем к преобразованию получившейся последовательности — для этого мы и используем инструкцию процессора, применяя ее к полученному промежуточному результату и ключу соответствующего раунда.

Вышеупомянутая инструкция реализует один раунд шифрования, таким образом, мы повторяем ее выполнение до предпоследнего раунда включительно.

Последний раунд шифрования отличается от предыдущих: выполнение инструкции осуществляется один раз над промежуточным результатом и ключом последнего раунда. Вот и все — необходимо лишь определенное число раз повторить выполнение соответствующих инструкций, и мы уже имеем шифрованный текст [5].

Расшифрование происходит аналогичным образом с помощью набора раундовых ключей и двух инструкций процессора. Главное требование, которое позволит провести успешный обмен сообщениями — секретность ключа.

### 3.2 Алгоритм шифрования RSA

Рассмотрим задачу сохранности содержимого посылки при передаче от отправителя к адресату (рис. 3.1).

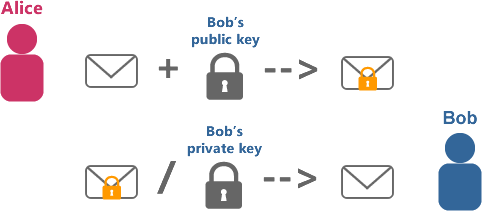


Рис. 3.1 Алиса и Боб, задача сохранности содержимого посылки

Алиса хочет передать Бобу посылку. Для начала Боб на своей стороне создает уникальные замок и ключ к нему (открытый и закрытый ключ соответственно). Далее, Боб делится с окружающим миром своим замком, чтобы любой желающий отправить ему посылку смог её закрыть. Поскольку ключ от подобного замка один и находится только у Боба, никто, кроме Боба, просмотреть содержимое после защёлкивания замка не сможет. В конце концов, Алиса с помощью полученного замка закрывает посылку и передаёт Бобу, который открывает её своим ключом. Таким образом устроены асимметричные криптографические системы, которой как раз является RSA [11].

В основе подобных алгоритмов шифрования лежат односторонние функции - то есть, такие математические функции, которые являются однозначными (одному аргументу всегда соответствует одно-единственное значение такой функции), но которые при этом технически невозможно обратить, то есть вычислить по значению функции значение её аргумента.

В RSA используется факторизация простых чисел (разложение их на составные множители). Эта задача является, с точки зрения вычислений, односторонней, что и позволяет использовать её в алгоритме шифрования с открытым ключом [12]. Алгоритм считается надёжным, когда берутся два случайных простых числа длиной не менее 512 битов каждое. Сами формулы, на которых базируется RSA-шифрование, приводить вряд ли имеет смысл, поскольку они достаточно сложны и вряд ли нужны каждому читателю.

Надёжность RSA во многом обеспечивается тем, что при использовании достаточно длинных чисел разложить их произведение на составные множители за приемлемое время пока что не удаётся, хотя и существует ряд атак, использующих специфичные просчёты при шифровании. Ожидается, что с созданием квантовых компьютеров от алгоритма RSA придётся отказаться, поскольку они будут в состоянии быстро раскладывать на множители длинные целые числа.базируется RSA-шифрование, приводить вряд ли имеет смысл, поскольку они достаточно сложны и вряд ли нужны каждому читателю.

Надёжность RSA во многом обеспечивается тем, что при использовании достаточно длинных чисел разложить их произведение на составные множители за приемлемое время пока что не удаётся, хотя и существует ряд атак, использующих специфичные просчёты при шифровании. Ожидается, что с созданием квантовых компьютеров от алгоритма RSA придётся отказаться, поскольку они будут в состоянии быстро раскладывать на множители длинные целые числа [7].

