

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет информационных
технологий и электроники

Специальность 10.05.03 –

Специализация -

Кафедра
«Информационная безопасность
систем и технологий»

Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНЫЙ КОМПЛЕКС ПО СКРЫТИЮ ИНФОРМАЦИИ ПРИ
ПОМОЩИ ГОМОМОРФНОГО ШИФРОВАНИЯ

Студент



Александрин Олег Михайлович

Руководитель



Кондратьев Р.Ю.

Нормоконтролёр



Егорова Н.А.

Рецензент зам. ген. директора по
спец. вопросам «ПНИЭИ»



Сомкин С.А.

Работа допущена к защите (протокол заседания кафедры от

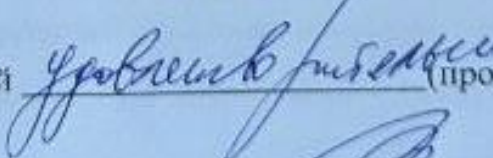
16.06.23 № 10)

Заведующий кафедрой



С.Л. Зефирова

Работа защищена с отметкой



(протокол заседания ГЭК

от 28.09.23 № 14)

Секретарь ГЭК



А.П. Иванов

*Удостоверено
16.06.23
С.Л. Зефирова*

Пенза 2023

Содержание

Введение	4
1 Скрытие информации при помощи гомоморфного шифрования	7
1.1 Система безопасного внешнего хранения данных.....	7
1.1.1 Облачные вычисления	7
1.1.2 Конфиденциальность данных в внешнем хранилище	9
1.1.3 Идея защищенной внешней памяти на сервере.....	12
1.1.4 Идея защищенных вычислений во внешней памяти	13
1.2 Гомоморфное шифрование.....	14
1.2.1 Термин.....	14
1.2.2 Математическая формулировка.....	15
1.2.3 Типы гомоморфных систем относительно набора операций .	16
1.2.4 Область применения	17
1.3 Выводы по разделу.....	18
2 Проектирование	20
2.1 Общая архитектура программного комплекса	20
2.2 Протоколы обмена	21
2.3 Клиентская часть	23
2.4 Серверная часть	24
2.5 База данных.....	25
2.6 Аутентификация между клиентом и сервером.....	28
2.7 Криптографический модуль	29
2.7.1 Описание	29
2.7.2 Гомоморфизм колец.....	29
2.7.3 Построение схемы гомоморфного шифрования	31
3 Реализация.....	33
3.1 Выбор программных средств	33
3.1.1 Выбор языка программирования	33
3.1.2 Используемые библиотеки	33

3.1.3 Реализация базы данных.....	34
3.2 Клиентская библиотека.....	34
3.3 Описание сервера	36
3.4 Инструкция по установке	37
4 Тестирование.....	45
5 Вредные психофизические факторы, влияющие на нервную систему специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	57
Заключение.....	60
Список использованных источников.....	62
Приложение А.Обязательное.Исходный код криптографического модуля	63
Приложение Б.Обязательное.Исходный код сервера	66
Приложение В.Обязательное.Исходный код клиентской библиотеки	70
Приложение Д.Обязательное.Исходный код демонстрационной программы ...	81

Введение

В наше время информационные технологии развиваются настолько стремительно, что одной из наиболее динамичных и перспективных областей в этой сфере становятся облачные вычисления. Этот термин означает предоставление вычислительных ресурсов по запросу пользователя, и здесь под вычислительными ресурсами понимаются не только серверы и системы хранения, но и сети передачи данных, программное обеспечение (ПО) и платформенные сервисы. Поставщики облачных услуг аккуратно распределяют эти ресурсы между потребителями, обеспечивая каждого требуемой мощностью для выполнения собственных задач.

Развитие облачных технологий открывает не только новые возможности, но и повышает эффективность компаний и частных лиц. Одной из ключевых привлекательных черт облачных вычислений является возможность существенного снижения расходов на поддержание собственной ИТ-инфраструктуры. Этот процесс происходит за счет передачи ответственности за хранение и обработку данных провайдером облачных сервисов. Неудивительно, что все больше компаний, независимо от их размеров, прибегают к этому удобному инструменту. Исследования рынка, проведенные исследовательской компанией International Data Group (ныне Foundry) [1], свидетельствуют о том, что в период с 2021 по 2022 год 69% организаций ускорили свой переход в облако. И этот тренд лишь усилится, ожидается, что доля компаний, где практически вся ИТ-инфраструктура будет размещена в облаке, вырастет с нынешних 41% до 63% в ближайшие 18 месяцев.

С ростом объема информации, которую генерируют, собирают и обрабатывают облачные системы, безопасность обработки больших объемов данных становится критически важной задачей для многих государственных и коммерческих организаций. Это особенно актуально, когда речь идет о конфиденциальных данных. Однако гарантировать безопасность и обработку

данных — задача сложная. Ведь с ростом объема данных возрастает и риск их утечки.

Собранные данные часто содержат личную или корпоративную информацию, и утрата контроля над этими данными может иметь серьезные последствия. Преступники и хакеры активно используют подпольные рынки для продажи и покупки украденной информации. Однако современная криптография располагает мощными инструментами, которые способны обеспечить безопасность данных как при их передаче в облако, так и при выполнении вычислений над зашифрованными данными. В данной выпускной квалификационной работе мы подробно рассмотрим один из таких инновационных методов — "гомоморфное шифрование", которое позволяет выполнять вычисления над данными, не раскрывая их содержания. Гомоморфное шифрование представляет собой перспективную криптографическую технологию, позволяющую проводить операции и вычисления над зашифрованными данными, не раскрывая ключей шифрования и не дешифруя информацию. Это означает, что даже проводя математические операции над зашифрованными данными, мы можем получать результат в зашифрованной форме, обеспечивая полную конфиденциальность информации на всех этапах ее обработки.

Для достижения поставленной цели необходимо решить следующие задачи:

- рассмотреть существующие системы с использованием внешнего хранения данных;
- изучить средства защиты, применяемые в таких системах – их плюсы и минусы;
- изучить возможности и способы реализации гомоморфного шифрования;
- спроектировать архитектуру программного комплекса;
- спроектировать серверную и клиентскую части комплекса;

– выбрать программные средства для реализации программного комплекса;

– разработать серверную часть;

– разработать клиентскую часть;

– разработать криптографический модуль;

– протестировать работу программного комплекса.

