ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»

Институт электронной техники и приборостроения

Кафедра «Информационная безопасность автоматизированных систем»

Специальность 10.05.03 «Информационная безопасность автоматизированных систем»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Разработка системы передачи сообщений с применением нейрокриптографических алгоритмов

Студент Плешаков Александр Сергеевич

курс 5 группа с-ИБС51

Руководитель

доцент каф. ИБС, к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­\_\_\_\_\_\_\_\_\_ А.С.Розов

(должность, уч. степень, уч. звание) подпись, дата

Допущен к защите

Протокол № 37 от «08» июня 2021 года

Зав. кафедрой «Информационная безопасность автоматизированных систем»

профессор каф. ИБС, д.ф.-м.н., профессор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Б. Байбурин

подпись, дата

Саратов 2021 г

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМЕНИ ГАГАРИНА Ю.А.»

Институт электронной техники и приборостроения

Кафедра «Информационная безопасность автоматизированных систем»

Специальность 10.05.03 «Информационная безопасность автоматизированных систем»

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

Студенту Плешакову Александру Сергеевичу

Тема ВКР: Разработка системы передачи сообщений с применением нейрокриптографических алгоритмов

утверждена на заседании кафедры, протокол № 37 от «08» июня 2021 г.

Дата защиты «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Оценка защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Секретарь ГЭК Мантурова И.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Саратов 2021 г

**Целевая установка и исходные данные**

**Целью** данной выпускной квалификационной работы является анализ способов шифрования, использующих нейрокриптографические алгоритмы, а также разработка программного обеспечения, в основе которого лежит нейрокриптографический алгоритм

**Исходными данными** принципы нейрокриптографии, материалы по теоретическим основам нейрокриптографии, ГОСТ 34.11-2018 определяющий основные математические понятия

|  |  |  |
| --- | --- | --- |
| **№** | **перечень чертежей, подлежащих разработке** | **формат, кол-во** |
| 1 | Слайд «Тема ВКР» | ppt, 1 |
| 2 | Слайд «Актуальность работы» | ppt, 1 |
| 3 | Слайд «Объект и предмет исследования» | ppt, 1 |
| 4 | Слайд «Задача исследования» | ppt, 1 |
| 5 | Слайд «Принцип нейрокриптографического алгоритма» | ppt, 2 |
| 6 | Слайд «Принцип работы RSA» | ppt, 1 |
| 7 | Слайд «Принцип работы программы» | ppt, 2 |
| 8 | Слайд «Результаты работы программы» | ppt, 1 |
| 9 | Слайд «Выводы» | ppt, 1 |
| 10 | Слайд «Спасибо за внимание» | ppt, 1 |

**Руководитель**

доцент каф. ИБС, к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­\_\_\_\_\_\_\_\_\_ А.С. Розов

(должность, ученая степень, уч. звание) подпись, дата

**Содержание расчетно-пояснительной записки**

1.Введение

2. Основные принципы и концепции криптографии

3. Анализ основных алгоритмов шифрования

4. Разработка мессенджера с применением нейрокриптографических алгоритмов

5. Безопасность автоматизированного рабочего места инженера-программиста

6.Заключение

**Основная рекомендуемая литература**

1.Применение искусственных нейронных сетей и системы остаточных классов в криптографии / Н.И. Червяков [и др.] – М.: ФИЗМАТЛИТ, 2012. – 280 с.

2.Фергюсон, Н. Практическая криптография: [пер. с англ.] / Н. Фергюсон, Б. Шнайер – М.: Издательский дом «Вильямс», 2004. – 432 с.

3.Almarimi, A.; Kumar, A.; Almerhag, I. 2008. A New Approach for Data Encryption Using Genetic Algorithms, in Proc. of the International Arab Conference on Information Technology 2008. Zarqa, 5 pp.

**Руководитель**

доцент каф. ИБС, к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­\_\_\_\_\_\_\_\_\_ А.С. Розов

(должность, ученая степень, уч. звание) подпись, дата

**Задание принял к исполнению: \_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Плешаков

подпись, дата

УТВЕРЖДАЮ:

**Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_ Розов А.С.**

**«05» апреля 2021 г.**

КАЛЕНДАРНЫЙ ГРАФИК

работы над ВКР

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | разделы, темы, их содержание | по плану | | фактически | | Отметка о выполнении |
| дата | % | дата | % |
| 1. | Введение | 07.04 | 3% |  |  |  |
| 2. | Раздел 1. Обзор основных принципы и концепции криптографии | 10.04 | 10% |  |  |  |
| 3. | Раздел 2. Анализ основных алгоритмов шифрования | 17.04 | 10% |  |  |  |
| 4. | Раздел 2. Рассмотрение основ разработки безопасных приложений | 21.04 | 12% |  |  |  |
| 5. | Раздел 3. Разработка серверной части приложения | 01.05 | 20% |  |  |  |
| 6. | Раздел 3. Разработка клиентской части приложения | 12.05 | 15% |  |  |  |
| 7. | Раздел 3. Разработка криптографической системы приложения | 22.05 | 15% |  |  |  |
| 9. | Безопасность рабочего места инженера-программиста | 30.05 | 5% |  |  |  |
| 10. | Заключение | 31.05 | 5% |  |  |  |
| 11. | Оформление пояснительной записки | 07.06 | 5% |  |  |  |
| 12. | Предоставление оформленной работы на кафедру | 08.06 |  |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Плешаков

подпись, дата

АННОТАЦИЯ

Данная выпускная квалификационная работа посвящена разработке защищенной системы обмена сообщениями на основе базовых, использующихся на практике, концепций и механизмов защиты информации в подобных системах на основе стека технологий Python.

ABSTRACT

This final qualifying work is devoted to the development of a secure messaging system based on basic, used in practice, concepts and mechanisms for protecting information in such systems based on the Python technology stack.

РЕФЕРАТ

Пояснительная записка содержит 70 листов формата А4, 9 рисунков, 4 таблицы. Использовано 22 литературных источника. Работа выполнена в текстовом редакторе Microsoft Word, файл работы имеет название 161787\_с-ИБС51\_2021\_1.

Ключевые слова: система обмена сообщениями, нейрокриптография, обмен ключами, шифрование, Python.

Объектом исследования выпускной квалификационной работы являются алгоритмы защиты информации в системах обмена сообщениями.

Целью выпускной квалификационной работы является практическое применение основных концепций и методов обеспечения информационной безопасности в мессенджерах.

Для достижения данной цели был проведён обзор основных подходов, применяемых к защите информации, проведен анализ, использующихся в современных системах, концепций и механизмов шифрования, а также разработана система, реализующая алгоритмы нейрокриптогрифического обмена ключами и шифрования с применением алгоритма RSA.

В результате выпускной квалификационной работы была разработана защищенная система обмена сообщениями с использованием языка Python, использующая методы шифрования на основе нейрокриптографического протокола обмена ключами и алгоритма RSA.

Содержание

[Введение 10](#_Toc73974362)

[Раздел 1. Основные принципы и концепции криптографии 11](#_Toc73974363)

[1.1 Симметричные шифрование 14](#_Toc73974364)

[1.2 Ассиметричное шифрование 17](#_Toc73974365)

[1.3 Гибридные шифрование 21](#_Toc73974366)

[1.4 Хеширование 23](#_Toc73974367)

[1.5 Основные виды криптоанализа 25](#_Toc73974368)

[Раздел 2. Анализ основных алгоритмов шифрования 28](#_Toc73974369)

[2.1 Протокол Диффи-Хеллмана 28](#_Toc73974370)

[2.2 RSA 29](#_Toc73974371)

[2.3 Elliptic Curve ECC 31](#_Toc73974372)

[2.4 Data Encryption Standard 32](#_Toc73974373)

[2.5 Advanced Encryption Standard 34](#_Toc73974374)

[2.6 Нейрокриптография. Общие понятия 38](#_Toc73974375)

[Раздел 3. Разработка мессенджера с применением нейрокриптографических алгоритмов 49](#_Toc73974376)

[3.1 Серверная часть 51](#_Toc73974377)

[3.2 Клиентская часть 53](#_Toc73974378)

[3.3 Вспомогательные сущности 54](#_Toc73974379)

[Раздел 4. Безопасность рабочего места инженера-программиста 58](#_Toc73974380)

[4.1 Анализ потенциально опасных и вредных факторов 58](#_Toc73974381)

[4.2 Анализ воздействия вредных и опасных факторов на инженера-программиста 59](#_Toc73974382)

[4.3 Требования к организации рабочих помещений 61](#_Toc73974383)

[4.4 Микроклимат рабочих помещений 61](#_Toc73974384)

[4.5 Требования к уровню шума и вибрации на рабочем месте 63](#_Toc73974385)

[4.6 Требования к степени освещенности 64](#_Toc73974386)

[4.7 Выводы 64](#_Toc73974387)

[Заключение 66](#_Toc73974388)

[Список литературы 67](#_Toc73974389)

# Введение

Информационная безопасность (ИБ) — это набор процессов, методологий и процедур для защиты информации и информационных систем от несанкционированного доступа, использования, модификации или уничтожения. Защита информации в потенциально враждебных средах является решающим фактором в развитии информационных процессов в промышленности, бизнесе и администрации. Криптография - ключевая технология для достижения ИБ в коммуникационных и компьютерных системах, определяемая как обмен данными в смешанный код, который можно расшифровать и отправить по общедоступным и частным сетям.

Криптография представляет собой изучение и практика различных методов безопасной связи и обмена информации в присутствии третьих лиц, называемых противниками. Она занимается разработкой и анализом протоколов, которые не позволяют третьим сторонам злоумышленников получить информацию, совместно используемую двумя объектами, тем самым соблюдая различные аспекты информационной безопасности.

За последние 60 лет нейронные сети привлекли к себе большое внимание как правдоподобная вычислительная модель того, как работает человеческий мозг. Сегодня эта область продолжает быть чрезвычайно активной и привлекает междисциплинарных исследователей из самых разных областей (биология, медицина, психология, физика, математика, информатика и т. Д.). Неудивительно, что исследователи также пытались использовать нейронные сети в криптографии. Нейронная криптография — это новая область, которая стремится объединить криптографию с нейронными сетями для приложений в криптоанализе и шифровании.

# Раздел 1. Основные принципы и концепции криптографии

Криптография является методологической основой современных систем обеспечения безопасности информации в компьютерных системах и сетях. Исторически криптография зародилась как способ скрытой передачи сообщений.

Криптография является совокупностью методов преобразования данных, применяемых для того, чтобы защитить эти данные от потенциальных перехватчиков, сделав их нечитаемыми или лишенными смысла для незаконных пользователей. Такие преобразования обеспечивают решение главных принципов защиты информации.

В современной криптографии можно рассматривать пять основных принципа: конфиденциальность, аутентичность, доступность, целостность и предотвращение отказа от авторства. Для их реализации разработана масса технологий криптографического шифрования, цифровой подписи и систем аутентификации.

* Конфиденциальность представляет собой состояние информации, гарантирующее, что она будет ограничена определенными людьми или местами.
* Подлинность или аутентичность является возможностью подтверждения того, что данные или привилегии, запрашиваемые пользователем, действительно ему доступны.
* Доступность – это обеспечение своевременного и надежного доступа к информации и информационным сервисам.
* Целостность данных — это процесс поддержания и обеспечения точности, непротиворечивости и неизменности информации на протяжении всего жизненного цикла.
* Неотказуемость авторства гарантирует, что стороны, участвующие в коммуникации, не могут отрицать подлинность своих подписей на документе или отправки созданного ими сообщения.

Шифр представляет собой совокупность процедур и правил математических обратимых модификаций информации, задействованных для шифрования и расшифровывания информации по так называемому ключу. Такие преобразования отображают множество возможных изначальных данных на множество возможных зашифрованных данных.

Открытый (исходный) текст — данные или сообщение, несущие смысловую нагрузку, которые нужно защитить от потенциальных перехватчиков.

Шифротекст или криптограмма — данные, полученные после применения к нему воздействия криптосистемы. Криптограмма, содержит исходную информацию в полном объеме, однако последовательность знаков в нем внешне представляется случайной и бессмысленной и не позволяет восстановить исходную информацию без знания ключа шифрования.

Шифрование — процесс применения криптографического преобразования над открытым текстом в результате которого возникает шифрованный текст. В основе такого преобразования лежат алгоритм шифрования и ключ.

Расшифровывание — процесс применения криптографического преобразования обратного шифрованию, в результате которого получается открытый текста.