### 3.3 Гибридное шифрование

Симметричный шифр обычно строится на основании ряда блоков с математическими функциями преобразования, ассиметричный — на математических задачах. Тот же RSA создан на задаче возведения в степень с последующим вычислением модуля. В результате алгоритмы симметричного шифрования модифицировать просто, а ассиметричного — практически невозможно. Так же стоит сказать, что основной недостаток асимметричной криптографии состоит в низкой скорости из-за сложных вычислений, требуемых её алгоритмами, в то время как симметричная криптография традиционно показывает высокую скорость работы [13].

Однако симметричные криптосистемы имеет один существенный недостаток — её использование предполагает наличие защищённого канала для передачи ключей. Для преодоления этого недостатка прибегают к асимметричным криптосистемам, которые используют пару ключей: открытый и закрытый.

Лучший эффект шифрования достигается при комбинации симметричного и асимметричного шифрования . Происходит это так:

1. посредством ассиметричного алгоритма отсылается сессионный ключ для симметричного шифрования;
2. происходит обмен информацией по симметричному алгоритму.

Тут возможны варианты, но общий смысл обычно не меняется.

Хорошим примером таких гибридных систем являются криптографические протоколы Security Sockets Layer (SSL) и Transport Layer Security (TLS), которые были разработаны для обеспечения безопасной связи в интернете. Протоколы SSL на данный момент считаются небезопасными и ими не рекомендуют пользоваться. В свою очередь, протоколы TLS считаются безопасными и широко используются всеми современными веб-браузерами [2].

### 3.4 Пример работы гибридного алгоритма

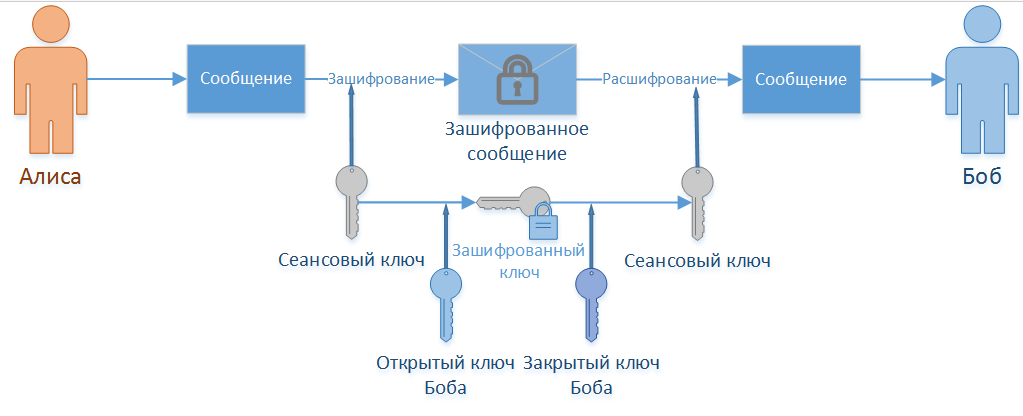
Передачу данных с использованием гибридного шифрования можно условно разделить на два этапа: этап отправки и этап приема (рис. 3.2).

Рис. 3.2 Пример работы гибридного алгоритма

**Этап отправки:**

* Алиса генерирует случайный сеансовый ключ;
* Сообщение Алисы зашифровывается сеансовым ключом (с помощью симметричного алгоритма);
* Сеансовый ключ зашифровывается открытым ключом Боба (асимметричным алгоритмом);
* Алиса посылает Бобу зашифрованное сообщение и зашифрованный сеансовый ключ.

**Этап приёма:**

* Боб получает зашифрованное сообщение Алисы и зашифрованный сеансовый ключ;
* Боб расшифровывает сеансовый ключ своим закрытым ключом;
* При помощи полученного, таким образом, сеансового ключа Боб расшифровывает зашифрованное сообщение Алисы.