1 Скрытие информации при помощи гомоморфного шифрования

1.1 Система безопасного внешнего хранения данных

1.1.1 Облачные вычисления

Облачные вычисления (англ. cloud computing) — это модель предоставления повсеместного, удобного сетевого доступа «по требованию» к разделяемому пулу (англ. pool – пространство) конфигурируемых вычислительных ресурсов (например, сети, серверы, память, приложения и сервисы), которые могут быть предоставлены и освобождены в короткие сроки с минимальными усилиями в управлении или с минимальным взаимодействием с поставщиком услуги.

Определение выше было представлено в сентябре 2011 г. Национальным Институтом стандартов и технологий (NIST) США, однако идея и технология возникли гораздо раньше из-за потребности доступа к вычислительным ресурсам. Так, еще в 60-ые годы прошлого века возникла концепция разделения доступа к мэйнфреймам (англ. mainframe – вычислительные машины) по времени, так как подобный ресурс не был доступен для сотрудников сторонних компаний [2]. После, с появлением на рынке дешевых персональных компьютеров, данная проблема была решена для большинства компаний и обычных пользователей. Но к концу 90-ых годов возникает задача вычисления больших данных, для которых ресурсов обычных персональных компьютеров стало не хватать, а приобретать и содержать инфраструктуру многим компаниям (и тем более обычным пользователям) было не выгодно. Таким образом, вновь возникшая потребность в обработке данных вскоре привела к развитию целой отрасли, в которой компании предоставляют удаленный доступ к своим вычислительным ресурсам.

При этом обеспечивается ряд следующих преимуществ над классическими сценариями использования собственной инфраструктуры:

- эффективность. Удаленные ресурсы независимы от вычислительных систем и их географического расположения. Это обеспечивает существенную

экономии благодаря легкому выделению ресурсов по потребностям и одновременно более полной их загрузки;

– гибкость. Все ресурсы, программное и аппаратное обеспечение могут быть переконфигурированы под новые информационные системы практически мгновенно. В то же время, объем технологических ресурсов можно легко масштабировать в моменты пиковых нагрузок, а затем вернуться к предыдущему уровню;

– надежность. В «облаке» реализован необходимый уровень резервирования, при этом на создание бэкапа (англ. backup – резервная копия) и восстановление по запросу отдельно выделяются необходимые ресурсы. Задача создания локальных резервных конфигураций отпадает;

– автоматизация. ПО для управления ресурсами «облака» автоматически выполняет свои функции, динамически направляя запрашиваемый объем ресурсов пользователю для их использования. Это приводит к уменьшению количества ежедневных действий корпоративного IT-персонала и более точным запросам на предоставление ресурсов;

– оптимизация. «Облако» управляется как единая система, поэтому конечный пользователь получает возможность существенной оптимизации используемых ресурсов за счет наилучшего сочетания их возможностей, производительности и стоимости;

– легкость доступа. Доступ автоматизирован и, как правило, осуществляется через веб-браузер.

Таким образом, использование облачных вычислений приводит к радикальному изменению способа использования технологических ресурсов. При этом потребление ресурсов, предоставляемых провайдером облачных ИТ-вычислений, легко поддается измерению, легко «дозируется» и конфигурируется. Это дает возможность отслеживать реальное потребление ресурсов, а значит лучше понимать необходимость расходов.

1.1.2 Конфиденциальность данных в внешнем хранилище

Обеспечение безопасности данных в облачных вычислениях представляет собой вопрос критической важности. Эта задача заключается в том, чтобы гарантировать конфиденциальность информации как на этапе ее передачи от клиента к серверу, так и в процессе ее обработки и хранения на сервере. Для эффективного решения этой задачи провайдеры облачных вычислений используют методы криптографии, включая шифрование данных с использованием специального ключа, который хранится на сервере.

Одним из важнейших преимуществ такого подхода является непроницаемость для прослушивания информации. Когда данные передаются от клиента к серверу и обратно, они остаются зашифрованными и недоступными для посторонних. Это означает, что, даже если злоумышленники попытаются перехватить передаваемую информацию, они не смогут прочитать ее без знания правильного ключа. Таким образом, конфиденциальность данных поддерживается на всех этапах их передачи и обработки.

Кроме того, использование секретного ключа, хранящегося на сервере, обеспечивает дополнительный уровень безопасности. Даже если клиентская сторона будет скомпрометирована, данные останутся недоступными для несанкционированного доступа. Это делает систему более надежной и защищенной.

Важно подчеркнуть, что шифрование данных с использованием ключа на сервере не только обеспечивает конфиденциальность, но также устраняет возможность прослушивания и несанкционированного доступа к информации. Это создает дополнительные гарантии для клиентов и делает облачные вычисления более безопасными и надежными. Общая структурная схема такого подхода к обеспечению безопасности представлена на рисунке 1.



Рисунок 1

Данный подход позволяет обеспечить защиту от несанкционированного доступа к данным от злоумышленника, находящегося за рамками системы. Модель атаки «извне» представлена на рисунке 2.



Рисунок 2

Несмотря на все преимущества шифрования данных с использованием ключа, хранящегося на сервере, следует также учитывать его ограничения. Одним из главных минусов данного подхода является то, что в случае, если злоумышленником является сотрудник самого сервиса или, скажем, администратор системы, который имеет доступ к ключу и инфраструктуре,

обрабатывающей данные, тогда этот подход не сможет полностью обеспечить защиту данных.

В такой ситуации, сотрудник или администратор может иметь возможность расшифровать зашифрованные данные, так как у него есть доступ к ключу, который используется для шифрования и расшифровки информации. Это создает потенциальный риск для конфиденциальности данных, поскольку доверие к сотрудникам или администраторам не всегда может быть абсолютным. Данная ситуация смоделирована на рисунке 3.