Ключ шифрования является некоей секретной информацией, используемой любым криптографическим алгоритмом при шифровании или расшифровке секретной информации, постановке и проверке цифровой подписи, вычислении кодов аутентичности. Непосредственный результат шифрования должен зависеть только от ключа при использовании одного и того же алгоритма шифрования. Для современных криптографических алгоритмов, обеспечивающих достаточно надежную защиту секрета, утрата ключа фактически влечет к практической невозможности расшифровать сообщение. И хотя в теории это возможно, такие алгоритмы построены таким образом, что на расшифровку информации на самых мощных вычислительных машинах могут уйти десятилетия.

Ключи шифрования различают в контексте алгоритма, в котором они используются. Выделяют:

* Симметричные ключи представляют собой разновидность ключей шифрования, используемых в симметричных алгоритмах. Такие алгоритмы как правило используются в шифровании и выработке кодов аутентичности. Основополагающим свойством симметричных ключей является то, что для выполнения как прямого, так и обратного криптографического преобразования, иными словами шифрование и расшифровывание, используется один и тот же секретный ключ. В зависимости от алгоритма данное свойство может нарушаться в силу того, что ключ для обратного преобразования легко вычисляется из ключа для прямого преобразования. С одной стороны таким образом достигается высокая конфиденциальность информации, однако с другой стороны, данный подход создаёт весомые проблемы распространения ключей, особенно это заметно в нагруженных системах с большим количеством пользователей.
* Асимметричные ключи представляют собой разновидность ключей шифрования, используемых в асимметричных алгоритмах. Такие алгоритмы как правило используются в шифровании, а также в электронной цифровой подписи. Такие ключи представлены в виде ключевой пары, поскольку состоят из двух ключей. Одна его часть называется закрытым ключом и является фрагментом ключа, к которому имеет доступ лишь его владелец. Сохранение пользователем своего закрытого ключа в тайне гарантирует конфиденциальность зашифрованных сообщений, а также невозможность подделки злоумышленником документа и своей цифровой подписи от имени заверяющего. Другая его часть называется открытым ключом, этот ключ является общедоступным и должен быть опубликован. В его функционал входит шифрование конфиденциальной информации, контроль подлинности цифровой подписи, а также предупреждение мошенничества со стороны заверяющего лица в виде его отказа от факта подписи документа. Зачастую открытый ключ рассчитывается, как результат некоторой функции, зависящей от закрытого ключа, однако знание открытого ключа ни в коем случае не должно давать возможности определения секретного ключа.
* Сеансовые (сессионные) ключи представляют собой разновидность ключей шифрования, вырабатываемые между двумя пользователями, зачастую к функционалу таких ключей относится защита канала связи. Зачастую сеансовый ключ может быть представлен в виде какого-либо общего секрета или информации, вырабатываемой на основе какого-либо потока данных уже используемой в криптосистеме. Существует несколько протоколов выработки сеансовых ключей и общих секретов, среди них, в частности, алгоритм Диффи — Хеллмана.

# Симметричные шифрование

Симметричная криптография, известная также как криптография с секретным ключом — это использование единого общего секрета для обмена зашифрованными данными между сторонами. Шифры в этой категории называются симметричными, потому что в нем используется один и тот же ключ для шифрования и расшифрования данных. Проще говоря, отправитель шифрует данные с помощью пароля, а получатель должен знать этот пароль для доступа к данным.

Симметричные криптосистемы обладают наиболее высокой скоростью преобразования данных. С помощью их применения обеспечиваются и конфиденциальность, и подлинность, и целостность передаваемой информации. Конфиденциальность передачи информации с помощью симметричной криптосистемы зависит от надежности шифра и обеспечения конфиденциальности ключа шифрования.

Симметричное шифрование — это двусторонний процесс. С блоком открытого текста и заданным ключом симметричные шифры всегда будут создавать один и тот же зашифрованный текст. Точно так же использование того же ключа в этом блоке зашифрованного текста всегда будет давать исходный открытый текст. Симметричное шифрование полезно для защиты данных между сторонами с установленным общим ключом, а также часто используется для хранения конфиденциальных данных.



Рис 1. Обобщённая схема симметричной криптосистемы

При использовании симметричной криптосистемы отправитель каким-либо образом генерирует исходное сообщение М, будь то обычное пользовательское общение двух клиентов, либо же взаимодействие двух систем посредством специальных сообщение. Далее это сообщение необходимо передать по незащищенному каналу. Чтобы это сделать клиент использует обратимое преобразование Е и передает ему общий секретный ключ k и сообщение М. В результате преобразования получается криптограмма С, который можно отправлять по незащищенному каналу. Таким образом перехватчик имея зашифрованный текст С не сможет узнать содержания сообщения М. Далее законный получатель расшифровывает сообщение С используя преобразование D, принимающее в качестве параметра общий секретный ключ k, и получает необходимое сообщение М.

Современные симметричные алгоритмы считаются надежными, если отвечают следующим требованиям:

* Отсутствие статистических паттернов в выходных данных. Другими словами, наиболее частотные символы алфавита не должны соответствовать наиболее частотным символам зашифрованного текста.
* Шифр должен быть нелинейным. Другими словами, в зашифрованной информации не должно быть отслеживаемых закономерностей, имея некоторое количество открытых и соответствующих зашифрованных текстов.

Большинство актуальных симметричных шифров для достижения результатов, соответствующих этим требованиям, используют комбинацию операций подстановки (замена фрагментов исходного сообщения, например букв, на другие данные, например цифры, по определенному правилу или с помощью таблицы соответствий) и перестановки (перемешивание частей исходного сообщения по определенному правилу), поочередно повторяя их. Один круг шифрования, состоящий из этих операций, называется раундом.

В зависимости от принципа работы алгоритмы симметричного шифрования делятся на два типа:

* блочные;
* потоковые.

Специфика блочных алгоритмов заключена в том, что шифрование данных происходит блоками фиксированной длины, зависящей от реализации алгоритма. Если все сообщение или его финальная часть меньше размера блока, система дополняет его предусмотренными алгоритмом символами, которые так и называются – дополнением.

К актуальным на сегодняшний день блочным алгоритмам относятся:

* AES
* ГОСТ 28147-89
* RC5
* Blowfish
* Twofish

Потоковое шифрование данных предполагает обработку каждого бита информации с использованием гаммирования, то есть изменения этого бита с помощью, соответствующего ему бита псевдослучайной секретной последовательности чисел, которая формируется на основе ключа и имеет ту же длину, что и шифруемое сообщение. Как правило, биты исходных данных сравниваются с битами секретной последовательности с помощью логической операции XOR (исключающее ИЛИ, на выходе дающее 0, если значения битов совпадают, и 1, если они различаются)[3].

Потоковое шифрование в настоящее время используют следующие алгоритмы:

* RC4
* Salsa20
* HC-256
* WAKE

Имитовставкой является специальный код, присоединяемый к передаваемым данным. Данный код вырабатывается по секретному ключу. С помощью такого преобразования в симметричных системах достигается целостность данных[5]. Имитовставка можно представить в виде некоей разновидности контрольной суммы, т. е. это некоторая эталонная характеристика сообщения, с помощью которой производится проверка целостности информации. Алгоритм генерации имитовставки предоставляет ее зависимость по некоторому сложному криптографическому правилу от каждого бита информации. Получатель сообщения выполняет проверку целостности сообщения с помощью выработки по секретному ключу имитовставки, соответствующей полученному сообщению, со своей стороны, затем она сравнивается с полученным значением имитовставки. Если есть равенство, то делается вывод о том, что информация не была изменена на пути от отправителя к получателю.

# Ассиметричное шифрование

Асимметричная криптография — это ветвь криптографии, где секретный ключ можно разделить на две части: открытый ключ и закрытый ключ. Открытый ключ может быть передан кому угодно, в то время как закрытый ключ должен храниться в секрете. Данный принцип предоставляет масштабируемую систему для использования в очень больших и постоянно расширяющихся средах, где данные часто обмениваются между различными партнерами. Оба ключа математически связаны (оба ключа вместе называются ключевой парой). Также оба ключа необходимы для выполнения операции.

 Рис 2. Обобщённая схема асимметричной криптосистемы

При использовании ассиметричной криптосистемы отправитель каким-либо образом генерирует исходное сообщение М, будь то обычное пользовательское общение двух клиентов, либо же взаимодействие двух систем посредством специальных сообщение. Далее это сообщение необходимо передать по незащищенному каналу. Чтобы это сделать клиент использует обратимое преобразование Е и передает ему открытый ключ k1 и сообщение М. В результате преобразования получается криптограмма С, который можно отправлять по незащищенному каналу. Таким образом перехватчик имея зашифрованный текст С не сможет узнать содержания сообщения М. Далее законный получатель расшифровывает сообщение С используя преобразование D, принимающее в качестве параметра закрытый ключ k2, и получает необходимое сообщение М.

Как и в случае симметричных криптографических систем, с помощью асимметричных криптосистем обеспечивается не только конфиденциальность, но также целостность передаваемой информации любого сообщения за счет формированием цифровой подписи этого сообщения и отправкой в зашифрованном виде сообщения вместе с цифровой подписью. Проверка соответствия подписи полученному сообщению после его предварительного расшифро­вывания представляет собой проверку целостности принятого сообщения.

Асимметричная криптография часто используется для обмена секретным ключом, чтобы подготовиться к использованию симметричной криптографии для шифрования данных. В случае обмена ключами одна сторона создает секретный ключ и шифрует его открытым ключом получателя. Затем получатель расшифрует его своим закрытым ключом. Оставшаяся связь будет осуществляться с секретным ключом, являющимся ключом шифрования. Асимметричное шифрование используется в обмене ключами, безопасности электронной почты, веб-безопасности и других системах шифрования, которые требуют обмена ключами по общедоступной сети.

Также стандарты SSL и TLS используют асимметричный алгоритм на стадии установки соединения или так называемого рукопожатия: с его помощью кодируют и передают ключ от симметричного шифра, которым и пользуются в ходе дальнейшей передачи данных[16].

Преимущества асимметричных криптографических систем перед симметричными криптосистемами:

* в асимметричных криптосистемах решена сложная пробле­ма распределения ключей между пользователями, так как каждый пользователь может сгенерировать свою пару клю­чей сам, а открытые ключи пользователей могут свободно публиковаться и распространяться по сетевым коммуника­циям;
* исчезает квадратичная зависимость числа ключей от числа пользователей; в асимметричной криптосистеме число ис­пользуемых ключей связано с числом абонентов линейной зависимостью (в системе из N пользователей используют­ся 2N ключей), а не квадратичной, как в симметричных системах;
* асимметричные криптосистемы позволяют реализовать про­токолы взаимодействия сторон, которые не доверяют друг другу, поскольку при использовании асимметричных крип­тосистем закрытый ключ должен быть известен только его владельцу.

Недостатки асимметричных криптосистем:

* на настоящий момент нет математического доказательства необратимости используемых в асимметричных алгоритмах функций;
* асимметричное шифрование существенно медленнее сим­метричного, поскольку при шифровании и расшифровке используются весьма ресурсоемкие операции. По этой же причине реализовать аппаратный шифратор с асимметрич­ным алгоритмом существенно сложнее, чем реализовать аппаратно симметричный алгоритм;
* необходимость защиты открытых ключей от подмены.

Наиболее распространенные алгоритмы асимметричного шифрования:

* [RSA](https://encyclopedia.kaspersky.ru/glossary/rsa/) (аббревиатура от Rivest, Shamir и Adelman, фамилий создателей алгоритма) — алгоритм, в основе которого лежит вычислительная сложность факторизации (разложения на множители) больших чисел. Применяется в защищенных протоколах SSL и TLS, стандартах шифрования, например в PGP и S/MIME, и так далее. Используется и для шифрования данных, и для создания цифровых подписей.
* DSA (Digital Signature Algorithm, «алгоритм цифровой подписи») — алгоритм, основанный на сложности вычисления дискретных логарифмов. Используется для генерации цифровых подписей. Является частью стандарта DSS (Digital Signature Standard, «стандарт цифровой подписи»).
* Схема Эль-Гамаля — алгоритм, основанный на сложности вычисления дискретных логарифмов. Лежит в основе DSA и устаревшего российского стандарта ГОСТ 34.10–94[22]. Применяется как для шифрования, так и для создания цифровых подписей.
* ECDSA (Elliptic Curve Digital Signature Algorithm) — алгоритм, основанный на сложности вычисления дискретного логарифма в группе точек эллиптической кривой. Применяется для генерации цифровых подписей, в частности для подтверждения транзакций в криптовалюте Ripple.

Фактически, любые алгоритмы обмена ключами уязвимы перед атаками «человек посередине». Для того, чтобы проверить, что абонент является именно тем, за кого себя выдает, используются некоторые решения.

Инфраструктура открытых ключей – набор служб, средств и компонентов, использующийся для поддержки криптографических задач на основе систем с открытым ключом.

Такая инфраструктура, выступает третьим лицом и может подтверждать (или опровергать) принадлежность открытого ключа какому-либо лицу. Выделяет несколько типов схем.