Большинство гибридных систем работают следующим образом. Для симметричного алгоритма (3DES, IDEA, AES или любого другого) генерируется случайный сеансовый ключ [1]. Такой ключ как правило имеет размер от 128 до 512 бит (в зависимости от алгоритма). Затем используется симметричный алгоритм для шифрования сообщения. Что касается самого случайного ключа, он должен быть зашифрован с помощью открытого ключа получателя сообщения, и именно на этом этапе применяется криптосистема с открытым ключом (RSA или алгоритм Диффи-Хеллмана) [2].

### 3.5 Слабости гибридного шифрования

С точки зрения организации протокола гибридная схема очень проста. Однако существуют два ограничения в использовании гибридных криптосистем.

Во-первых, эта схема использует сеансовый ключ, созданный одной из сторон (отправителем сообщения или инициатором протокола), а другая сторона (получатель сообщения или адресат протокола) должен целиком полагаться на компетентность и честность инициатора протокола. В некоторых ситуациях это нежелательно: например, в протоколе SSL, в рамках которого клиентом является отправитель, а точнее – его программное обеспечение, которое, как известно, представляет собой весьма слабый датчик случайных чисел [12].

Во-вторых, гибридные системы шифрования инертны. В таких системах перехватчик, который может силой заставить получателя раскрыть свой закрытый ключ, получает возможность расшифровывать все сообщения. Такая особенность называется недостатком «заблаговременной секретности». Заблаговременная секретность означает, что перехватчик не в состоянии расшифровать исходное сообщение в будущем, используя зашифрованные сообщения, полученные в прошлом, ни с помощью криптоанализа, ни с помощью принуждения [12].

# ГЛАВА 4 РАЗРАБОТКА ПРОГРАММНОГО ПРИЛОЖЕНИЯ, ПОСТРОЕННОГО НА ГИБРИДНЫХ АЛГОРИТМАХ ШИФРОВАНИЯ

В качестве результата проведенного исследования в этой главе я предоставляю описание приложения для безопасного обмена данными, а также пример его программной реализации. В состав программной реализации, разработанного в соответствии с заданием входят:

* программа, выполняющая функции сервера;
* программа, выполняющая функцию клиента.

При разработке программного приложения требовалось решить следующие задачи:

* Выбор технологий и среды разработки;
* Формирование меню;
* Формирование процедур, реализующих обмен сообщений;
* Реализация методов шифрования/дешифрования:

### 4.1 Выбор технологий и среды разработки

Большинство интернет-провайдеров предлагают передачу данных через свои серверы. То есть, они обеспечивают клиент-серверную архитектуру связи. Такую ситуацию можно увидеть в популярных мессенджерах для общения: Viber, Telegram, WhatsApp. Если говорить о безопасности передачи данных, то такая структура изначально представляет угрозу, если пользователь не доверяет разработчикам приложения, так как личная информация передается через сервера, где может сохраняться, изменяться, передаваться третьим лицам.

Эту проблему можно решить, организовав соединение типа точка-точка (peer-to-peer) между абонентами-пользователями. Такое соединение позволяет передавать данные напрямую между абонентами, что значительно снижает вероятность доступа к этому трафику третьими лицами. Вопрос лишь в том, как обменяться данными для подключения друг к другу, ведь когда у приложения клиент-серверная архитектура, клиенты просто подключаются к серверу, который всегда расположен по одному и тому же адресу. Есть разные способы обмена контактной информацией между клиентами: можно как передать данные в реальном мире, так и отправить электронным письмом. Стоит также помнить о том, что при передаче информации, пользователь может менять своё местоположение. При этом его ноутбук будет отключаться от одного маршрутизатора и подключатся к другому, что в свою очередь повлечет за собой смену и внешнего, и внутреннего IP-адресов пользователя. Это в свою очередь повлечет за собой разрыв прямого соединения.