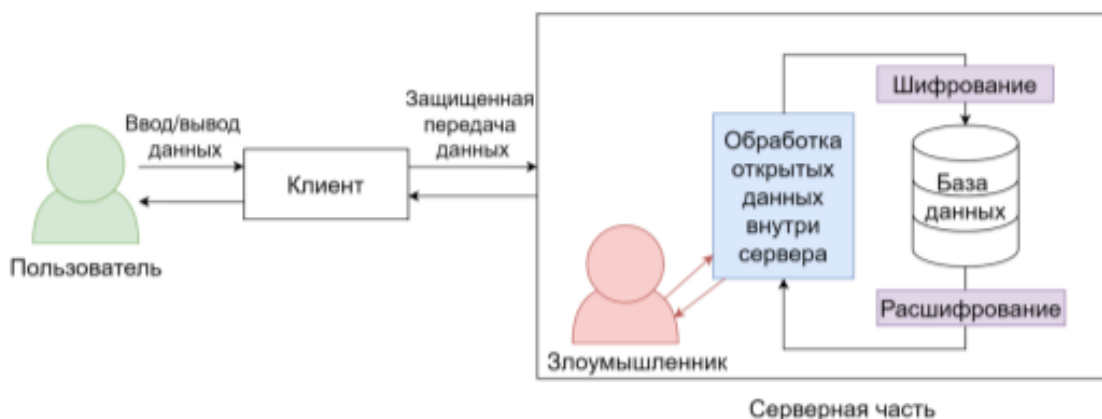


Рисунок 3

В связи с этим получила распространение концепция шифрования на клиентской стороне. Эта идея подразумевает, что перед отправкой данных на удаленный сервер, владелец зашифрует их собственным секретным ключом, защищенное хранение которого сделает невозможным получение доступа к файлам внутри облака.

Идея шифрования на клиентской стороне представляется решением проблемы безопасности хранимых пользователями данных «в облаке». Однако существует следующая проблема, возникающая при использовании данной концепции, – работа с зашифрованными данными. А именно вычисление над ними. Наиболее простым, с точки зрения реализации, и очевидным решением подобной проблемы является последовательное выполнение следующих операций:

- 1) создание локальной копии удаленной области памяти;
- 2) расшифрование данных;
- 3) осуществление необходимых операций;
- 4) получение результатов;
- 5) отправка зашифрованного результата на сервер (если необходимо).

Описанная выше процедура умалает возможности облачного сервиса обработки данных, обеспечивая при этом их надлежащую (в рамках поставленной задачи) защиту. Необходим подход, позволяющий как производить шифрование на стороне клиента, так и выполнять определенные действия над данными, хранящимися удаленно. Но для начала нужно более детально рассмотреть саму идею защищенного хранения данных вне локального компьютера.

1.1.3 Идея защищенной внешней памяти на сервере

Данные, над которыми необходимо будет производить действия удаленно, должны быть не только зашифрованы клиентом, но и неким образом организованы на сервере. Самым простым и привычным способом обеспечить такую организацию является использование именованных переменных на серверной части системы. Тогда логика работы и вычислений сводится к использованию области памяти, подобной той, что есть у пользователя на его локальном компьютере. Так, пользователь сможет:

- 1) инициализировать переменные;
- 2) менять значения переменных;
- 3) удалять переменные.

Каждому отдельному пользователю принадлежит своя удаленная и изолированная от других область памяти, содержащая все выделенные им переменные и их значениями (рисунок 4)

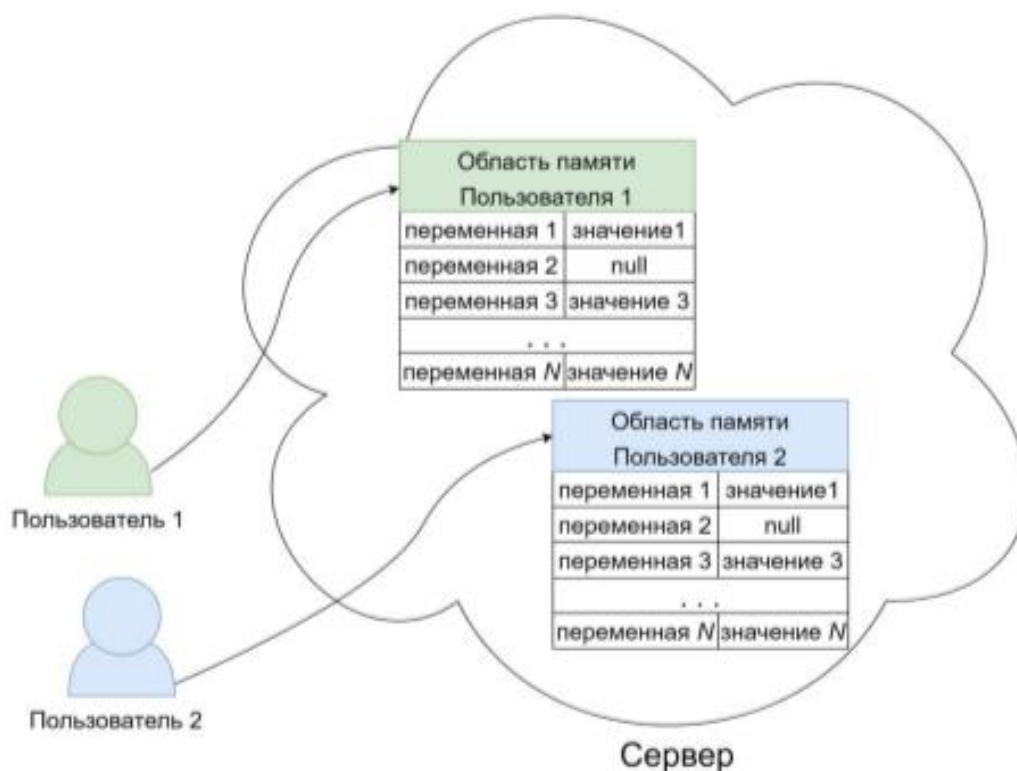


Рисунок 4

1.1.4 Идея защищенных вычислений во внешней памяти

Однако просто хранить переменные недостаточно, необходимо иметь возможность производить над ними определенные вычисления во внешней области памяти. Например, выполнять над переменными базовые математические операции:

- сложение;
- вычитание;
- умножение;
- деление.

Также должна быть возможность записать получившийся результат вычислений в переменную в защищённой внешней памяти.

Теперь, сформировав представление об устройстве внешней области памяти на сервере, можно перейти к рассмотрению криптографического инструмента, который позволяет производить математические операции с

данными в зашифрованном виде и, следовательно, даёт возможность обеспечить шифрование на клиентской стороне секретным ключом пользователя.