* Иерархическая схема. Существует некий корневой центр сертификации, которому по умолчанию доверяют все. Он выдает сертификаты (в том числе и сертификаты на выдачу сертификатов) неким промежуточном центрам сертификации. Внизу древа находятся выпускающие центры сертификации, работающие с конечными пользователями. Пользователь доверяет своему центру, далее по цепочке идет проверка до корневого.
* Сетевая схема. Схема представляет собой сеть доверия, состоящую из центров сертификации, которые связаны друг с другом равноправными отношениями. Другими словами, нет головного центра, а участники доверяют только рядом стоящим. Пользователь доверяет только тому центру, который выдал ему сертификат.

Таким образом, данная инфраструктура может использоваться, например, как третье лицо, для подтверждения принадлежности открытых ключей в алгоритме RSA.

# Гибридные шифрование

В чистом виде симметричные криптосистемы и криптосистемы с открытым ключом почти не используются. Гораздо эффективнее комбинировать такие системы ради использования их преимуществ и перекрытия недостатков. Именно такие системы и называются гибридными. Гибридная схема шифрования сочетает в себе удобство схемы асимметричного шифрования с эффективностью схемы симметричного шифрования.

Следует сразу же оговориться, что гибридное шифрование не является «отдельным методом», как симметричное или асимметричное, в нем используются все преимущества обоих методов и создается синергия надежных систем шифрования.

Каждый из алгоритмов шифрования имеет свои недостатки. Например, метод симметричного шифрования отлично подходит для быстрого шифрования больших объемов данных. Но он не обеспечивает проверку личности, что является необходимым, когда речь заходит о безопасности в Интернете. Также один из важных недостатков симметричных схем шифрования заключается в необходимости обеспечения защищенного канала между лицами или сервисами, осуществляющими обмен информацией. Что в свою очередь исключает смысл шифрования.

С другой стороны, асимметричное шифрование предоставляет доступ к данным предполагаемого получателя. Однако эта проверка делает процесс шифрования очень медленным. Но в случае асимметричного шифрования нет необходимости в обеспечении такого канала, однако при этом сильно падает производительность.

В случае гибридных схем шифрования асимметричные система шифрования используется для обмена ключами, использующиеся в симметричных системах. Что в свою очередь отметает необходимость в создании защищенного канала и позволяет воспользоваться скоростью симметричных систем без заведомого обмена ключами.

Метод гибридного шифрования используется в SSL/TLS сертификатах во время последовательной связи между серверами и клиентами (веб-браузерами) в процессе, известном как “TLS handshake”. Сначала проверяется личность обеих сторон с использованием закрытого и открытого ключа. После того, как обе стороны подтвердят свою личность, шифрование данных происходит посредством симметричного шифрования с использованием эфемерного (сеансового) ключа. Это обеспечивает быструю передачу большого объема данных.

Большинство современных SSL сертификатов используют гибридный метод: асимметричное шифрование для аутентификации и симметричное шифрование для конфиденциальности. Такой сертификат не дает мошенникам перехватить или подменить личные данные пользователей: контактную информацию, номера банковских карт, логины, пароли, адреса электронной почты и т.д.

# Хеширование

Криптографические хеш-функции — незаменимый и повсеместно распространенный инструмент, используемый для выполнения ряда важных задач, таких как поддержание целостности данных, аутентификацию, защиту файлов, а также обнаружение зловредного программного обеспечения. На практике представлено много алгоритмов хеширования, различных в реализации и таких свойствах как криптостойкостью, сложностью, разрядностью и других. Полагают, что принцип хеширования принадлежит одному из сотрудников IBM, который появился около 50 лет назад и с тех пор не претерпел принципиально важных изменений. Однако в наши дни хеширование обрело массу новых свойств и используется в очень многих областях информационных технологий.

Криптографическая хеш-функция, чаще называемая просто хешем, — это математический алгоритм, преобразовывающий произвольный массив данных в состоящую из букв и цифр строку фиксированной длины[1]. Причем при условии использования того же типа хеша длина эта будет оставаться неизменной, вне зависимости от объема вводных данных.

Криптостойкой хеш-функция может быть только в том случае, если выполняются главные требования:

* Детерминированние. Это означает, что независимо от того, сколько раз преобразовывается определенный входные даные через хэш-функцию, всегда будет получен один и тот же результат.
* Быстрое вычисление. Хэш-функция должна быть способна быстро возвращать хэш-вход. Если процесс недостаточно быстрый, система просто не будет эффективна.
* Сложность обратного вычисления. Сложность обратного вычисления означает, что с учетом H (A) невозможно определить A, где A – вводимые данные и H(А) – хэш.
* Небольшие изменения в вводимых данных значительно изменяют хэш. Одно из незаменимых свойств хеширования — его уникальность. Одно и то же значение хеша не может использоваться для разного текста. Малейшее изменение в тексте полностью изменит значение хеша. Это называется эффектом лавины.
* Коллизионная устойчивость. Учитывая два разных типа исходных данных A и B, где H (A) и H (B) являются их соответствующими хэшами, для H (A) не может быть равен H (B). Это означает, что, по большей части, каждый вход будет иметь свой собственный уникальный хэш.

Стоит заметить, что под данные требования формально не подпадает ни один из современных алгоритмов хеширования, поскольку нахождение обратного хешу значения — вопрос лишь вычислительных мощностей, доступных при криптоанализе. По факту же в случае с некоторыми особо продвинутыми алгоритмами этот процесс может занимать очень много времени.

Одним из первых стандартов алгоритма хеширования был MD5 hash, который широко использовался для проверки контрольных сумм файлов, другими словами, для поддержания целостности, а также хранения хешированных паролей в базах данных веб-приложений[16]. Его функциональность довольно проста, так как она выводит фиксированную 128-битную строку для каждого входа и использует тривиальные однонаправленные операции в нескольких раундах для вычисления детерминированного результата. Его короткая выходная длина и простота операций сделали MD5 очень легким для взлома и восприимчивым к атаке «дня рождения». Фиксированные ограничения на выход означают, что существует фиксированная степень перестановок, на которых можно найти коллизию. Стоит отметить, что данный недостаток присущ не только алгоритму MD5, а всем современным алгоритмам хеширования.

# Основные виды криптоанализа

Появление новых криптографических алгоритмов приводит к разработке методов их взлома. Результатом возникновения каждого нового метода криптоанализа является пересмотр оценок безопасности шифров, что, в свою очередь, влечет необходимость создания более стойких шифров. Таким образом, исторические этапы развития криптографии и криптоанализа неразрывно связаны.

Попытка криптоанализа называется атакой. Результаты криптоанализа могут варьироваться по степени практической применимости. Так, криптограф Л. Кнудсен предлагает следующую классификацию успешных исходов криптоанализа блочных шифров в зависимости от объёма и качества секретной информации, которую удалось получить:

* полный взлом – криптоаналитик извлекает секретный ключ;
* глобальная дедукция – криптоаналитик разрабатывает функциональный эквивалент исследуемого алгоритма, позволяющий зашифровывать и расшифровывать информацию без знания ключа;
* Частичная дедукция – криптоаналитику удается расшифровать или зашифровать некоторые сообщения.
* Информационная дедукция - криптоаналитик получает некоторую информацию об открытом тексте или ключе.

Однако взлом шифра совсем не обязательно подразумевает обнаружение способа, применимого на практике для восстановления открытого текста по перехваченному зашифрованному сообщению[10]. В научной криптологии другие правила. Шифр считается взломанным, если в системе обнаружено слабое место, которое может быть использовано для более эффективного взлома, чем метод полного перебора ключей («brute-force approach»).

Под взломом понимается лишь подтверждение наличия уязвимости криптоалгоритма, свидетельствующее о том, что свойства надёжности шифра не соответствуют заявленным характеристикам. Как правило, криптоанализ начинается с попыток взлома упрощённой модификации алгоритма, после чего результаты распространяются на полноценную версию.

Рассмотрим некоторые виды атак:

* Атака по ключам

Одной из причин ненадёжности криптосистем является использование слабых ключей. Фундаментальное допущение криптоанализа, впервые сформулированное О. Кирхгоффом, состоит в том, что секретность сообщения всецело зависит от ключа, т.е. весь механизм шифрования, кроме значения ключа, известен противнику. Данная атака основана как раз на таком несовершенстве ключей.

Слабый ключ – это ключ, не обеспечивающий достаточного уровня защиты или в работе которого, можно найти некие закономерности, которые могут быть взломаны. Обычно считается, что алгоритм шифрования должен по возможности не иметь слабых ключей. Если это невозможно, то количество слабых ключей должно быть минимальным, чтобы уменьшить вероятность случайного выбора одного из них[1]. Тем не менее, все слабые ключи должны быть заранее известны, чтобы их можно было отбраковать в процессе создания ключа.

* Метод «встречи посередине»

Другой популярный метод криптоанализа – алгоритм «встречи посередине» – поддается эффективному распараллеливанию. Например, для логарифмирования в группе порядка p при параллельной работе n процессоров, где n×p, время работы алгоритма уменьшается в n раз.

Данный метод криптоанализа основан на «парадоксе дней рождения». Пусть нам нужно найти ключ k по известному открытому тексту x и криптограмме y. Если множество ключей криптоалгоритма замкнуто относительно композиции, т.е. для любых ключей k′ и k′′ найдется ключ k такой, что результат шифрования любого текста последовательно на k′ и k′′ равен результату шифрования этого же текста на k, т.е. Ek′′(Ek′ , x) = Ek(x), то можно воспользоваться этим свойством. Поиск ключа k сведем к поиску эквивалентной ему пару ключей k′, k′′. Для текста x построим базу данных, содержащую случайное множество ключей k′ и соответствующих криптограмм w = Ek(x), и упорядочим её по криптограммам w.

Затем подбираем случайным образом ключи k′′ для расшифровки текстов y и результат расшифрования v = Ek′′(y) сравниваем с базой данных. Если текст v окажется равным одной из криптограмм w, то ключ k′k′′ эквивалентен искомому ключу k.

* Атаки по сторонним, или побочным, каналам

Такие атаки используют информацию, которая может быть получена с устройства шифрования и не является при этом ни открытым текстом, ни криптограммой. Такие атаки основаны на корреляции между значениями физических параметров, измеряемых в разные моменты во время вычислений, и внутренним состоянием вычислительного устройства, имеющим отношение к секретному ключу. Этот подход менее обобщённый, но зачастую более мощный, чем классический криптоанализ.

В последние годы количество криптографических атак, использующих слабости в реализации и размещении механизмов криптоалгоритма, резко возросло. Противник может замерять время, затрачиваемое на выполнение криптографической операции, анализировать поведение криптографического устройства при возникновении ошибок вычисления. Другой подход предполагает отслеживание энергии, потребляемой смарткартой в процессе выполнения операций с секретным ключом (например, расшифрования или генерации подписи). Побочную информацию собрать несложно – сегодня выделено более десяти побочных каналов, в т.ч. электромагнитное излучение, ошибки в канале связи, кэшпамять и световое излучение.

# Раздел 2. Анализ основных алгоритмов шифрования

Сегодня алгоритмы шифрования данных находят широкое применение в передачах по протоколу передачи файлов (FTP) и компьютерных системах для обеспечения защищенных передач. Когда алгоритмы используются для передачи, информация изначально преобразуется в нечитаемый зашифрованный текст и отправляется в этом формате, после чего получатель использует секретный ключ или пароль для декодирования зашифрованного текста в его исходный формат. Если злоумышленник получит доступ к файлу до того, как достигнет конечного компьютера, он не сможет прочитать его, поскольку он представляет собой нечитаемый, и на первый взгляд бессмысленный набор символов.

Такие алгоритмы разрабатываются учеными математиками или целыми коллективами сотрудников компаний или научных центров. Этот список распространенных алгоритмов шифрования включает RSA, ECC, 3DES, AES и т. д. Далее к рассмотрению представляются некоторые представители подобных алгоритмов.

# Протокол Диффи-Хеллмана

Алгоритм был разработан Уитфилдом Диффи и Мартином Хеллманом, а также, независимо от них, Ральфом Мерклом. Появление в 1970-х годах протокола Диффи-Хеллмана (или Диффи-Хеллмана-Меркла) имеет большое значение в мире криптографии, ведь протокол решил большую проблему XX века – передача ключа по незащищенному каналу[4]. Эта работа заложила начало криптографии с открытым ключом.

Пусть пользователей зовут Алиса и Боб. Сначала они договариваются о двух больших простых числах *p* и *g*, эти числа являются публичными.

Далее Алиса генерирует большое случайное число *a*, Боб в свою очередь генерирует большое случайное число *b*.

Далее они должны произвести следующие вычисления. Алиса считает

*A=ga(mod p)* (1)

и отправляет это Бобу*.* Боб считает

*B=gb(mod p)* (2)

и отправляет это Алисе.

