**Содержание**

[Введение 2](#_Toc73563309)

[Раздел 1. Основные принципы и концепции криптографии 3](#_Toc73563310)

[1.1 Симметричные криптосистемы 3](#_Toc73563311)

[1.2 Ассиметричное шифрование 4](#_Toc73563312)

[1.3 Гибридные криптосистемы 5](#_Toc73563313)

[1.4 E2EE 6](#_Toc73563314)

[Раздел 2. Анализ основных алгоритмов шифрования 6](#_Toc73563315)

[2.1 Протокол Диффи-Хеллмана 7](#_Toc73563316)

[2.2 RSA 8](#_Toc73563317)

[2.3 Нейрокриптография. Общие понятия 10](#_Toc73563318)

[2.4 Нейрокриптографическая аутентификация 15](#_Toc73563319)

[Раздел 3. Разработка мессенджера с применением нейрокриптографических алгоритмов 17](#_Toc73563320)

[3.1 Серверная часть 18](#_Toc73563321)

[3.2 Клиентская часть 20](#_Toc73563322)

[3.3 Вспомогательные сущности 22](#_Toc73563323)

[ГЛАВА 4. Безопасность автоматизированного рабочего места инженера-программиста 25](#_Toc73563324)

[4.1 Анализ потенциально опасных и вредных факторов 25](#_Toc73563325)

[4.2 Анализ воздействия вредных и опасных факторов на инженера-программиста 26](#_Toc73563326)

[4.3 Требования к организации рабочих помещений 28](#_Toc73563327)

[4.4 Микроклимат рабочих помещений 29](#_Toc73563328)

[4.5 Требования к уровню шума и вибрации на рабочем месте 30](#_Toc73563329)

[4.6 Требования к степени освещенности 31](#_Toc73563330)

[4.7 Выводы 32](#_Toc73563331)

# 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

В современной криптографии можно рассматривать пять основных принципа: конфиденциальность, аутентичность, доступность, целостность и предотвращение отказа от авторства.

* 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.

# Симметричные криптосистемы

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 вы используете один и тот же ключ для шифрования и дешифрования данных. Проще говоря, отправитель шифрует данные с помощью пароля, а получатель должен знать этот пароль для доступа к данным.

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.

# 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 по обмену данными. Оба ключа математически связаны (оба ключа вместе называются ключевой парой). Также оба ключа необходимы для выполнения операции.

При шифровании данных с помощью закрытого ключа создается так называемая цифровая подпись. Это гарантирует, что сообщение пришло от указанного отправителя (потому что только отправитель имел доступ к закрытому ключу, чтобы иметь возможность создать подпись).

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.

# Гибридные криптосистемы

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.

Главный недостаток симметричных схем шифрования заключается в необходимости обеспечения защищенного канала между лицами или сервисами, осуществляющими обмен информацией. Что в свою очередь исключает смысл шифрования. 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.

# E2EE

Сквозное шифрование (E2EE) — это система связи, в которой только общающиеся пользователи могут читать сообщения. По сути, такая система не позволяет потенциальным перехватчикам, включая поставщиков телекоммуникационных услуг, интернет провайдеров и даже провайдеров услуг связи, получить доступ к криптографическим ключам, необходимым для расшифровки передаваемой информации. Во многих системах обмена сообщениями, включая электронную почту и многие чат-сети, сообщения проходят через посредников и хранятся третьей стороной, из которой они извлекаются получателем. Даже если сообщения зашифрованы, они шифруются только «в пути» и, таким образом, доступны для поставщика услуг, независимо от того, используется ли шифрование диска на стороне сервера. Шифрование диска на стороне сервера просто предотвращает просмотр этой информации неавторизованными пользователями, но оно не мешает самой компании просматривать информацию, поскольку у них есть ключ и они могут просто расшифровать эти данные. Это позволяет третьей стороне предоставлять поиск и другие функции или сканировать незаконный и неприемлемый контент, но также означает, что они могут быть прочитаны и использованы любым лицом, имеющим доступ к сохраненным сообщениям в сторонней системе, независимо от того, преднамеренно ли это сделано. или через черный ход. Это можно рассматривать как проблему во многих случаях, когда конфиденциальность очень важна, например, компании, репутация которых зависит от их способности защищать данные третьих лиц, переговоры и обмен информацией, которые достаточно важны, чтобы иметь риск целевого «взлома» или наблюдения, и где затрагиваются деликатные темы.

# 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 [8]. 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)*

и отправляет это Бобу*.* Боб считает

*B=gb(mod p)*

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)*,

а Бобу

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

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. 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 .

Затем необходимо выбрать некоторое целое число , такое, что:

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

Получаем пару ключей:

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 :

Боб, получив , расшифровывает сообщение:

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

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

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.

# 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. 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.

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.