1.2 Гомоморфное шифрование

1.2.1 Термин

Гомоморфное шифрование представляет собой важную форму криптографии, которая обеспечивает уникальные возможности для безопасной обработки зашифрованных данных. Этот метод позволяет выполнять определенные математические операции над зашифрованным текстом, не раскрывая сам текст, и получать зашифрованный результат, который соответствует результату вычислений, выполненных над открытым текстом. Другими словами, гомоморфное шифрование позволяет проводить вычисления на зашифрованных данных, сохраняя их конфиденциальность.

Одним из примеров использования гомоморфного шифрования является возможность выполнять арифметические операции над зашифрованными числами. Например, один человек может сложить два зашифрованных числа, не зная их реальных значений. Затем другой человек, обладая соответствующим ключом, может расшифровать получившуюся зашифрованную сумму и получить точный результат сложения, не раскрывая при этом исходные числа.

История гомоморфного шифрования уходит в 1978 год, когда в работе "О банках данных и гомоморфизмах, сохраняющих конфиденциальность" Рональд Ривест, Леонард Адлеман и Майкл Дертусос, создатели системы RSA, были впервые введен этот термин. Они предположили, что возможно выполнение произвольных операций над зашифрованными данными без необходимости расшифровывать их.

Одним из ключевых преимуществ гомоморфного шифрования является его способность обрабатывать данные, не раскрывая их содержание. Это делает его идеальным инструментом для облачных вычислений и обработки конфиденциальных данных, так как он позволяет выполнять вычисления на зашифрованных данных без необходимости расшифровывать их. Это значит, что

даже в случае, если сервер или облачный сервис являются недоверенными, данные останутся надежно защищенными.

Тем не менее, стоит также отметить некоторые ограничения гомоморфного шифрования. Например, это вычислительно сложный процесс, который требует значительных ресурсов. Кроме того, не все виды операций можно выполнять с такой же эффективностью, как над открытыми данными. Но несмотря на эти ограничения, гомоморфное шифрование остается мощным инструментом для обеспечения конфиденциальности данных в современных информационных системах.

1.2.2 Математическая формулировка

Гомоморфизм алгебраических систем в общем виде можно описать следующим образом. Пусть даны две однотипные алгебраические системы:

$$U = \{S; f_1, \dots, f_k; p_1, \dots, p_l; s_1, \dots, s_m\},$$

$$C = \{S'; f'_1, \dots, f'_k; p'_1, \dots, p'_l; s'_1, \dots, s'_m\},$$

где S и S' – множества; $f_1, \dots, f_k, f'_1, \dots, f'_k$ – функции, $p_1, \dots, p_l, p'_1, \dots, p'_l$ – предикаты, определенные на множествах S и S' соответственно, $\{s_1, \dots, s_m\} \subset S$, $\{s'_1, \dots, s'_m\} \subset S'$ – известные константы.

Отображение $\varphi : S' \rightarrow S$ называется гомоморфизмом алгебраической системы U в C , если выполняются следующие условия:

1. $\forall i \in \{1, \dots, k\}: \varphi(f_i(a_1, \dots, a_n)) = f'_i(\varphi(a_1), \dots, \varphi(a_n))$;
2. $\forall i \in \{1, \dots, l\}: p_i(a_1, \dots, a_n) = p'_i(\varphi(a_1), \dots, \varphi(a_n))$;
3. $\forall i \in \{1, \dots, m\}: \varphi(s_i) = s'_i$.

Согласно гомоморфизму конфиденциальности [3] на отображение φ накладываются следующие условия:

1. $\varphi : S' \rightarrow S$ является шифрующей функцией;
2. существует обратная функция $\varphi^{-1} : S' \rightarrow S$, являющаяся дешифрующей;
3. функции φ и φ^{-1} легко вычислимы;
4. операции f'_i и предикаты p'_i также эффективно вычислимы;

5. представление зашифрованного значения $\varphi(di)$, где $di \in S$, не должно занимать намного больше места, чем представление незашифрованного значения di ;

6. знания $\varphi(di)$ для большого количества di , недостаточно для восстановления φ (атака по известному шифротексту);

7. знания нескольких пар $(di, \varphi(di))$ недостаточно для восстановления φ (атака по подобранному открытому тексту);

8. знание алгоритмов вычисления операций $f' i$ и предикатов $p' i$ недостаточно для построения алгоритма вычисления φ .

Будем говорить, что для алгебраических систем U и S , описанных выше, можно построить гомоморфное шифрование, если:

1) существует достаточно большое количество функций φ , удовлетворяющих определению гомоморфизм конфиденциальности, причем такие функции достаточно легко строить;

2) число операций $k > 0$.

Проще говоря, гомоморфное шифрование является отображением одной алгебраической системы в другую так, чтобы при выполнении определенного набора операций с элементами в первой, сохранялось единообразное отображение во вторую.

1.2.3 Типы гомоморфных систем относительно набора операций

Полностью гомоморфное шифрование – это гомоморфное шифрование, где алгебраические системы S и U (введенные в разделе 1.2.2) являются кольцами, причем среди операций $f i$ существуют операции сложения (+) и умножения ($\times$).

Если же среди операций $f i$ присутствует только операция сложения или умножения, в таком случае шифрование будет являться частично гомоморфным.

Потенциальную возможность создания полностью гомоморфной системы шифрования впервые доказал аспирант Стэнфордского университета Крейг Гентри в 2009 году своей работе «Полное гомоморфное шифрование, основанное

на криптографии идеальных решеток». Другие криптосистемы, существовавшие до этого (среди коих криптосистема RSA, система Эль-Гамала и др.), являлись лишь частично гомоморфными, т.е., грубо говоря, позволяли выполнять лишь одну из двух операций с сохранением свойства гомоморфизма.