Теперь они могут посчитать их общий секретный ключ, который равен *K=gab(mod p)*. Алисе достаточно посчитать

*K=Ba(mod p)=(gb)a(mod p)*, (3)

а Бобу

*K=Ab(mod p)=(ga)b(mod p)*. (4)

Криптоанализ данного алгоритма относится к так называемым задачам дискретного логарифмирования. На данный момент такая задача является вычислительно сложной, то есть нет алгоритма, способного ее решить за полиномиальное время.

В то же время данный алгоритм остается уязвимым к атаке «человек посередине», поскольку участники обмена информации не в состоянии достоверно выяснить с кем они общаются. Злоумышленник в свою очередь может выдавать себя за участников общения и незаметно для них изменять отправляемую информацию.

# RSA

Идея RSA основана на том, что большое целое число сложно разложить на множители. Открытый ключ состоит из двух чисел, где одно число является умножением двух больших простых чисел. И закрытый ключ также является производным от тех же двух простых чисел. Поэтому, если кто-то может факторизовать большое число, закрытый ключ будет скомпрометирован. Следовательно, надежность шифрования полностью зависит от размера ключа, и, если мы удвоим или утроим размер ключа, надежность шифрования возрастет в геометрической прогрессии[1]. Ключи RSA обычно могут иметь длину 1024 или 2048 бит, однако 1024-битные ключи на сегодняшний день уже взломаны.

Пусть пользователей зовут Алиса и Боб. Сначала участники должны произвести генерацию ключей. Каждый из участников предпринимают следующие действия:

Необходимо выбрать 2 случайных простых больших числа и .

Далее высчитывается:

и . (5)

Затем необходимо выбрать некоторое целое число , такое, что:

(6)

Далее, найти , зачастую это реализуется с помощью расширенного алгоритма Евклида, такое, что:

(7)

Получаем пару ключей:

(8)

(9)

Алиса и Боб получили пары . Далее стороны обмениваются открытыми ключами.

Если Алиса хочет отправить сообщение Бобу, то сначала ей необходимо представить его в виде наборов чисел, такие что *m < nb*. Для этого сообщение разбивается на блоки и представляется в виде шестнадцатеричных чисел. Далее Алиса производит следующее вычисление и отправляет Бобу число :

(10)

Боб, получив , расшифровывает сообщение:

(11)

(12)

Если Алиса хочет подписать некий документ, то ей необходимо вычислить подпись T, используя свой закрытый ключ.

(13)

Боб получает от Алисы документ t и подпись T. Чтобы проверить, что именно Алиса подписала документ, Боб, имея доступ к открытому ключу Алисы, вычисляет:

(14)

(15)

Есть преимущества и недостатки алгоритма RSA. К преимуществам относятся: алгоритм RSA безопасен и надежен для пользователей благодаря использованию “сложной” задачи, не имеющей полиномиальное время решения, данный алгоритм включает факторизацию простых чисел, которые трудно разложить на множители. Более того, алгоритм RSA использует открытый ключ для шифрования данных, и этот ключ известен всем, поэтому поделиться открытым ключом легко.

К недостаткам можно отнести; Алгоритм RSA может быть очень медленным в тех случаях, когда на одном компьютере необходимо зашифровать большие данные. Требуется третья сторона для проверки надежности открытых ключей. Данные, передаваемые через алгоритм RSA, могут быть скомпрометированы через посредников, которые могут вмешаться в систему открытых ключей.

# ****Elliptic Curve ECC****

В поисках более надежных функций с потайным входом, в середине 80-х годов криптографы пришли к задействованию ветви математики, посвященной эллиптическим кривым.

Эллиптическая кривая определяется уравнением, которое выглядит следующим образом: y2=x3+ax+b

Вторым объектом является циклическая подгруппа над конечным полем. В алгоритме ECC используются следующие параметры:

* Простое число *p*, определяющее размерность конечного поля;
* Коэффициенты *a* и *b* уравнения эллиптической кривой;
* Базовая точка *G*, генерирующая уже упомянутую подгруппу;
* Порядок *n* подгруппы;
* Кофактор *h* подгруппы.

В итоге, набор параметров для наших алгоритмов представляется шестеркой (*p, a, b, G, n, h*)

Точки эллиптической кривой принадлежат конечному полю Fp, где *p* это достаточно большое простое число. Итак, у нас есть множество целых чисел по модулю *p*, где возможны такие операции как сложение, вычитание, умножение, взятие обратного.

Так как кривая симметрична относительно оси х, для любой точки P мы можем взять −P и получить точку, противоположную ей. Мы сразу оговариваем, что точка −O соответствует нулю, то есть −O будет просто O.

Сложение точек на кривой определено так, что, зная точки P и Q, можно нарисовать прямую, проходящую через обе этих точки, а также третью - R, так что P+Q=−R и P+Q+R=0.

Умножение на скаляр определяется достаточно очевидно: n⋅P=P+P+P+⋯+P.

Односторонняя функция с потайным входом в данном случае опирается на задачу дискретного логарифма для эллиптических кривых, а не на факторизацию, как в случае с RSA[5]. Проблема дискретного логарифма в данном случае формулируется так: если известны P и Q, то как найти k, такое что Q=k⋅P?

И задача факторизации (лежащая в основе RSA), и дискретного логарифма для эллиптических кривых (являющаяся фундаментом ECDSA и ECDH) считаются трудными - другими словами, неизвестно алгоритмов для решения этих задач за полиномиальное время при заданной длине ключа.

# Data Encryption Standard

Стандарт шифрования данных DES (Data Encryption Standard) – блочный шифр с симметричными ключами, разработан Национальным Институтом Стандартов и Технологии (NIST – National Institute of Standards and Technology).

Данный алгоритм изначально предназначался для использования в государственных и правительственных учреждениях США для защиты от несанкционированного доступа важной, но несекретной информации. Алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 году был одобрен ANSI. Фактически с этого момента DES превращается в стандарт не только по названию. Появляются программное обеспечение и специализированные микроЭВМ, предназначенные для шифрования/расшифрования информации в сетях передачи данных и на магнитных носителях. К настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации.

Для шифрования DES принимает 64-битовый открытый текст и порождает 64-битовый зашифрованный текст и наоборот, получив 64 бита зашифрованного текста, он выдает 64 бита расшифрованного[5]. В обоих случаях для шифрования и дешифрования применяется один и тот же 56-битовый ключ.

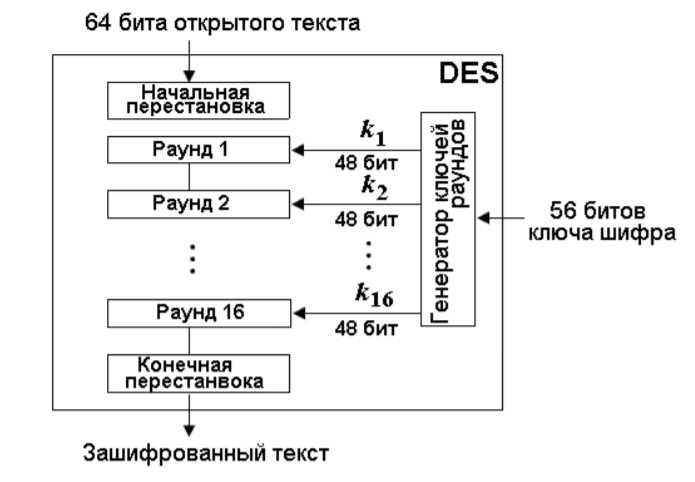


Рис 3. Структура DES

Процесс шифрования состоит из двух перестановок, которые называют начальной и финальной (конечной) перестановками, и 16 раундов Фейстеля. Каждый раунд использует различные сгенерированные 48-битовые ключи.

На вход каждой из них поступает 64 бита, которые затем переставляются в соответствии с заданными таблицами. Эти перестановки взаимно обратны. Другими словами, 58-й бит на входе начальной перестановке переходит в 1-ую позицию на выходе из нее. А финальная перестановка 1-ый входной бит переведет в 58-ую позицию на выходе.

DES использует 16 раундов. Каждый раунд DES применяет шифр Фейстеля, как это показано на рисунке.

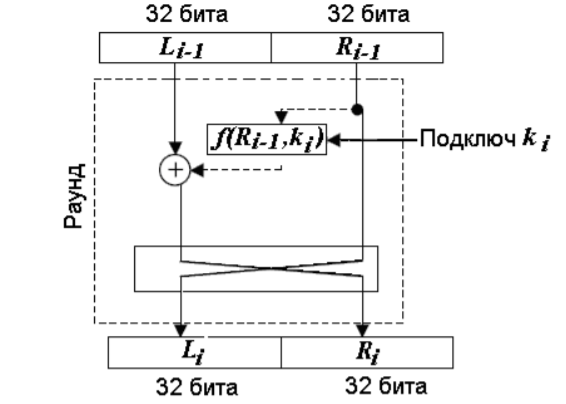


Рис 4. Раунды DES

Раунд принимает полублоки Li−1 и Ri−1 от предыдущего раунда (или начального блока перестановки) и создает полублоки Li и Ri для входа в следующий раунд (или конечный блок перестановки). Все необратимые элементы сосредоточены в функции *f*(Ri-1, ki).

Основные достоинства алгоритма DES:

* используется только один ключ длиной 56 битов;
* зашифровав сообщение с помощью одного пакета, для расшифровки вы можете использовать любой другой;
* относительная простота алгоритма обеспечивает высокую скорость обработки информации;
* достаточно высокая стойкость алгоритма.

# Advanced Encryption Standard

Advanced Encryption Standard (AES) представляет собой один из наиболее широко используемых и наиболее безопасных алгоритмов шифрования на сегодняшний день. Он общедоступный, то есть за его реализацию не нужно платить создателю, поскольку с этого алгоритма сняты патентные ограничения. NSA использует именно этот шифр для обеспечения конфиденциальности документов с пометкой «Совершенно секретно».

Данный алгоритм шифрования обладал очевидными преимуществами, благодаря которым он был официально анонсирован в 2001 году новым стандартом шифрования AES[3]. Алгоритм в процессе работы использует несколько подстановок, перестановок и линейных преобразований, применяемые на блоках данных длинной по 16 байтов. Как и в случае алгоритма DES эти операции повторяются несколько раз и именуются «раундами». На кадом раунде работы алгоритма включается в вычисления новый уникальный ключ, рассчитываемый из ключа шифрования. Благодаря его блочной структуре, AES ведет себя таким образом, что изменение отдельного знака либо в ключе, либо в блоке исходной информации приводит к совершенно другому блоку шифрограммы, что дает ему явное преимущество перед традиционными потоковыми шифрами. Наконец, разница между AES-128, AES-192 и AES-256 — это длина ключа: 128, 192 или 256 бит - все радикальные улучшения по сравнению с более коротким 56-битным ключом DES. В качестве иллюстрации: взлом 128-разрядного ключа AES с помощью современного суперкомпьютера займет больше времени, чем предполагаемый возраст вселенной. На сегодняшний AES остается предпочтительным стандартом шифрования для правительств, банков и систем высокой безопасности по всему миру.

Как отмечалось, в рассматриваемой версии алгоритма AES-128 ключ шифра состоит из 128 битов, поделенных на 16 частей. Эти данные фиксируются в матрице InputKey. Описанная матрица разработана таким образом, что каждый ее столбец образует слово, т.е. фактически ключ шифра – это четыре слова w0 w1 w2 w3, где w0 = k0 k1 k2 k3, и т.д.

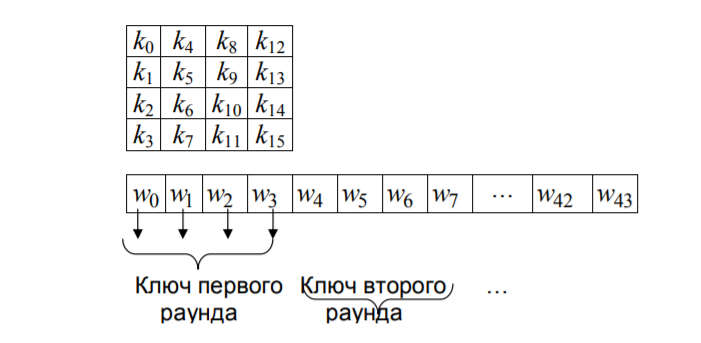
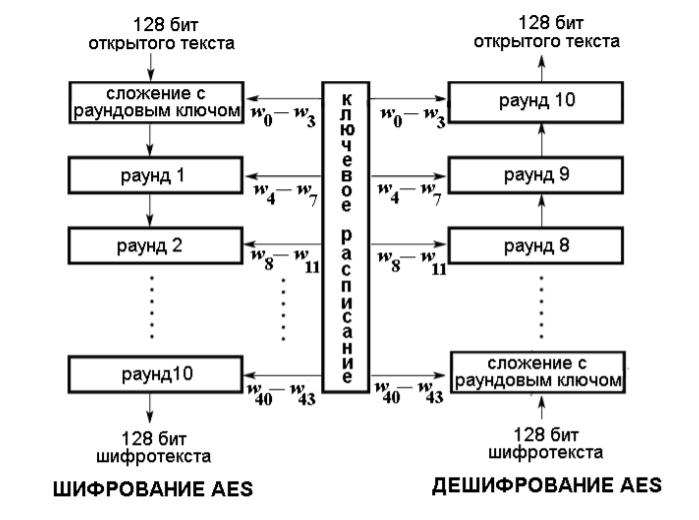


Рис 5. Структура ключа AES

Из этих слов с помощью специального алгоритма (о нем позже) образуется последовательность из 44 слов: w0, w1, w2,...,w43. Каждое из которых представлено в виде 32 бит. Каждый новый раунд шифрования используются по 4 слова этой последовательности. Так и образуется раундовый ключ.