С учетом всего этого предлагается клиент-серверная архитектура приложения. Клиент с помощью сервера контролирует свои IP-адреса, отправляет их на сервер приложения и обновляет при смене. Таким образом возможно будет решить проблему с разорванным соединением, а также будет отображаться статус собеседника (в сети, не в сети). Сервер в свою очередь хранит идентификаторы пользователей и их корректные IP-адреса. При создании соединения двух пользователей, сервер должен отправить им адреса друг друга, чтобы они смогли соединиться напрямую, а также должен обновлять адреса для поддержания соединения. Дополнительным плюсом такой архитектуры является отсутствие необходимости передавать все данные сервером. То есть, затраты на сервер будут существенно меньше, при этом приложение проще масштабировать.

Практическая часть курсовой работы представлена в виде приложение Windows Forms созданного в среде Microsoft Visual Studio Community 2019 v16.7.6 и написанного на языке C # (.NET Framework v4.7.2). Приложение создано с использованием классов TcpListener и TcpClient, CspParameters и RSACryptoServiceProvider. А также функции асинхронного чтения и записи. Приложения-клиенты реализуют выполнение следующих функций: открытие сокета, установка связи с сервером, подготовка и передача данных в соответствии с заданием. Сервер использует многопоточность для приема клиентов (т.е. сервер может принимать несколько клиентов). Это приложение обеспечивает шифрование и дешифрование пользовательских данных, а так же обмен зашифрованными письмами внутри сети предприятия на основе гибридного алгоритма.

Протестировано на ОС Windows 10 версии 21H1 (64-бит).

### 4.2 Обмен сообщений и описание работы сервера

Для реализации TCP сервера, мне потребовалось использовать классы TcpListener (для прослушивания определённого порта и управления подключения клиентов) и TcpClient (для обмена данными с подключенными клиентами).

Поддержка сокетов TCP на платформе .NET значительно усовершенствована по сравнению с предыдущей моделью программирования. В отличие от класса Socket, в котором для отправки и получения данных применяется побайтовый подход, TpListener придерживается потоковой модели. В этом классе все взаимодействие между клиентом и сервером базируется на потоке с использованием класса NetworkStream. Однако при необходимости можно работать с байтами.

Конструктор класса TcpListenet принимает два параметра IP адрес и номер порта. IP адрес устанавливается по умолчанию – это значение адреса компьютера на котором запускается сервер. Номер порта выбран из области свободно используемых значений и равен 5000.

Запуск сервера в работу осуществляется при помощи метода server.Start().

После вызова метода Start() запускается бесконечный цикл, в котором объект TcpListener будет ожидать подключения клиента и как только оно произойдёт при помощи метода AcceptTcpClient будет создан объект User, который позволяет производить обмен данными с подключившимся клиентом.

Для получения и отправки данных с помощью потоков в .NET используется класс потоков NetworkStream из пространства имен System.Net.Sockets. Он наследуется от базового класса Stream. В то же время он отличается от других классов потоков тем, что он не является буферизованным и не поддерживает перемещение в произвольную позицию с помощью метода Seek.

Поскольку клиент может послать данные гораздо большего размера, чем размер буфера потока, то для считывания всех данных, я использовал цикл do..while, который проверяем наличие данных с помощью свойства stream.DataAvailable

История переписки не сохраняется во внешней памяти клиента. Она существует только пока работает приложение. После выхода, переписка удаляется. Информация о том, кто и когда подключился и отключился и с кем вел переписку сохраняется на сервера в Log файле. Также стоит обратить внимание на то, что в случае возникновения исключения, работа сервера останавливается, чтобы в случае критической ошибки программа не «зависла».

К сожалению, в данном приложении не был реализован механизм постоянной поддержки связи за счет сервера при смене IP-адреса.