1.2.4 Область применения

Гомоморфное шифрование нашло применение в там, где есть третья недоверенная сторона, которой субъект вынужден отправить свои данные. Применение вышеупомянутого инструмента позволило бы предоставлять различные услуги, не предоставляя открытые пользовательские данные для каждой услуги (общая схема взаимодействия приведена на рисунке 5)



Рисунок 5

Областей, где можно применить данный инструмент, достаточно много (и с развитием этого направления их будет становиться всё больше), однако к наиболее значимым можно отнести:

- облачные вычисления. Как было сказано выше, применение гомоморфного шифрования позволит обеспечить целостность, доступность и конфиденциальность информации в недоверенной среде («облаке»), сохранив при этом возможность ее обработки;

- система электронного голосования. Система будет способна зашифровать голоса избирателей и провести расчёты над зашифрованными данными (подсчет голосов), сохраняя анонимность каждого избирателя;

– защищенный поиск информации. Гомоморфное шифрование делает возможным извлечение необходимой информации (например, из поисковых систем) с сохранением конфиденциальности запроса пользователя: сервисы смогут получать и обрабатывать запросы и выдавать результаты обработки, не анализируя и не фиксируя их содержимое;

– защита беспроводных децентрализованных сетей связи. Одной из основных проблем при построении децентрализованных сетей связи является обеспечение безопасности передаваемых данных. Для решения этой проблемы может применяться гомоморфное шифрование, встраиваемое в протоколы маршрутизации. В этом случае операции над шифрованными текстами смогут безопасно выполняться промежуточными узлами;

– медицинские исследования. В медицине конфиденциальность пациентских данных имеет высший приоритет, и гомоморфное шифрование позволяет исследователям обрабатывать зашифрованные медицинские данные, не раскрывая личную информацию пациентов;

– финансовые операции. Проведение безопасных финансовых транзакций и обработка финансовых данных требуют высокой степени защиты.

Гомоморфное шифрование позволяет банкам и финансовым учреждениям выполнять вычисления и анализ данных, не раскрывая чувствительную информацию клиентов.

В дополнение к вышеперечисленным областям применения, гомоморфное шифрование также находит применение в интернете вещей (IoT), государственных системах связи и защите личной информации. Развитие этой технологии продолжает расширять ее горизонты и делать ее незаменимой в сфере информационной безопасности и обработки конфиденциальных данных.

1.3 Выводы по разделу

Из всего вышеописанного можно сделать вывод, что в современном мире как у людей, так и у компаний есть большая необходимость в использовании ресурсов и мощностей, предоставляемых третьими лицами, для безопасного

выполнения собственных расчетов и хранения данных. В то же время современная криптография способна удовлетворить потребность в безопасности как хранимых, так и обрабатываемых данных. В частности - гомоморфное шифрование представляет собой инновационный подход к обработке данных, который нашел широкое и разнообразное применение в различных сферах. Оно обеспечивает высокий уровень конфиденциальности, защищая данные от несанкционированного доступа и анализа. Вместе с тем, по мере развития этой технологии, можно ожидать появления новых возможностей и расширения областей ее применения.

В данном разделе была рассмотрена идея облачных вычислений и рассмотрены причины значимости этого направления в современном мире. Была введена модель атакующего и сформулирована задача обеспечения конфиденциальности хранимой и обрабатываемой на сервере информации. В общем виде был рассмотрен математический инструмент, предоставляющий решение в рамках поставленной задачи, а также рассмотрена область его применения.

2 Проектирование

2.1 Общая архитектура программного комплекса

Первым шагом в разработке любой системы является разработка архитектуры. Сервис должен представлять собой цепочку взаимосвязанных логических модулей, каждый из которых выполняет собственную уникальную задачу. На рисунке 6 представлена структурная графическая схема всей системы.

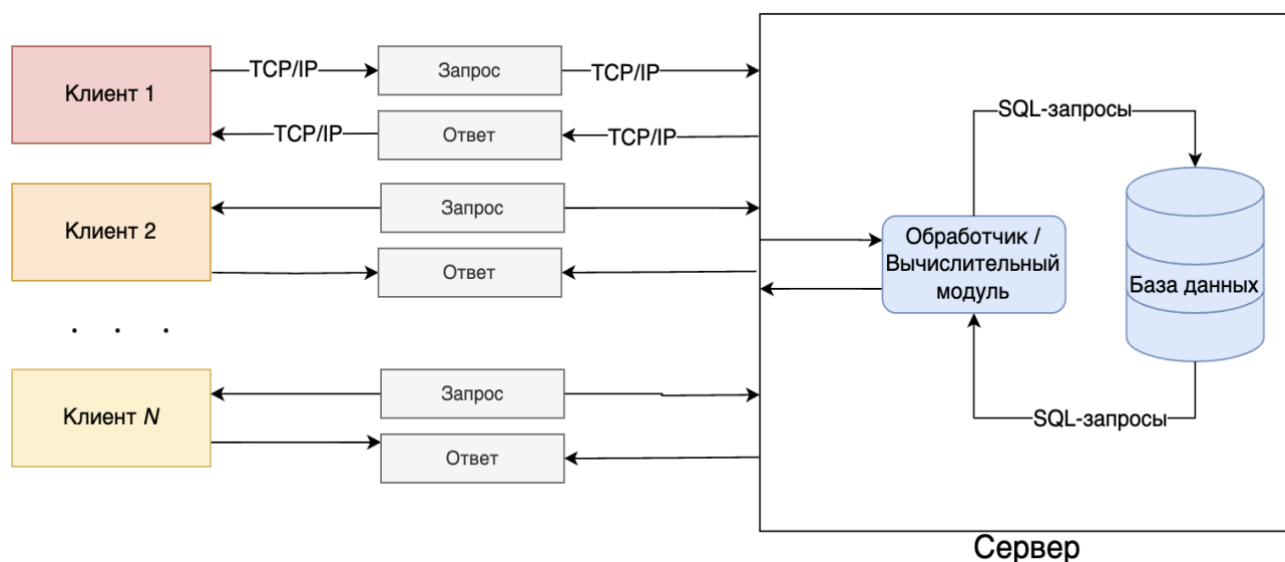


Рисунок 6

На рисунке 6 представлены основные компоненты системы и протоколы обмена данными между модулями программного комплекса. Среди них:

- 1) клиент. Клиентская библиотека, обеспечивающая функционал для пользователя системы. С помощью нее пользователь может выполнять основные функции над удаленной областью памяти и получать результат выполнения действий;
- 2) серверная часть. Модуль, отвечающий непосредственно за выполнение приходящих запросов пользователя по работе с данными и отправке данных на клиент. Связан с базой данных, к которой выполняет запросы для последующего выполнения вычислений;
- 3) база данных. Агрегатор пользовательских данных, а именно, учетной записи пользователя и принадлежащей ему изолированной области памяти.