AES 128 выполняет в общей сложности 10 раундов для шифрования данных. Первые девять раундов выполняют вышеупомянутые операции в заданном порядке, но в десятом раунде операция столбцов смешивания опущена. Как только все это будет сделано, ваши первые 128 бит данных будут зашифрованы.

В случае AES 128 и 256 из-за увеличения размера ключа количество раундов увеличивается до 12 и 14 соответственно[3]. Это увеличение количества раундов делает эти версии AES еще сильнее и их сложнее сломать.

 Рис. 6 Схема преобразования данных

Раунд состоит из 4 различных преобразований:

* SubBytes – побайтовая подстановка в S-боксе с фиксированной таблицей замен;
* ShiftRows – побайтовый сдвиг строк матрицы State на различное количество байт;
* MixColumns – перемешивание байт в столбцах;
* AddRoundKey – сложение с раундовым ключом (операция XOR).

В соответствии с оценками разработчиков шифр устойчив против таких видов криптоаналитических атак:

* дифференциального криптоанализ;
* линейного криптоанализ;
* криптоанализ на основе связанных ключей (слабых ключей в алгоритме нет).

Единственный работающий способ взлома шифра AES – это атаки по побочным каналам. Такие атаки не связаны с математическими особенностями AES, а используют определённые особенности реализации систем, использующих шифр, с целью раскрыть частично или полностью секретные данные, в том числе ключ.

Алгоритм обладает не только очень высокой защищенностью, но и очень высокой скоростью шифрования. Программная реализация на машине с частотой 2 ГГц позволяет шифровать данные со скоростью 700 Мбит/с. Такой скорости достаточно для шифрации видео в формате MPEG-2 в реальном масштабе времени. Аппаратные реализации работают еще быстрее

# Нейрокриптография. Общие понятия

В последние годы криптография как приложение искусственных нейронных сетей стала более привлекательной и более широко изучаемой. О возможности применения нейронных сетей в криптографии заговорили спустя 30 лет, после их появления. В 1995 году идеи впервые были озвучены Себастьяном Дорленсом[2]. Первоначально вдохновленные нейробиологией для имитации нейронных структур мозга, искусственные нейронные сети использовались для решения проблем, аналитические решения которых не дали результатов. Это мощные инструменты для автоматического поиска решения путем расчета соответствующих параметров (весов), чтобы обеспечить совместимость одной системы с другой. Они могут быть использованы для решения задач обмена ключами, генерации псевдослучайных чисел и хеширования.

Искусственная нейронная сеть (ИНС) — математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

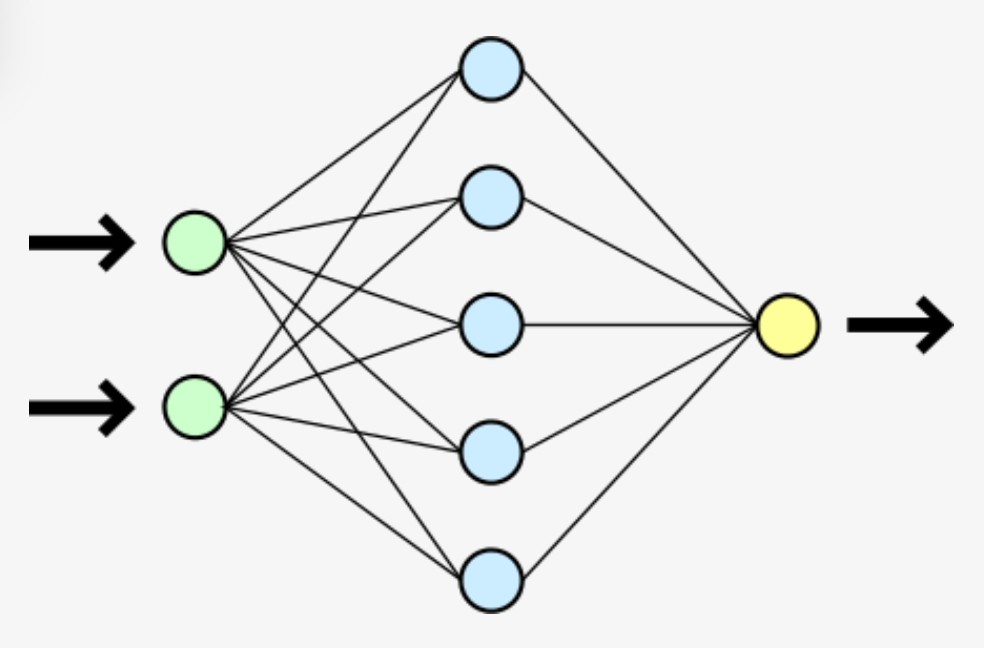


Рис 7. Структура простейшей нейронной сети

Нейронные сети возникли из исследований в области искусственного интеллекта, а именно, из попыток воспроизвести способность биологических нервных систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга[4]. Основной областью исследований по искусственному интеллекту в 60-е - 80-е годы были экспертные системы. Такие системы основывались на высокоуровневом моделировании процесса мышления (в частности, на представлении, что процесс нашего мышления построен на манипуляциях с символами). Скоро стало ясно, что подобные системы, хотя и могут принести пользу в некоторых областях, не ухватывают некоторые ключевые аспекты человеческого интеллекта. Согласно одной из точек зрения, причина этого состоит в том, что они не в состоянии воспроизвести структуру мозга. Чтобы создать искусственных интеллект, необходимо построить систему с похожей архитектурой.

Мозг состоит огромного числа нейронов, соединенных многочисленными связями Нейроны — это специальная клетки, способные распространять электрохимические сигналы. Нейрон имеет ветвистую структуру получения информации (дендриты), ядро и также разветвленный выход (аксон). Аксоны - клетки соединяются с дендритами других клеток с помощью синапсов. При активации нейрон посылает электрохимический сигнал по своему аксону. Через синапсы этот сигнал достигает других нейронов, которые могут в свою очередь активироваться[8]. [Нейрон](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neuron) активируется тогда, когда суммарный уровень сигналов, пришедших в его ядро из входящих сигналов, превысит порог активации.

Интенсивность сигнала, получаемого нейроном, очень зависит от активности синапсов. Каждый синапс имеет протяженность, и специальные химические вещества передают сигнал вдоль него. Один из самых авторитетных исследователей нейросистем, Дональд Хебб, высказал постулат, что обучение заключается в первую очередь в изменениях "силы" синаптических связей[11]. Например, в классическом опыте Павлова, каждый раз непосредственно перед кормлением собаки звонил колокольчик, и собака быстро научилась связывать звонок колокольчика с пищей. Синаптические связи между участками коры головного мозга, ответственными за слух, и слюнными железами усилились, и при возбуждении коры звуком колокольчика у собаки начиналось слюноотделение.

Таким образом, будучи построен из очень большого числа совсем простых элементов (каждый из которых берет взвешенную сумму входных сигналов и в случае, если суммарный вход превышает определенный уровень, передает дальше двоичный сигнал), мозг способен решать чрезвычайно сложные задачи. Разумеется, мы не затронули здесь многих сложных аспектов устройства мозга, однако интересно то, что искусственные [нейронные сети](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural%20Networks) способны достичь замечательных результатов, используя модель, которая ненамного сложнее, чем описанная выше.

Чтобы отразить суть биологических нейронных систем, определение искусственного нейрона дается следующим образом:

* Он получает входные сигналы (исходные данные либо выходные сигналы других нейронов нейронной сети) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенную интенсивность (или вес); этот вес соответствует синаптической активности биологического нейрона. С каждым нейроном связано определенное пороговое значение. Вычисляется взвешенная сумма входов, из нее вычитается пороговое значение и в результате получается величина активации нейрона (она также называется постсинаптическим потенциалом нейрона - PSP).
* Сигнал активации преобразуется с помощью функции активации (или передаточной функции) и в результате получается выходной сигнал нейрона.

Как правило, нейронная сеть используется тогда, когда неизвестен точный вид связей между входами и выходами, - если бы он был известен, то связь можно было бы моделировать непосредственно. Другая существенная особенность нейронных сетей состоит в том, что зависимость между входом и выходом находится в процессе обучения сети. Для обучения нейронных сетей применяются алгоритмы двух типов: управляемое ("обучение с учителем") и не управляемое ("без учителя").

Для того чтобы как-то систематизировать уже имеющиеся и будущие нейросети, делаются попытки их классификации. Далее приведено современное представление о такой классификации:

* Классификация по типу входных данных: аналоговые (на входе действительные числа), двоичные (на входе двоичные числа) и образные (на входе знаки, иероглифы, символы) нейронные сети.
* Классификация по характеру обучения: обучение с учителем (выходное пространство решений нейронной сети известно), обучение без учителя (выходное пространство решений формируется только на основе входных воздействий; такие сети называют самоорганизующимися); обучение с подкреплением (используется система назначения штрафов и поощрений, получаемых в результате взаимодействия ИНС со средой).
* Классификация по характеру настройки синапсов: сети с фиксированными связями (весовые коэффициенты нейронной сети выбираются сразу, исходя из условий задачи), сети с динамическими связями (у этих сетей в процессе обучения происходит настройка синаптических связей).
* Классификация по времени передачи сигнала: синхронные сети (время передачи для каждой синаптической связи равно либо нулю, либо фиксированной постоянной), асинхронные сети (время передачи для каждой связи между элементами свое, но тоже постоянное).
* Классификация по характеру связей: сети прямого распространения (все связи направлены строго от входных нейронов к выходным), рекуррентные сети (сигнал с выходных нейронов или нейронов скрытого слоя частично передается обратно на входы нейронов входного слоя), рекуррентная сеть Хопфилда (фильтрует входные данные, возвращаясь к устойчивому состоянию и, таким образом, позволяет решать задачи сжатия данных и построения ассоциативной памяти), двунаправленные сети (между слоями существуют связи как в направлении от входного слоя к выходному, так и в обратном).

Говоря о криптографии, алгоритмы, основанные на нейронных сетях, основываются не на вычислительной сложности задачи, а на взаимном обучении двух сетей. Это выделяет данный подход и дает некоторые преимущества. Из числа таких преимуществ стоит сразу отметить стойкость к криптоанализу с использованием квантовых инструментов. Так же в числе этих свойств самообучение, взаимное обучение, стохастическое поведение и низкая чувствительность к шуму.

В январе 2002 года физики Кантер, Кинзель и Кантер предложили новый протокол обмена ключами между двумя сторонами A и B. Он использует новое понятие хаотической синхронизации, которое позволяет двум слабо взаимодействующим хаотическим системам сближаться, даже если каждая из них (рассматриваемая по отдельности) продолжает хаотично двигаться.

Кроме всего прочего, одной из важных особенностей нейрокриптографии является то, что ее реализация может быть основана на древовидных машинах четности. ДМЧ на самом деле представляет собой нейронную сеть с прямой связью, в которой нейроны входного слоя построены по модели МакКуллоха-Питтса. На втором уровне в сети представлены нейроны с определенными функциями активации. Результатом выходного нейрона является результат всей сети. Каждая сеть описывается тремя параметрами: количеством скрытых нейронов - K, количеством входных нейронов, подключенных к каждому скрытому нейрону - N, и максимальным значением веса {-L, ... L}.

(16)

Сеть состоит из K\*N случайных входных элементов xij = ± 1, j = 1, ..., N,

(17)

K двоичных скрытых единиц σi = ± 1, i = 1, ..., K,

(18)

(19)

и одного двоичного модуля вывода . Значение выходного нейрона вычисляется путем перемножения значений всех скрытых нейронов, а следовательно, также является бинарным.