### 4.3 Методы шифрования/ дешифрования

Общим способом шифрования большего объема является использование гибридного шифрования. Поэтому было принято решение в его использование. Сначала мы создаем случайный симметричный ключ, который используется для шифрования данных. В качестве симметричного алгоритма я решил взять принятый за стандарт правительством США симметричный алгоритм блочного шифрования AES c длиной ключа 128 бит. Этот алгоритм хорошо проанализирован и сейчас широко используется. Затем мы шифруете симметричный ключ с помощью асимметричного ключа. Используя в качестве ассиметричного алгоритма – RSA. После шифрования зашифрованные данные + зашифрованный симметричный ключ затем объединяются и составляют полные зашифрованные данные. Что касается самого симметричного ключа, он должен быть зашифрован с помощью открытого ключа получателя сообщения, и именно на этом этапе применяется криптосистема с открытым ключом RSA.

### 4.4 Шифрование и дешифрование AES и RSA в C#

При запуске приложения создаются закрытый и открытый ключи пользователя. Чтобы соединиться с собеседником необходимо, чтобы он присутствовал в списке контактов на сервере, где должен храниться его открытый ключ. Открытый ключ представляет собой точку на эллиптической кривой. При передаче данных о точке координаты x и y "склеиваются", образуя длинную строку открытого ключа. Закрытый ключ представляет собой большое целое число. Инициатор пишет сообщение и нажимает кнопку отправить. При этом приложение шифрует сообщение и прикрепляет к началу шифротекста временный, сеансовый ключ. Получатель принимает сообщение и с помощью временного ключа расшифровывает его.

### 4.5 Установка соединения с сервером

После запуска программы происходит инициализация данных пользователя, а именно: IP адресс, MAC адресс, и собственное имя клиента. Данные сервера вбиваются администратором на предприятии самостоятельно.

Клиентская часть использует только класс TcpClient. Создав экземпляр класса TcpClient, следующим шагом идет установка соединения с удаленным сервером. Для соединения клиента с сервером TCP используется метод Connect(), где передается объект IPEndPoint, представляющий удаленную конечную точку, с которой надо соединиться. Если соединение будет неудачным или возникнут другие проблемы, порождается исключение SocketException.

Для обработки на уровне потока, как канал между двумя соединенными приложениями, используется класс NetworkStream. Прежде чем отправлять и получать любые данные, нужно определить базовый поток. Класс TcpClient предоставляет метод GetStream() исключительно для этих целей. С помощью базового сокета он создает экземпляр класса NetworkStream и возвращает его вызывающей программе. Получив поток, используются методы Read() и Write() класса NetworkStream для чтения из приложения хоста и записи к нему. Метод Write() принимает три параметра: массив байтов, содержащий данные, которые надо отправить хосту, позицию в потоке, с которой хотим начать запись, и длину данных.

### 4.5 Описание работы программы

Для простоты использования на предприятии необходимо было создать удобный интерфейс для пользователя.

Программа предназначена для приятного и полезного использования в рабочее время. Поэтому в меню использованы максимально лаконичные команды и ответы на них. Это во многом облегчает работу пользователя.

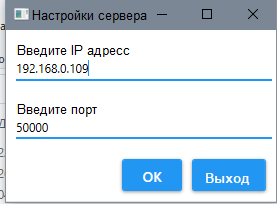


Рисунок 4.1 Окно запуска сервера

После успешной запуска пользователь попадает в основное окно сервера, где у него доступен весь функционал сервера (рис. 4.2).  Администратор может как отключать клиентов так и смотреть log файлы.