Однако в данном программном комплексе так же представлены вспомогательные модули, которые используются в модулях, описанных выше:

- модуль аутентификации. Обеспечивает регистрацию и авторизацию пользователя в системе. Так же, как и криптографический модуль, является частью клиента;

- криптографический модуль. Отвечает непосредственно за шифрование пользовательских данных. Является частью клиентского модуля, что очевидно в рамках данной работы. Также ответственен за процедуру выполнения смены секретных ключей (которая сопряжена с временным выкачиванием дампа памяти, хранящейся на сервере для выполнения процедуры перешифрования).

В связи с тем, что архитектура программы подразумевает наличие таких компонентов, как клиент и сервер, необходимо обеспечить надежную (а в рамках данной работы и защищенную) передачу данных между вышеупомянутыми модулями, используя подходящие протоколы. В то же время, исходя из структурной схемы, приведенной на рисунке 6, необходимо выполнять запросы к базе данных для извлечения хранимых в ней переменных и данных пользователя, а также записи новых данных. Для этой цели тоже необходимо применить надежный протокол обмена данными с БД. О протоколах и о каждом реализуемом модуле будет подробно изложено ниже.

2.2 Протоколы обмена

На рисунке 7 представлены все программные узлы реализуемого комплекса, для связи которых необходимо прибегнуть к использованию протоколов для передачи данных.

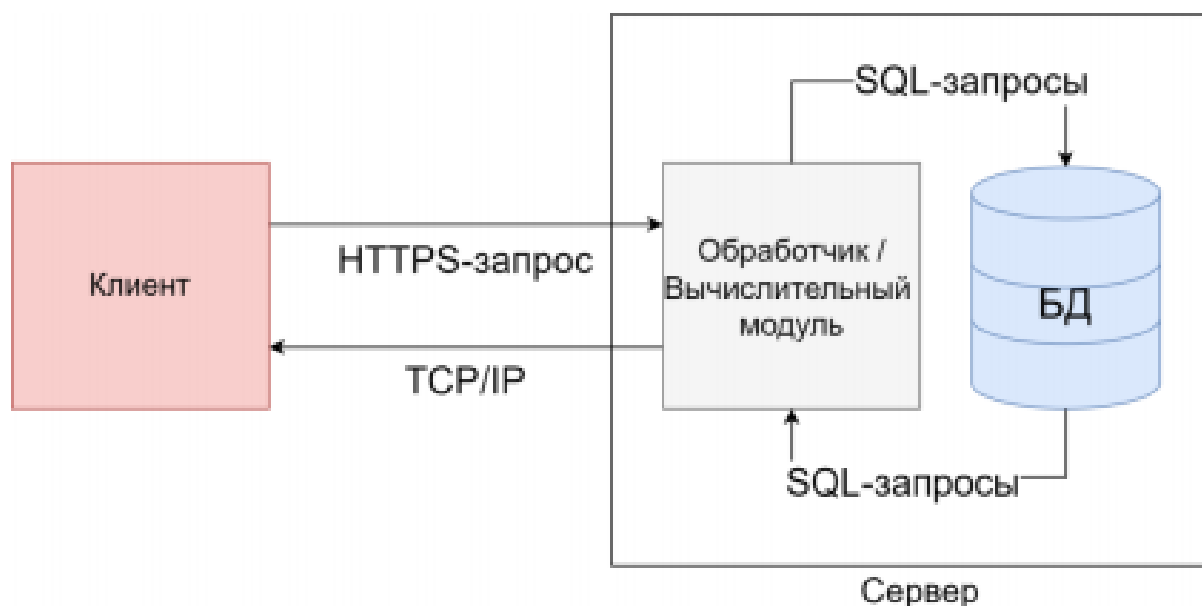


Рисунок 7

Так, для реализации надежной доставки текстовой информации от клиента к серверу наиболее предпочтительным является протокол транспортного уровня TCP [4]. Он позволяет достоверно передавать потоки байт, а в случае пропажи пакета, повторно отсылать его получателю до тех пор, пока он не будет принят. Структура заголовка сегмента TCP протокола представлена на рисунке 8.

Бит	0 – 3	4 – 6	7 – 15	16 – 31
0	Порт источника, Source Port			Порт назначения, Destination Port
32	Порядковый номер, Sequence Number (SN)			
64	Номер подтверждения, Acknowledgment Number (ACK SN)			
96	Длина заголовка, (Data offset)	Зарезервировано	Флаги	Размер Окна, Window size
128	Контрольная сумма, Checksum			Указатель важности, Urgent Point
160	Опции (необязательное, но используется практически всегда)			
160/192+	Данные			

Рисунок 8

Так как передаваемыми данными являются строки, содержащие какие-либо ключевые слова, удобнее всего для их передачи использовать прикладной протокол HTTP. Однако, передача данных по такому протоколу происходит в открытом виде, и, хотя передаваемые данные содержат зашифрованные криптографическим модулем значения переменных, раскрывать прочую информацию все равно нежелательно. С этой целью был выбран протокол, являющийся расширением вышеупомянутого, а именно протокол HTTPS.

Данные в протоколе HTTPS передаются поверх криптографических протоколов TLS или SSL, что гарантирует сохранность конфиденциальности выполняемого клиентом запроса.

Однако помимо взаимодействия клиента и сервера в программном комплексе также необходимо обмениваться запросами с базой данных. В связи с выбором реляционной базы данных, обращение с ней осуществляется путем выполнения SQL-запросов.

2.3 Клиентская часть

Клиентская часть программы (рисунок 9), представляющая из себя клиентскую библиотеку (использует демонстрационный модуль), модуль аутентификации и криптографический модуль позволяет пользователю осуществлять следующие действия:

1) выполнять регистрацию в системе. Так, пользователь может создать учетную запись в системе и иметь возможность присоединиться к использованию сервиса. По окончании регистрации пользователю нужно создать секретный ключ;

2) выполнять аутентификацию для входа в зарегистрированную ранее учетную запись, после чего может полноценно начать пользоваться функционалом программы;

3) через главное меню демонстрационной программы осуществлять действия с защищенной удаленной областью памяти, а именно: создавать переменные, изменять их, присваивать в них значения и удалять;

4) через главное меню демонстрационной программы выполнять вычисления используя имеющиеся переменные: складывать, вычитать, умножать и делить значения;

5) через главное меню пользователь при желании может сменить свой секретный ключ на новый: клиент сгенерирует новый экземпляр, выполнив при этом процедуру перешифрования на новом ключе всей имеющейся в

распоряжении пользователя удаленной памяти. Процедура перешифрования будет осуществлена клиентом.