(20)

Основной концепцией синхронизации машин четности является взаимное обучение. Процесс синхронизации начинается со случайно заданных входных векторов и случайно выбранных весовых коэффициентов. Обновление весов происходит только в том случае, если выходные значения двух машин равны.

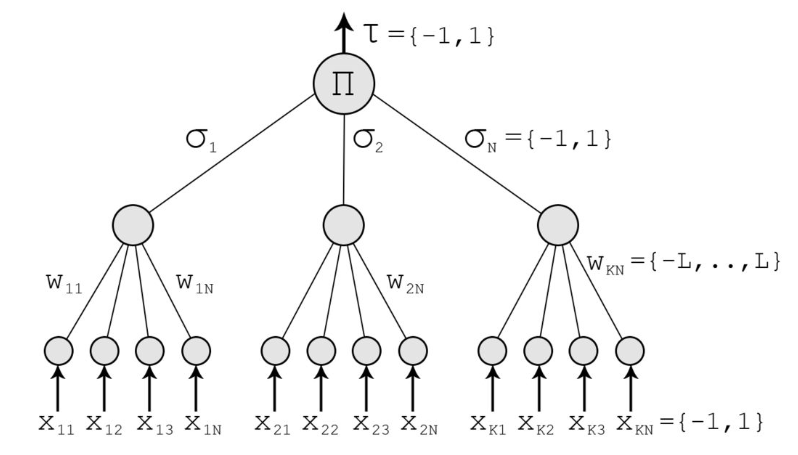


Рис 8. Структура древовидной машины четности

Для достижения равенства весов необходимо выполнить следующие действия:

* Инициализация нейронной сети. В начале процесса синхронизации значения весовых коэффициентов задаются случайным образом.
* Генерация случайного входного вектора, который подается на вход сети.
* Вычисление значения скрытых и выходного нейронов.
* Проверка равенства выходных значений обоих сетей.
* В случае несовпадения выходных значений необходимо вернуться ко второму шагу. Если выходные значения равны, тогда для каждой сети используем одно из трех правил обучений, представленных выше.
* После достижения полной синхронизации сетей на основе полученных весовых коэффициентов генерируется ключ шифрования.

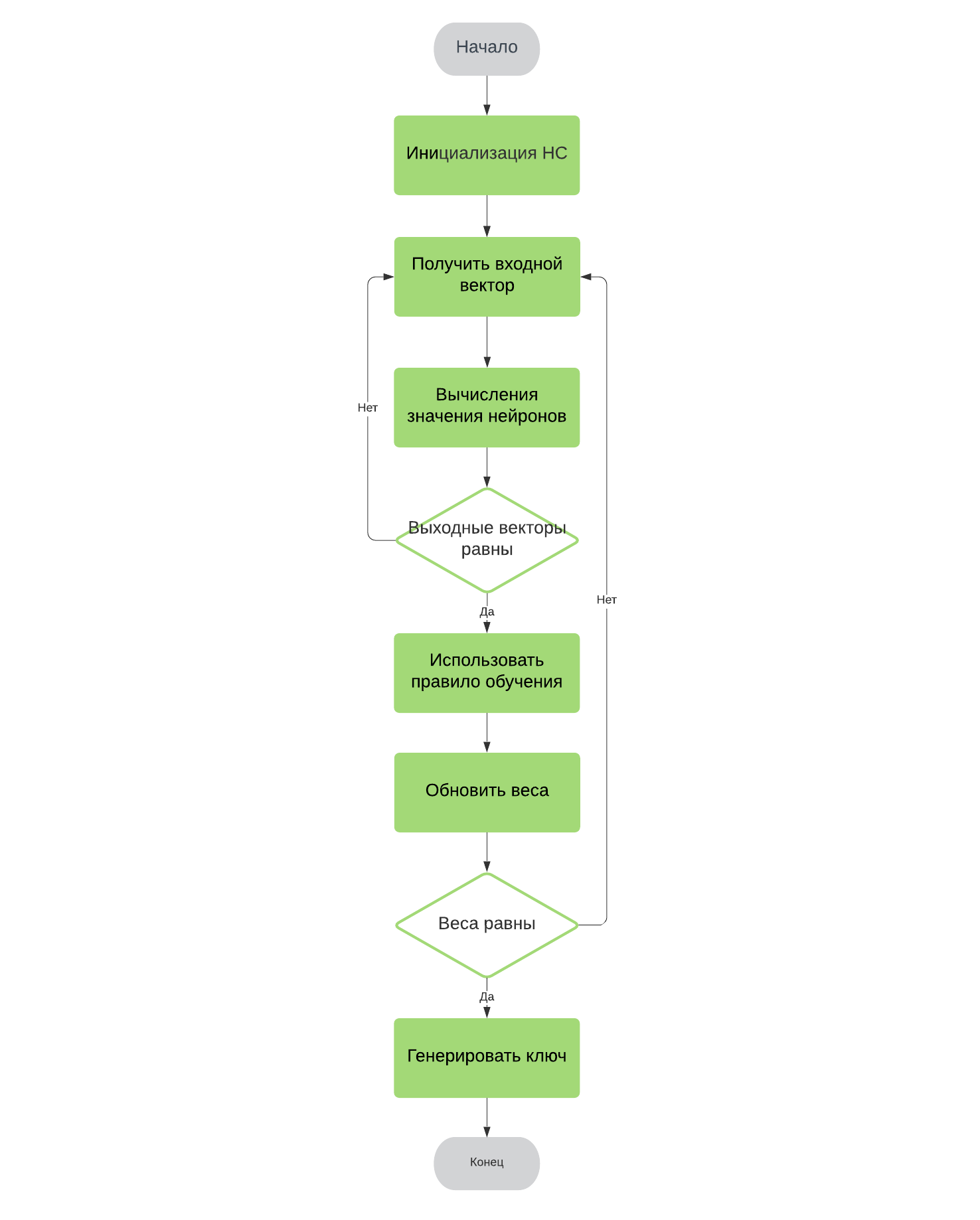


Рис 9. Структура алгоритма синхронизации

Выделяют следующие правила обновления:

* правило Хебба.
* анти-правило Хебба.
* случайное блуждание.

Правило Хебба сформулировано следующим образом:

(21)

Анти-правило Хебба:

(22)

Случайное блуждание:

(23)

Где:

(24)

(25)

Эти правила обучения изменяют только веса, которые связаны со скрытыми единицами с σi = τ. Функция реализована именно для этой цели. Поступая таким образом, невозможно сказать, какие веса обновляются, не зная внутреннего представления (σ1, σ2, ..., σK). Такое свойство является принципиальным для криптографического приложения нейронной синхронизации. Правила обучения должны гарантировать, что веса остаются в допустимом диапазоне от -L до +L. Если какой-либо вес перемещается за пределы этой области, он сбрасывается до ближайшего граничного значения ± L. Это достигается функцией *g(x)* в каждом правиле обучения.

В рамках данной работы также была проанализирована эффективность правил обучения, применяемых к сетям воплощающих различные топологии. В качестве главного параметра было рассмотрено количество обновлений до достижения синхронизации. Чем меньше обновлений необходимо до достижения синхронизации – тем быстрее работает алгоритм. Вследствие чего были выявлены следующие закономерности: алгоритмы, построенные на случайном блуждании, показывают наилучшие результаты при относительно малых значениях количества нейронов, правило Хебба в свою очередь становиться быстрее при увеличении количества нейронов, анти-правило Хебба всегда работало медленнее представленных ранее правил.

По сравнению с другими алгоритмами, основанными на теории чисел, нейроалгоритм имеет ряд преимуществ. Его простота - самая важная из них. Кроме того, это легко реализовать аппаратно. К тому же количество вычислений невелико. Для каждого сообщения может быть сгенерирован новый секретный ключ. Таким образом, никакая секретная информация не должна храниться долгое время. Безопасность алгоритма основана на коротком времени на обучение древовидных машин четности.

Рассмотрим три метода атак, с помощью которых Ева может узнать секретный ключ:

* Метод перебора. Наиболее очевидным методом атаки на машину четности является метод перебора. Однако данный способ является наименее эффективным вследствие того, что необходимо перебрать тысячи комбинаций, прежде чем будет найдена правильная.
* Простая атака. Для осуществления этого типа атаки Ева использует машину четности с такой же архитектурой, как у Алисы и Боба, а также входные xi и выходные τA значения. Как говорилось выше, эти значения передаются по общедоступному каналу связи. Ева так же, как Алиса и Боб, начинает синхронизацию со случайных весов. При этом Ева для обучения использует то же правило обучения, что Алиса и Боб. На каждом шаге возможны три ситуации: τA ≠ τB (веса не обновляются); τA = τB = τE (веса обновляются); τA = τB ≠ τE (веса обновляют Алиса и Боб). Таким образом, сеть Евы обучается медленнее, чем сети Алисы и Боба. Следовательно, Ева может определить ключ лишь с малой вероятностью.
* Генетическая атака. Данный тип атаки основан на эволюционном алгоритме, т.е. создании большой популяции сетей. Перед началом процесса синхронизации Ева имеет одну машину четности с такой же архитектурой, как у Алисы и Боба. Всякий раз, когда Алиса и Боб обновляют веса (т.е. τA = τB), происходит следующее: пока число сетей, которыми обладает Ева, не превышает некоторый за-данный порог M, то все машины четности Евы заменяются на репрезентацию из F новых машин, каждая из которых получается альтернативной заменой скрытого нейрона на противоположное значение, собственно для этого подбираются новые веса.

# Раздел 3. Разработка мессенджера с применением нейрокриптографических алгоритмов

Для реализации поставленных задач был выбран интерпретируемый язык python. Python — объектно-ориентированный язык общего назначения, который разработан с целью повышения продуктивности программиста. Согласно индексу TIOBE (ежемесячный индикатор популярности языков программирования на базе подсчетов результатов поисковых запросов) Python 3 раза определялся языком года: в 2007, 2010 и 2018. Награда присуждается языку программирования, который имеет самый высокий рост рейтинга за год.

Далее приведены несомненные преимущества языка python за счет которых он и был выбран для реализации ВКР:

* Гибкость. Это, по мнению многих разработчиков, основное преимущество языка, так как благодаря своей гибкости язык получил популярность среди многих разработчиков. Как сказал один из моих знакомых, когда начинал изучать этот язык: «Да в Python можно все по винтикам разобрать и собрать в любой момент». И он был прав.
* Расширяемость. Существуют библиотеки и фреймворки под любой тип задач и надобностей. Также огромным плюсом является то, что мы можем использовать C код из Python.
* Простота синтаксиса. Код на Python отличается чистотой и понятеностью без лишних скобок и выражений.
* Интерпретируемость. Интерпретатор Python существует для всех популярных платформ и по умолчанию входит в большинство дистрибутивов Linux, а значит есть на большинстве серверов «из коробки».
* PEP — единый стандарт для написания кода, что делает код поддерживаемым и читабельным даже при переходе от одного программиста к другому. Это поддерживает популярность Python.
* Open Source — код интерпретатора Python является открытым, что позволяет любому, кто заинтересован в развитии языка поучаствовать в его разработке и улучшить его. Если смотреть детали релиза одной из версий языка, то можно заметить, что огромные части нового функционала реализованы сторонними разработчиками.

Также стоит отметить, что данный язык обладает всем необходимым набором инструментов прямо «из коробки». Кроме того, данный язык, благодаря своей ясности и удобочитаемости идеально подходит для решения образовательных задач.

Как и любой другой инструмент Python имеет и недостатки. Ниже описаны некоторые из них.

* Производительность. Большинство разработчиков, и сам создатель языка, сходятся во мнении, что Python не настолько быстрый, насколько хотелось бы. Это обусловлено тем, что Python интерпретируемый язык. Но даже по сравнению с остальными интерпретируемыми языками заметно, что Python проигрывает в производительности. Но это легко можно нивелировать при помощи C реализаций того или иного проблемного участка кода. В условиях сегодняшних мощностей — это несильно заметно.
* Динамическая типизация. Из-за динамической типизации Python потребляет больше ресурсов, чем мог бы, но это зачастую компенсируется внутренним кешированием.
* Global Interpriter Lock. На данный момент это является основной проблемой производительности в Python, а также этим обусловлена плохая реализация многопоточности. Код GIL не менялся с первой версии языка. Это явно указывает на то, что он устарел.

Стоит отметить, что благодаря использованию языка Python код программы очень легко масштабируется под любые задачи.

В целях реализации данной работы помимо сервера и клиента было реализовано некое упрощенное подобие специального протокола, осуществляющего обмен системными сообщениями, необходимыми для инициализации, синхронизации, получение конечного результата, полноценной поддержки всех надлежащих системных или пользовательских сообщений и тд. В нем представлено несколько специфичных заголовков, за каждым из которых стоит определенная реакция сервера или клиента. Ниже приведен список этих заголовков.