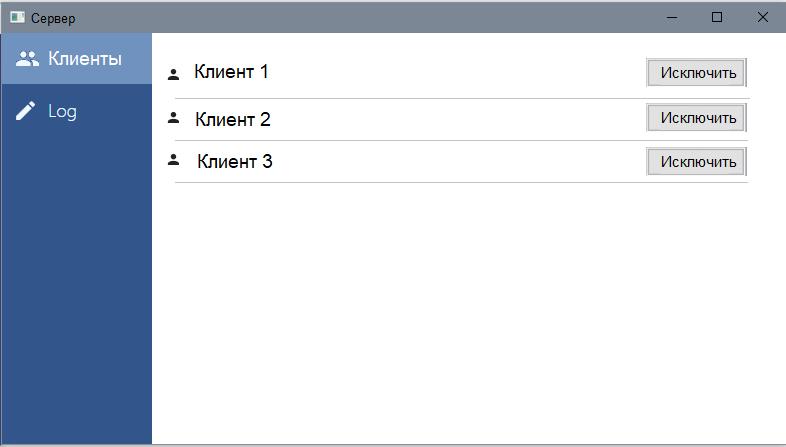


Рисунок 4.2 Основное окно сервера

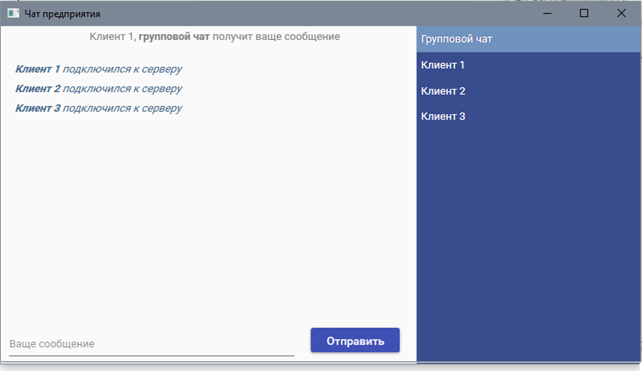
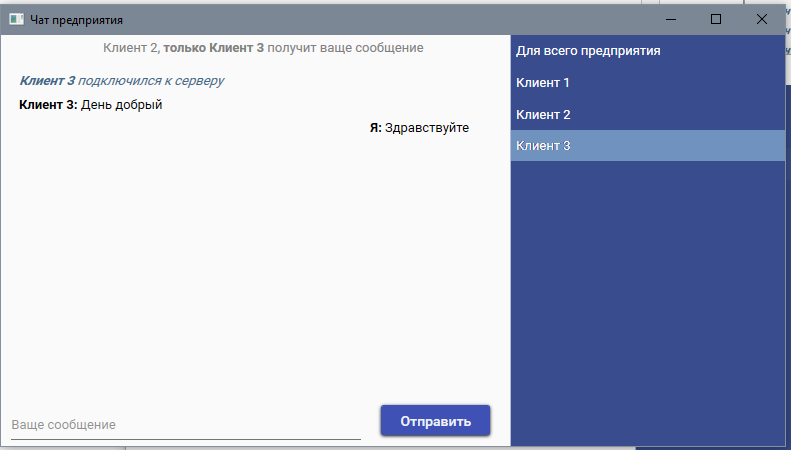
После запуска сервера клиенты смогут подключиться к серверу. Для этого нужно будет всего лишь запустить клиентское приложения (рис. 4.3).

Рисунок 4.3 Основное окно клиента

В приложении сотрудник сможет как общаться в общем рабочем чате, так и общаться непосредственно с каждым работником отдельно (рис. 4.4).

Рисунок 4.4 Окно личного чата

# ЗАКЛЮЧЕНИЕ

В настоящее время популярность мессенджеров, как средств общения неизменно растет. В больших корпорациях обмен конфиденциальной информации, ежедневно происходит с использованием средств связи. Так же стоит отметить рост необходимости наличия возможности качественного шифрования данных, для защищенного обмена.

Поэтому в рамках курсовой работе были рассмотрены методы шифрования данных с использованием схемы гибридного шифрования. Эта схема является комбинацией криптосистем RSA с симметричной криптосистемой AES. В ходе выполнения данной курсовой работы был создан программный продукт, предназначенный для шифрования/дешифрования текстовой информации с использованием алгоритма AES, применяя криптосистему с открытым ключом.