Рисунок 9

После выбора пользователем действия формируется запрос на сервер. Если этот запрос содержит переменную, посылаемую в удаленную память, тогда ее значение шифруется криптографическим модулем при помощи секретного ключа пользователя. Так или иначе, содержание любого запроса, передающегося по протоколу HTTPS, шифруется при помощи сертификата сервера перед дальнейшей передачей по сети. Ответ, полученный от сервера, проходит обратную процедуру, и клиентская библиотека выдает его пользователю в соответствующем обработанном виде. Если с сервера запрашивалась переменная, то к ней дополнительно применяется процедура дешифрования, инициированная криптографическим модулем.

2.4 Серверная часть

Серверная часть программного комплекса, схема которой представлена на рисунке 10, представляет из себя 3 компонента:

- 1) обработчик пользовательских запросов (программный интерфейс серверного приложения);
- 2) модуль для выполнения операций с удаленной памятью;
- 3) база данных с зашифрованной информацией.



Рисунок 10

Все запросы, приходящие от пользователей (включая регистрацию и аутентификацию) по протоколу HTTPS попадают в программный интерфейс приложения, после чего дешифруются соответствующим образом, и в случае успешного запроса происходит вызов модуля для работы с удаленной памятью. После выполнения работы, ответ передается в программный интерфейс и возвращается пользователю.

2.5 База данных

Хоть база данных и является частью сервера, её имеет смысл рассмотреть отдельно и более подробно.

Разрабатываемая база данных должна хранить следующую информацию:

- учетные записи пользователя;
- закрепленную за каждым пользователем удаленную область памяти.

Как было написано ранее, в данной работе используется база данных, основанная на реляционной модели: такая база имеет набор данных с predetermined связями между ними. В описанном выше подходе, где за учетной записью пользователя закреплена своя область памяти (набор переменных), наиболее предпочтительна именно реляционная модель базы данных. Логическая модель разрабатываемой базы данных с описанной связью двух сущностей (таблиц) приведена ниже, на рисунке 11.

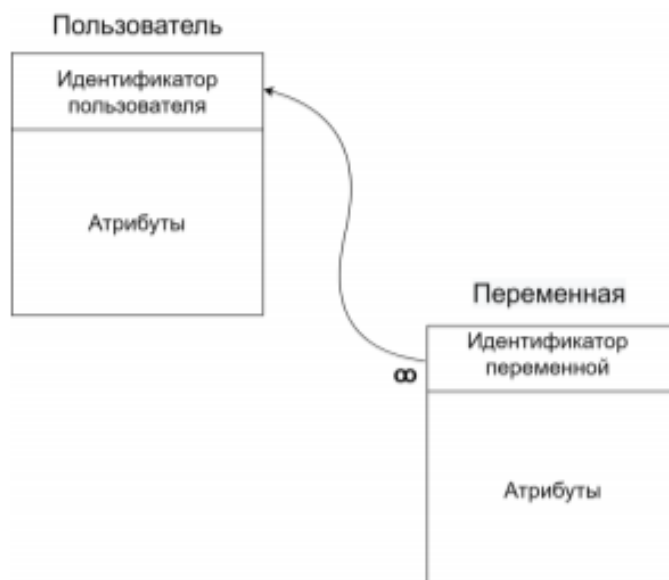


Рисунок 11

Атрибуты таблицы пользователя должны включать его логин и пароль, но также могут содержать и прочие поля, идентифицирующие пользователя. Схема этой таблицы приведена ниже (таблица 1).

Таблица 1

Название поля	Тип данных	Описание
login	Строка	Логин – уникальный идентификатор пользователя
name	Строка	Имя пользователя
passwd_hash	Строка	Пароль пользователя. Хранится не сам пароль в открытом виде, а его хэш-код

Уникальным идентификатором пользователя является его логин (первичный ключ). Существование другого пользователя с тем же логином невозможно.

Таблица, непосредственно являющаяся «удаленной областью памяти» должна иметь не только поля для названия переменной и значения, но также поле с логином пользователя, которому данная конкретная переменная принадлежит. Её описание представлено в таблице 2.

Таблица 2

Название поля	Тип данных	Описание
var_name	Строка	Имя переменной – часть первичного ключа
value	Строка	Значение переменной. Хранится только в зашифрованном виде
login	Строка	Логин пользователя, владеющего данной переменной. Внешний ключ. Часть первичного ключа

Первичный ключ в таблице «удаленной области памяти» является композицией двух атрибутов – имени переменной и ключа пользователя: таким образом достигается уникальность каждой переменной для отдельного пользователя, при том сохраняя возможность иметь другому пользователю переменную с таким же названием (идея изолированной области памяти). Полная атрибутивная логическая модель, представленных полей представлена на рисунке 12.

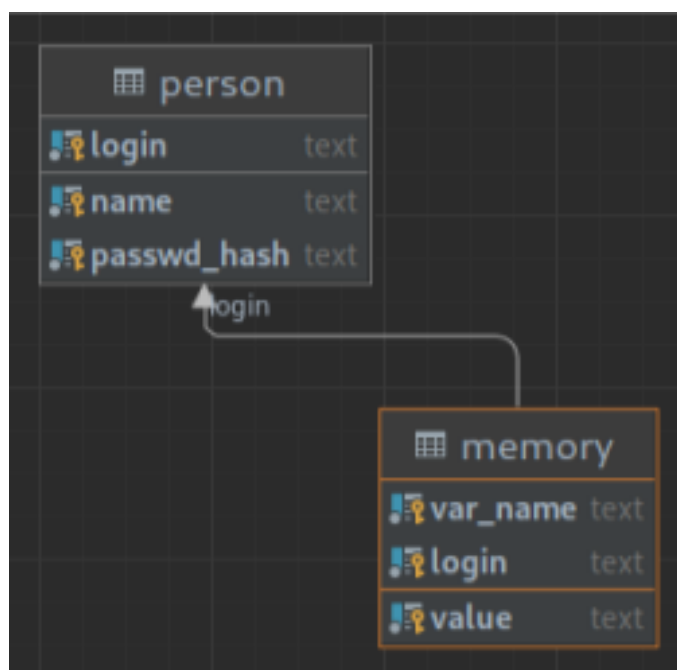


Рисунок 12

2.6 Аутентификация между клиентом и сервером

Для авторизации пользователей интерфейс использует модуль аутентификации. Этот модуль также используется и для регистрации в системе.