* INITHEADER. Необходим для запуска процессов аутентификации пользователя и связи сущности пользователя с сокетом, через который происходит обмен информацией, и соответствующей пользователю нейронной сетью, а также для начала процесса синхронизации.
* SYNCHRONIZ. Необходим для обмена результатами работы нейронных сетей клиента и сервера.
* VECTORHEAD. Используется для обмена входными векторами для нейронных сетей.
* WEIGHTHEAD. С помощью данного заголовка осуществляется обмен значениями весов между нейронными сетями. Так как важны не сами значения, а непосредственно равенство, для большей безопасности можно передавать хэши этих значений.
* COMMONHEAD. Необходим для обмена и соответствующей обработки личными сообщениями пользователей.
* SYCOMPLETE. Нотифицирует систему об окончании синхронизации сетей. Тем самым запускает все необходимые методы окончания синхронизации.

Кроме прочего, в функционал данного протокола так же включена передача длинны сообщения. Таким образом, сокет будет принимать и отправлять ровно столько байт сообщения, сколько нужно. Что уменьшит нагрузку на сеть и немного ускорит систему.

# Серверная часть

На серверной части приложения сконцентрирована вся логика работы мессенджера, а также управление аутентифицированными пользователями и обучение соответствующих нейронных сетей. Обмен сообщениями, как системными, так и пользовательскими основан на технологии websocket.

При подключении потенциального пользователя на стороне сервера инициализируется специальная нейронная сеть необходимая для процесса синхронизации. Для каждого пользователя создается своя отдельная криптографическая нейронная сеть.

Прежде чем принять подключение пользователя, система дожидается синхронизации сетей пользователя и сервера. Стоит отметить, что в рамках данной работы были проанализированы. В системе представлен функционал по изменению правил обучения в зависимости от топологий синхронизируемых сетей. Эта оптимизация позволяет получать наибольшую скорость при любых условиях.

Ниже представлен код, инициализирующий и конфигурирующий серверный сокет на прослушивание и прием новых подключений.

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

server\_socket.bind((IP, PORT))

server\_socket.listen()

В дальнейшем, с помощью системного вызова select, сервер опрашивает все открытые сокеты. В случае если поступает запрос на подключение на серверный сокет, запускаются процессы нейрокриптографической аутентификации и последующей авторизации пользователя в системе. За это отвечает следующая функция, запускаемая в отдельном потоке.

def process\_neural\_crypt(user):

print(f"Started neural synchronization for user: {user.nickname}")

while True:

user.share\_vector()

server\_net\_result = user.crypto.perform()

user.send\_message(server\_net\_result, SYNC\_HEADER)

user\_net\_result = int(user.receive\_message()["data"])

if user\_net\_result \* server\_net\_result > 0:

user.crypto.learn()

user.send\_message("weight request", WEIGHT\_HEADER)

data\_weights = user.receive\_message()["data"]

converted\_data\_weights = convert\_string\_to\_collection(data\_weights)

if user.crypto.weights == converted\_data\_weights:

user.send\_message("1", SYNC\_COMPLETE\_HEADER)

print(f"Synchronization passed successfully for user: {user.nickname}")

break

В случае если поступает сообщение на сокет зарегистрированного пользователя, запускается процесс обработки сообщения и его последующее вещание собеседнику. Процесс обработки нотифицированного сокета представлен ниже.

def main():

while True:

read\_socket, \_, exception\_socket = select.select(clients, [], clients)

for notified\_socket in read\_socket:

if notified\_socket == server\_socket:

client\_socket, client\_address = server\_socket.accept()

user = register\_user(client\_socket)

sync\_thread = threading.Thread(target=process\_neural\_crypt(user))

sync\_thread.start()

print(f"Accepted new connection from {client\_address[0]}:{client\_address[1]} "

f"username:{user.nickname}")

else:

message = clients[notified\_socket].receive\_message()

if message is False:

print(f"Closed connection from {clients[notified\_socket].nickname}")

notified\_socket.close()

del clients[notified\_socket]

continue

notified\_user = clients[notified\_socket]

print(f"Received message from {notified\_user.nickname}: {message['data']}")

cast(message["data"], notified\_user)

for notified\_socket in exception\_socket:

del clients[notified\_socket]

# Клиентская часть

К основному функционалу клиентской части приложения относится отрисовка и поддержка пользовательского интерфейса и формирование реакций на сообщения от сервера. Рассмотрим функцию, отвечающую за взаимодействие с сервером. В ее задачи входит формирование соответствующих ответов на запросы от сервера.

def process\_message(self):

while self.running:

try:

if self.gui\_done:

received\_message = self.user.receive\_message()

if received\_message["main\_header"] == SYNC\_HEADER:

user\_net\_result = self.user.crypto.perform()

self.user.send\_message(user\_net\_result, SYNC\_HEADER)

server\_net\_result = int(received\_message["data"])

if server\_net\_result \* user\_net\_result > 0:

self.user.crypto.learn()

elif received\_message["main\_header"] == VECTOR\_HEADER:

self.user.crypto.inputs = convert\_string\_to\_collection(received\_message["data"])

elif received\_message["main\_header"] == WEIGHT\_HEADER:

weight\_message = convert\_collection\_to\_string(self.user.crypto.weights)

self.user.send\_message(weight\_message, WEIGHT\_HEADER)

elif received\_message["main\_header"] == COMMON\_HEADER and self.synchronized:

self.view\_message(received\_message["username"], received\_message["data"])

elif received\_message["main\_header"] == SYNC\_COMPLETE\_HEADER:

self.view\_message("system", "Authentication passed successfully\n")

self.synchronized = True

except ConnectionAbortedError as err:

print(f"ConnectionAbortedError: {err}")

break

Кроме того, к пользовательскому функционалу относится хранение информации о клиенте, инициализация пользовательской нейронной сети, подключение к серверу, отправка сообщений на сервер и тд.

# Вспомогательные сущности

До этого рассматривалась непосредственно реализация клиентского и серверного приложений, однако также немаловажную роль играют необходимые для корректной работы приложений сущности. В первую очередь стоит рассмотреть класс User. Этот класс содержит в себе всю необходимую информацию о пользователе, сокет для сообщения с сервером, а также реализацию нейронной сети. Данный класс используется как на стороне клиента в единственном экземпляре, олицетворяя пользователя в программном виде, так и на стороне сервера в множественном виде, формируя список, для сбора, сохранения и взаимодействия со всеми подключенными пользователями.

К функционалу этого класса относится отправка и получение сообщений с соответствующей обработкой всех заголовков и поддержкой шифрования, а также отправка и получение входного вектора и хэшей весов нейронных сетей через соответствующий сокет. Как уже было сказано выше хэши при обмене весов используются для дополнительной защиты. Первые два метода необходимы для реализации основного функционала приложения, а именно обмен сообщениями, последние два - активно используются при синхронизации нейронных сетей клиента и сервера.

Так же к функционалу приложения относится класс CryptoNetwork. Данный класс реализует нейронную сеть, используемую в синхронизации между сервером и клиентом. Он используется как на стороне клиента, так и на стороне клиента без каких-либо изменений. К данным этого класса относятся такие поля как количество входных нейронов для каждого скрытого нейрона, количество самих скрытых нейронов, допустимые границы весов, массив весов, используемые для подсчета результата работы нейронной сети, массив входных данных или входной вектор, массив результатов, полученных на каждом скрытом нейроне и числовая переменная для передачи результата работы сети, ее значение по умолчанию равно 0. Все эти данные представляют собой необходимый набор для реализации древовидной машины четности, используемой в данной работе. Количество внешних и внутренних нейронов, а также допустимые значения весов для простоты и повышения читаемости кода определяются константами.

К функционалу данного класса относятся все функции необходимые для подсчета конечного результата нейронной сети и ее обучения. К ним относится функция \_tetta(self, number), она используется для проверки того, что веса не выходят за рамки допустимых значений. В качестве веса выступает передаваемое в эту функцию число, в положительном случае функция возвращает сам передаваемый ей вес, в противном случае – возвращает ближайшую ему числовую границу. Данная функция используется при обучении нейронной сети.

В модуле вместе с классом CryptoNetwork также есть функция \_sign(number). Эта функция не входит в состав класса, но активно в нем используется. Она на вход принимает число и возвращает 1 если число больше нуля, ноль если число равно нулю и -1 если число меньше нуля. Эта функция используется для подсчета значения скрытых нейронов

Функция \_init\_weights(self) необходима для генерации случайных весов при инициализации нейронный сети.

Функция def init\_vector(input\_neurons, hidden\_neurons) возвращает входной вектор необходимый нейронной сети на каждой итерации обучения. Эта функция так же как и \_sign(number) не входит в функционал класса, Так как фактически она возвращает ни к чему не прикрепленный массив данных.

Функция \_multiply\_input\_by\_weights(self) подсчитывает результат умножения входного вектора на весы нейронной сети, тем самым подсчитываются значения скрытых нейронов. Данная функция используется при подсчете выходного значения нейронной сети.

Функция learn(self) отвечает за процесс обучения сети, то есть изменения ее весов по выбранному правилу обучения. В рамках данной работы было выбрано правило Хебба, поскольку оно обеспечивает большую скорость работы при больших значениях количества нейронов, за счет чего достигается большая защищенность системы.

Функция perform(self) возвращает выходное значение нейронной сети при заданной конфигурации. Этим значением может быть либо 1, либо -1.

Так же в приложении присутствуют модули с набором необходимых функций для реализации шифрования и вспомогательный модуль с функциями преобразующие данные. Первый такой пакет называется rsa. В нем присутствуют такие функции как:

* реализация алгоритма решето Эратосфена, для поиска простых чисел.
* обычного алгоритма Евклида, а также его расширенной версии. Данные алгоритмы используются для поиска НОД и коэффициентов соотношения Безу. Эти коэффициенты необходимы для поиска обратного числа по модулю.
* Функция поиска обратного элемента при заданных значениях модуля и экспоненты.
* Функция инициализации экспоненты.
* Функция выборки двух случайных простых числа.

Второй модуль называется Converter. Он содержит необходимые работе приложения константы. А также в нем находятся три вспомогательные функции:

* Функция преобразования двумерной коллекции к строке.
* Функция преобразования строки к двумерной коллекции.
* Функция нахождения количества символов числе. Эта функция использует логарифм для ускорения работы.

# Раздел 4. Безопасность рабочего места инженера-программиста

Сущность охраны труда в современных реалиях представляет собой четкую, систематизированную структуру законодательных мер, социально-экономических, технических, организационных, гигиенических и лечебно-профилактических мероприятий, обеспечивающих безопасность, сохранение здоровья человека и эффективность производственного процесса. Технологическое развитие привнес серьезные преобразования в условия производственной деятельности работников интеллектуального труда. Их работа стала более напряженной и требует значительных затрат энергии. Для этого требовалось комплексное решение проблем эргономики, гигиены и организации труда, регулирования труда и отдыха.

Защита здоровья рабочих, обеспечение безопасности условий труда, устранение профессиональных заболеваний и производственных травм - одна из важнейших забот человеческого общества. Использование прогрессивных форм научной организации труда, сокращение до минимума ручного, неквалифицированного труда, создание среды, исключающей профессиональные заболевания и производственные травмы.

В данном разделе будут изложены вопросы безопасной жизнедеятельности инженера-программиста в процессе создания программного обеспечения.

# Анализ потенциально опасных и вредных факторов

При классификации воздействий различных производственных факторов выделяют две группы:

Вредные производственные факторы. К данному типу факторов относят факторы среды или трудового процесса, которые в определенных условиях, такие как длительность, интенсивность и тд., могут приводить к профессиональной заболеваемости специалиста и, как следствие, к снижению трудоспособности. К этим факторам можно отнести:

* Монотонность трудового процесса
* Электромагнитное излучение
* Электростатическое поле
* Пыль
* Посторонний шум
* Повышение психической нагрузки
* Неравномерные нагрузки в течение трудового процесса
* Перенапряжение глаз

Опасныепроизводственные факторы.К таким факторам относят характеристики производственной среды или помещения, непосредственно приводящие к травме, в том числе и смертельной. В контексте работы инженера-программиста можно выделить удар электрическим током. В общем случае выделяют следующие факторы: механические, электрические, падение с высоты, падение предметов, термические ожоги, химические ожоги, воздействие повышенных или пониженных температур, ДТП, падение, обрушение обвалы предметов и деталей, воздействие вредных веществ, и т. д.