Программа выполнена в виде приложения в стандартном оконном режиме, обладает интерфейсом пользователя эстетичного внешнего вида с эргономичным расположением управляющих элементов, строго определяющих диалог пользователя с программой. В разработанной программе также имеет место обработчик ошибок пользователя и выполнения алгоритма с выдачей диагностических сообщений.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Cryptographic methods and information security tools [Electronic resource] / John Simpson et al. – Oxford English Dictionary, second edition, 2009. – ISBN 0-19-861186-2. Mode of access: https://www.oed.com/. – Date of access: 1.05.2021.

2. Информационные системы и радиотехнологии 2020: [Электронный ресурс]/ Богуш В.А. Data Management. – 2020. Режим доступа: https://arxiv.org/pdf/1710.05364.pdf. – Дата доступа: 1.05.2021.

3. Secure schemes for secret sharing and key distribution / A. Beimel, // Psychological Bulletin, Vol. 116, no. 1, 1994. – P. 75-98.

4. Ciphertext-policy attributebased encryption / M. Potthastet al. // European Conference on Information Retrieval (Padua, Italy, March 20-23, 2016), 2016. – P. 810–817. DOI:10.1007/978-3-319-30671-1\_72.

5. Dong, M. Similarity-Aware Deep Attentive Model for Clickbait Detection / M. Dong et al. // Pacific-Asia Conference on Knowledge Discovery and Data Mining: Advances in Knowledge Discovery and Data Mining (Delhi, India, May 11-14, 2021), 2019. – P. 56-69. DOI:10.1007/978-3-030-16145-3\_5.

6. Attribute-based encryption for fine-grained access control of encrypted data / A. Gianotto. Mode of access: https:// downworthy.snipe.net/. – Date of access: 1.05.2021.

7. Attribute-based access control with efficient revocation in data outsourcing systems / A. Chakraborty et al. // IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (Davis, California, USA, August 18-21, 2016), 2016. – P. 9-16. DOI:10.1109/ASONAM.2016.7752207

8. Secure management of personal health records by applying attribute-based encryption / H.T. Zheng et al. // Symmetry. – 2018. Vol. 10(5), no. 138. – pp. 1-12. DOI:10.3390/sym10050138.

9. Advances in Cryptology / Y. Shen et al. // Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, November, 2014, 2014. – P. 101-110.

10. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption/ C. D. Manning et al. // Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (Baltimore, Maryland, USA, June 22-27, 2014), 2014. – P. 55-60. DOI:10.3115/v1/P14-5010

11. Marneffe, M.-C. De Generating typed dependency parses from phrase structure parses / M.-C. De Marneffe et al. // Proceedings of the Fifth International Conference on Language Resources and Evaluation (Genoa, Italy, May 22-28, 2006), 2006. – P. 449-454.

12. Topics in Cryptology / R. Socher et al. // Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (Seattle, Washington, USA, October 18-21, 2013), 2013. – P. 1631–1642.

13. Privacy preserving system using attribute-based infrastructure / D. Bahdanau et al. ArXiv preprint. – 2014. Mode of access: https://arxiv.org/pdf/1409.0473.pdf. – Date of access: 24.05.2021.

14. Fundamentals of Database Systems (3rd Edition), Pearsson Education [Electronic Resource] / Elamasri R. and Navathe S., 2000. — Mode of access: <http://www.uoitc.edu.iq/institute/Competitive_exam/Database_Systems.pdf>. — Date of access: 22.04.2021.

15. Методы криптоанализа классических шифров [Электронный ресурс] / Ростовцев А. Г. и Михайлова Н. В., 2006. — Режим доступа: http://www.uoitc.edu.iq/institute/Competitive\_exam/Database\_Systems.pdf. — Дата доступа: 22.04.2021.

16. Основы криптографии: Учебное пособие. Теория гибридных алгоритмов [Электронный ресурс] — Режим доступа: http://www.mstu.edu.ru/study/materials/zelenkov/ch\_4\_2.html. — Дата доступа: 22.04.2021.