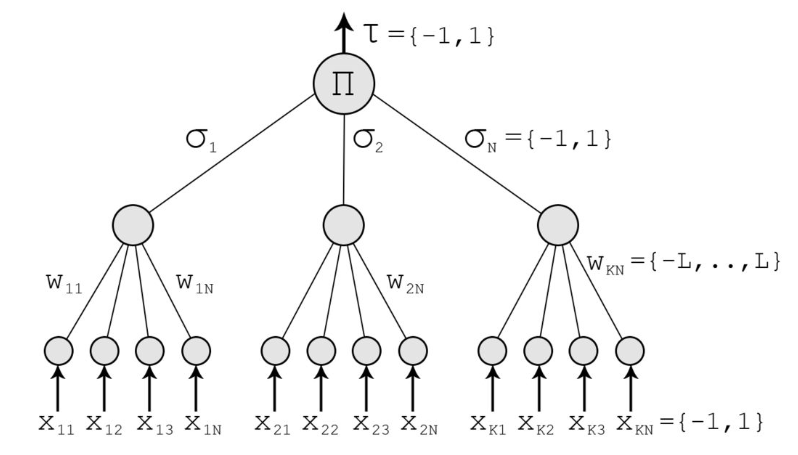
Кроме всего прочего, одной из важных особенностей нейрокриптографии является то, что ее реализация может быть основана на древовидных машинах четности. ДМЧ на самом деле представляет собой нейронную сеть с прямой связью, в которой нейроны входного слоя построены по модели МакКуллоха-Питтса. На втором 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}.

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

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

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

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.



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.

Выделяют следующие правила обновления:

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

Правило Хебба сформулировано следующим образом:

Анти-правило Хебба:

Случайное блуждание:

Где:

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.

# Нейрокриптографическая аутентификация

Аутентификация — это процесс определения того, является ли кто-то или что-то на самом деле тем, кем или чем оно себя объявляет. Технология аутентификации обеспечивает контроль доступа для систем, проверяя, совпадают ли учетные данные пользователя с учетными данными в базе данных авторизованных пользователей или на сервере аутентификации данных.

Пользователи обычно идентифицируются с помощью идентификатора пользователя, и аутентификация выполняется, когда пользователь предоставляет учетные данные, например пароль, которые совпадают с этим идентификатором пользователя. Большинство пользователей лучше всего знакомы с использованием пароля, который как часть информации, которая должна быть известна только пользователю, называется фактором проверки подлинности знаний. Другие факторы аутентификации и способы их использования для двухфакторной или многофакторной аутентификации (MFA) описаны ниже.

Аутентификация важна, потому что она позволяет организациям обеспечивать безопасность своих сетей, разрешая только аутентифицированным пользователям (или процессам) доступ к своим защищенным ресурсам, которые могут включать компьютерные системы, сети, базы данных, веб-сайты и другие сетевые приложения или службы.

После аутентификации пользователь или процесс обычно также подвергаются процессу авторизации, чтобы определить, следует ли аутентифицированному объекту разрешить доступ к защищенному ресурсу или системе. Пользователь может быть аутентифицирован, но не может получить доступ к ресурсу, если этому пользователю не было предоставлено разрешение на доступ к нему.

Термины аутентификация и авторизация часто используются как синонимы; хотя они часто могут быть реализованы вместе, эти две функции различны. В то время как аутентификация — это процесс проверки личности зарегистрированного пользователя перед предоставлением доступа к защищенному ресурсу, авторизация — это процесс проверки того, что аутентифицированному пользователю было предоставлено разрешение на доступ к запрошенным ресурсам. Процесс, посредством которого доступ к этим ресурсам ограничивается определенным числом пользователей, называется контролем доступа. Процесс аутентификации всегда предшествует процессу авторизации.

Посредством синхронизации двух древовидных машин четности можно реализовать систему аутентификации. Поскольку было показано, что третье лицо не может обучать свою нейронную сеть достаточно быстро. То можно утверждать, что в случае успешной синхронизации нейронных сетей, процесс обмена информацией действительно проходит между теми лицами или процессами, за которых они себя выдают. Таким образом предотвращаются многие виды атак, такие как «человек посередине» и другие.

В дальнейшем полученные в результате синхронизации сетей веса можно использовать в качестве ключей для симметричного шифрования, увеличивая тем самым производительность системы.

# Раздел 3. Разработка мессенджера с применением нейрокриптографических алгоритмов

Для реализации поставленных задач был выбран интерпретируемый язык python. Поскольку этот язык обладает всем необходимым набором инструментов прямо «из коробки». Кроме того, данный язык очень ясный и удобочитаемый и идеально подходит для решения образовательных задач. Из минусов стоит отметить, что код на python сложно параллелить. Хотя он и предоставляет некоторые возможности для симуляции запуска нескольких потоков, для реально параллельных приложений стоит выбрать иной язык.

В целях реализации данной работы помимо сервера и клиента был реализован специальный протокол, осуществляющий обмен системными сообщениями, необходимыми для инициализации, синхронизации, получение конечного результата, полноценной поддержки всех надлежащих системных или пользовательских сообщений и тд. В протоколе 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. 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. 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']}")

broadcast(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 автоматизированного 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 3.

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.