# Анализ воздействия вредных и опасных факторов на инженера-программиста

Согласно ГОСТу 12.0.003-2015 «ССБТ. Опасные и вредные производственные факторы. Классификация», выделяют следующие вредные и опасные воздействия:

1. **Высокий уровень напряжения в электрических сетях.** Специфика работы инженера-программиста заключается в непосредственном длительном взаимодействии с электронными вычислительными машинами. Системный блок и периферийные устройства питаются от сетей общего пользования (220В и 50Гц). Поражение током при неправильном обустройстве рабочего места может вызвать поражения различных органов. В результате поражения током человека наиболее часто встречаются следующие явления: судороги, фибрилляция сердца, прекращение дыхания, паралич сердца и ожоги.
2. **Воздействия вредных факторов от экрана монитора.** Одним из наиболее вредных факторов при работе инженера-программиста является электромагнитное излучение, так как его влияние незаметно для окружающих и тем самым связанно с отягощающими факторами такими как длительность, что приводит к различным заболеваниям. Нормативы излучений должны соответствовать стандартам во всех направлениях: от центра экрана, боковых и задних стенок монитора. Стандарты нормируют излучение в двух поддиапазонах: 20 Гц-2 кГц и 2-400 Гц.
3. **Микроклимат рабочего помещения.** Поддержание микроклимата рабочей зоны персонала в пределах допустимых гигиенических норм является основополагающим фактором для эффективного ведения рабочей деятельности человека. Микроклимат представляет собой комплекс физических факторов среды, имеющий большое влияние на теплообмен человека. Нарушение норм микроклимата рабочего помещения неизбежно приводит изменениям самочувствия, работоспособности, здоровья и производительности человека.
4. **Уровень освещенности в рабочем помещении.** Поддержание здоровья глаз человека напрямую зависит от уровня освещенности рабочего места. Необходимо четко соблюдать нормативы по санитарным нормативам освещенности рабочей зоны, так как, помимо здоровья глаз, уровень освещенности так же влияет на сопротивляемость стрессам, усталости, психологическое здоровье человека, и как следствие здоровье всего организма.
5. **Повышенный уровень шума в рабочем помещении.** Шум является острым биологическим раздражителем и оказывает влияние не только на слух человека, но также воздействует на головной мозг и возбуждает негативные изменения во всем организме. К таким неблагоприятным воздействиям относят: деградацию речи, повышенная утомляемость и снижение производительности труда, различные шумовые заболевания.

В связи с этими угрозами для здоровья сотрудников созданы и записаны различные требования для обеспечения безопасного рабочего места.

# Требования к организации рабочих помещений

Рационально организованное и грамотно приспособленное к трудовой деятельности рабочее место специалиста обеспечивает ему удобное положение в пространстве и помогает поддерживать высокую производительность труда. Требования к организации рабочих помещений изложены в СанПиН 2.2.2 / 2.4.1340-03 «Гигиенические требования к электронным ПК и организации труда». В вышеупомянутом документе изложены принципы и рекомендации к обустройству непосредственно рабочего места, отделке и организации рабочих помещений, размещение различных элементов в нем и т.д. Конструкция рабочего места, включая все элементы взаимодействия, должно соответствовать рациональным, физиологическим и психическим потребностям человека. Далее приведены основные требования:

* Площадь рабочего места не менее 6 м2.
* Необходимо применения как искусственного, так и естественного освещения.
* Необходимо соблюдать оптимальное размещение оборудования, с которым взаимодействует трудящийся.
* Уровень акустического шума не должен превышать заданных значений.
* Рабочее помещение должно соответствовать всем санитарно-эпидемиологическим нормам.
* Все используемые приборы должны быть заземлены в соответствии с требованиями по эксплуатации.

# Микроклимат рабочих помещений

Требования к параметрам микроклимата рабочих помещений приведены в ГОСТ 30494-2011. Данные требования необходимы для предотвращения неблагоприятного воздействия микроклимата рабочих мест, производственных помещений на самочувствие, функциональное состояние, работоспособность и здоровье человека. Они распространяются на показатели микроклимата на рабочих местах всех видов производственных помещений и являются обязательными для всех предприятий и организаций

Таблица 1. Оптимальные величины показателей микроклимата на рабочих местах производственных помещений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Период года | Категория работ, Вт | Температура воздуха, С | Температура поверхностей, С | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | 1a(до 139) 1б(140-174) 2а(175-232) 2б(233-290) 3(от 290) | 22-24  21-23  19-21  17-19  16-18 | 21-25  20-24  18-22  16-20  15-19 | 60-40  60-40  60-40  60-40  60-40 | 0.1  0.1  0.2  0.2  0.3 |
| Теплый | 1a(до 139) 1б(140-174) 2а(175-232) 2б(233-290) 3(от 290) | 23-25  22-24  20-22  19-21  18-20 | 22-26  21-25  19-23  18-22  17-21 | 60-40  60-40  60-40  60-40  60-40 | 0.1  0.1  0.2  0.2  0.3 |

Существуют температурные нормы для рабочих помещений. На основании ГОСТ 12.1.005-88 оптимальные значения относительной влажности, температуры и скорости движения воздуха в производственных помещениях приведены в таблице 2.

Таблица 2. Оптимальные значения температуры, относительной влажности и скорости движения воздуха

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Период года | Категория работ | Температура  воздуха С º,  не более | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | Легкая – 1а | 22-24 | 40-60 | 0,1 |
|  | Легкая – 1б | 21-23 | 40-60 | 0,1 |
| Теплый | Легкая – 1а | 23-25 | 40-60 | 0,1 |
|  | Легкая – 1б | 22-24 | 40-60 | 0,2 |

Кроме того, в таблице 7 приведены допустимые величины интенсивности теплового облучения поверхности тела работников от производственных источников.

Таблица 3. Допустимые величины интенсивности теплового облучения поверхности тела работающих от производственных источников

|  |  |
| --- | --- |
| Облучаемая поверхность тела, % | Интенсивность теплового облучения, 2 Вт/м, не более |
| 50 и более | 35 |
| 25-50 | 70 |
| не более 25 | 100 |

# Требования к уровню шума и вибрации на рабочем месте

Одним из неблагоприятных факторов производственной среды является высокий уровень шума, создаваемый различными комплектующими рабочего места работника, а также персонального компьютера и его периферийных устройств. С физиологической точки зрения, шум – это всякий неприятный, нежелательный для восприятия человека звук. Высокий уровень шума значительно ухудшает условия труда, а также оказывает негативное воздействие на здоровье работника. При длительном воздействии шума происходит нежелательное влияние на здоровье работника:

* повышается кровяное давление;
* снижается внимательность;
* острота слуха и зрения.

Уровень шума на рабочем месте программистов, по СП 2.2.3670-20 не должен превышать 50 дБА, а в залах обработки информации на вычислительных машинах - 65 дБА.

# Требования к степени освещенности

Особое место в работе сотрудника занимает освещенность рабочего места. Освещенность рабочего места прямым образом влияет на нагрузку зрительного аппарата инженера-программиста, следовательно, требуется поддерживать необходимый для безопасной работы уровень освещения, используя комбинации искусственного и естественного источников света. Рациональное освещение рабочего места является одним из важнейших факторов, влияющих на эффективность трудовой деятельности человека, предупреждающих травматизм и профессиональные заболевания.

Таблица 4. Общепринятые величины освещенности

|  |  |  |
| --- | --- | --- |
| Типы работ | Общая освещенность, лк | Скомбинированная  освещенность, лк |
| Работа, требующая высокой  зрительной концентрации | 300 | 750 |
| Работа, требующая средней  зрительной концентрации | 200 | 300 |

Неправильно организованное освещение приводит к таким последствиям как напряжение зрения, ослабление внимания, скорая утомляемость. Чрезмерно яркое освещение вызывает раздражение и резь в глазах, что приводит к общему дискомфорту трудящегося.

# Выводы

В данном разделе были изложены и описаны опасные и вредные факторы, связанные с работой инженера-программиста. Также был произведен анализ их негативного воздействия на организм человека. Были рассмотрены установленные нормы и промежутки допустимых значений воздействия этих факторов.

Стоит принять во внимание, что поддержание рабочего места работника в пределах этих норм является основополагающей целью работодателя, поскольку это влияет не только на работоспособность, производительность и заинтересованность в труде, но и предотвращает появление различных профессиональных заболеваний и получение травм.

# Заключение

В ходе данной работы были успешно решены поставленные задачи. Были рассмотрены базовые механизмы и принципы защиты информации, а также основные виды криптосистем. Были проанализированы основные виды шифров и их математического основания и некоторые виды атак на них.

В ходе практической части данной работы была разработана система обмена сообщениями (мессенджер), являющаяся приложением разработанный с использованием языка Python. Реализация всех вариантов работы приложения в режиме реального времени основана на использовании протокола WebSocket. Все передаваемые в системе сообщения защищены алгоритмами шифрования. В роли алгоритма первичного обмена ключами и предотвращения возможных атак был реализован и использован нейрокриптографический протокол обмена ключами (neural key exchange protocol), использующий синхронизацию нейронных сетей. Система обмена ключами также оснащена дополнительными слоями защиты в виде непредсказуемости размеров топологии сетей, алгоритмов шифрования RSA, а также хеширования.

# Список литературы

1. Акритас А. Основы компьютерной алгебры с приложениями: Пер. с англ. — М., Мир, 1994. — 544 с.
2. Голиков В.Ф. Атака на синхронизируемые искусственные нейронные сети, формирующие общий секрет, методом отложенного перебора / В.Ф. Голиков, А.Ю. Ксеневич // Доклады БГУИР 2017, № 8 (110) С.48 – 53.
3. Фергюсон, Н. Практическая криптография: [пер. с англ.] / Н. Фергюсон, Б. Шнайер – М.: Издательский дом «Вильямс», 2004. – 432 с.
4. Криптографический алгоритм, основанный на синхронизации нейронных сетей [Электронный ресурс] – Режим доступа: https://www.researchgate.net/publication/1832873\_Secure\_exchange\_of\_information\_by\_synchronization\_of\_neural\_networks
5. Криптография. От примитивов к синтезу алгоритмов / М. А. Еремеев, М. А. Молдовян – Санкт-Петербург: «БВХ-Петербург», 2004
6. Лаборатория линуксоида - Решето Эратосфена [Электронный ресурс] – Режим доступа: <https://younglinux.info/algorithm/sieve>
7. Нейрокриптографическая система реккурентных конвергентных нейросетей защиты информации [Электронный ресурс] – Режим доступа: <https://cyberleninka.ru/article/n/neyrokriptograficheskaya-sistema-rekurrentnyh-konvergentnyh-neyrosetey-zaschity-informatsii/viewer>
8. Применение искусственных нейронных сетей и системы остаточных классов в криптографии / Н.И. Червяков [и др.] – М.: ФИЗМАТЛИТ, 2012. – 280 с.
9. Расширенный алгоритм Евклида [Электронный ресурс] – Режим доступа: <https://www.e-olymp.com/ru/blogs/posts/18>
10. Теоретическая криптография / А. Г. Ростовцев, Е. Б. Маховенко – Санкт-Петербург: НПО «Профессионал»
11. Analysis of Neural Cryptography [Электронный ресурс] – Режим доступа: <https://link.springer.com/content/pdf/10.1007%2F3-540-36178-2_18.pdf>
12. Analysis of Neural Cryptography [Электронный ресурс] – Режим доступа: <https://www.researchgate.net/publication/221326789_Analysis_of_Neural_Cryptography>
13. Cryptography Introduction [Электронный ресурс] – Режим доступа: <https://www.geeksforgeeks.org/cryptography-introduction/>
14. Introduction to WebSockets [Электронный ресурс] – Режим доступа: <https://www.linode.com/docs/guides/introduction-to-websockets/>
15. Klimov, A. Analysis of Neural Cryptography / A. Klimov, A. Mityagin, A. Shamir // Advances in Cryptology. ASIACRYPT 2002, LNCS, 2501. С. 288–298
16. Kanter I., Kinzel W., Kanter E. Secure exchange of information by synchronization of neural networks 2002
17. MAXimal [Электронный ресурс] – Режим доступа: <https://e-maxx.ru/algo/>
18. Neural cryptography [Электронный ресурс] – Режим доступа: <https://www.researchgate.net/publication/301907526_Neural_cryptography>
19. Neural Cryptography: From Symmetric Encryption to Adversarial Steganography [Электронный ресурс] – Режим доступа: <https://courses.csail.mit.edu/6.857/2018/project/Modesitt-Henry-Coden-Lathe-NeuralCryptography.pdf>
20. Secure exchange of information by synchronization of neural networks [Электронный ресурс] – Режим доступа: <https://www.researchgate.net/publication/1832873_Secure_exchange_of_information_by_synchronization_of_neural_networks>
21. Symmetric Cryptography [Электронный ресурс] – Режим доступа: <https://www.sciencedirect.com/topics/computer-science/symmetric-cryptography>
22. Searchsecurity – cryptography [Электронный ресурс] – Режим доступа: <https://searchsecurity.techtarget.com/definition/cryptography>