Авторизоваться под своей учетной записью может только пользователь, зарегистрированный в системе. Для регистрации пользователю достаточно ввести имя, логин и пароль (соответствующие атрибуты в базе данных «пользователь») и при условии, что выбранный пользователем логин не используется в системе, происходит успешное окончание регистрации. Общий алгоритм регистрации пользователя в системе выглядит следующим образом:

- 1) ввод пользователем имени;
- 2) ввод логина. В случае неудачи ему предлагается ввести другой;
- 3) ввод пароля и его подтверждение (повторный ввод);
- 4) взятие хэш-кода пароля и занесение его вместе с другими атрибутами в таблицу «пользователь».

Аутентификация основана на использовании и валидации пары логин-пароль. Для авторизации в системе пользователю достаточно ввести логин и пароль, и при условии, что введенная пара существует в базе данных, пользователь успешно завершит процедуру. Общий алгоритм авторизации пользователя выглядит следующим образом:

- 1) ввод пользователем логина;
- 2) запрос хэш-кода пароля из базы данных модулем аутентификации, используя введенный логин;
- 3) сравнение хэш-кода введенного пароля с полученным хэш кодом из базы данных;
- 4) вынесение вердикта аутентификации: гарантировать доступ в случае равенства хэш-кодов, в противном случае – в доступе отказать.

Схема аутентификации в общем виде представлена на рисунке 13.

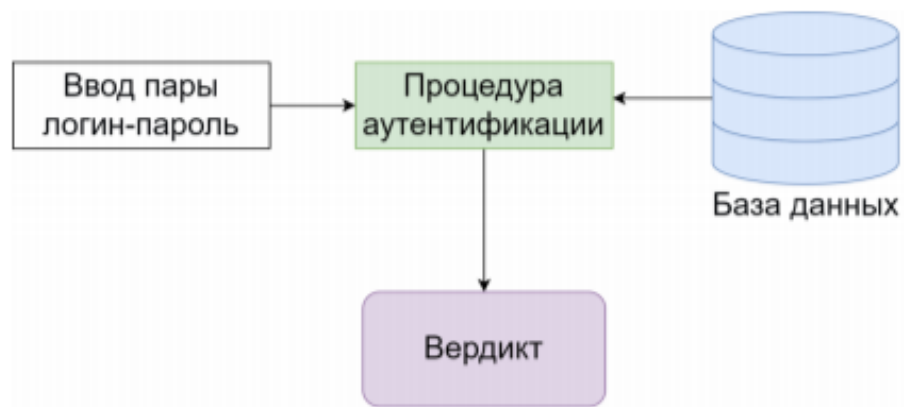


Рисунок 13

2.7 Криптографический модуль

2.7.1 Описание

Криптографический модуль представляет из себя библиотеку, позволяющую осуществлять необходимые преобразования над числами с применением к ним свойств гомоморфизма реализованной системы. В условиях поставленной задачи выполнения над переменными как можно большего числа операций была выбрана полностью гомоморфная криптосистема, основанная на свойстве гомоморфизма колец полиномов.

Идея шифрования заключается в том, что шифруемое целое число переводится в некую систему счисления, после генерируется подстановочный полином и приравнивается к основанию выбранной системы счисления. Такой подстановочный полином должен иметь единственный корень в кольце вещественных чисел: так достигается единообразное отображение множества целых чисел во множество полиномов. Результат композиции и есть представление выбранного числа в кольце.

2.7.2 Гомоморфизм колец

Ниже рассмотрены алгоритмы, необходимые для построения отображения Gom из кольца целых чисел Zn в кольцо полиномов $Zn[x]$.

Преобразование числа в полином:

1) для каждого целого числа $a \in Z$ выбирается $r \in Z: r < a$ и переводится в систему счисления с основанием r : $a_{10} = (a_m, a_{m-1}, \dots, a_0)r$;

2) полученное представление числа a в системе счисления с основанием r перезаписывается в виде $a(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_m x^m$;

3) выбирается произвольный полином $c(x)$, учитывая, что уравнение $c(x) = r$ должно иметь единственный действительный корень в кольце $Z(x)$: $E! g: c(g) = r$;

4) результат композиции $res(x) = a(c(x))$ и есть представление выбранного числа a в кольце $Z(x)$.

Преобразование полинома в число (обратный алгоритм):

1) для обратного преобразования необходимо знать r и $c(x)$ из алгоритма выше;

2) результатом будет $res(xr)$, где xr - корень $s(x) = r$.

Приведенное выше отображение Gom является полным гомоморфизмом колец Zn и $Zn[x]$, т.е.:

$$Gom(a + b) = Gom(a) + Gom(b)$$

$$Gom(a * b) = Gom(a) * Gom(b)$$

для произвольных $a, b \in Z$.

Доказательство:

Выбирается основание системы счисления r : $r < \{a, b\}$, тогда представление a и b :

$$a(x) = a_m x^m + a_{m-1} x^{m-1} + \dots + a_1 x + a_0$$

$$b(x) = b_k x^k + b_{k-1} x^{k-1} + \dots + b_1 x + b_0$$

в кольце $Z[x]$ следующее:

$$a_{10} = (a_m, a_{m-1}, \dots, a_0)$$

$$b_{10} = (b_k, b_{k-1}, \dots, b_0)$$

Положим, что $m > k$. Тогда:

1) $Gom(a) + Gom(b) = a(x) + b(x) = a_m x^m + \dots + (a_k + b_k) x^k + \dots + (a_0 + b_0) = Gom(a + b)$;

2) Аналогично сложению.

Таким образом, данная система является полностью гомоморфной и может быть взята за основу криптографического модуля.

2.7.3 Построение схемы гомоморфного шифрования

Рассмотрим данную схему гомоморфного шифрования, основанную на гомоморфизме, более подробно. Эта технология предоставляет возможность проводить вычисления над данными, не раскрывая сами данные, что делает ее уникальной и полезной в различных областях.

Основой данной схемы является возможность вычислять произвольные полиномиальные функции, обозначенные как $f(A)$, где A представляет собой целочисленный вектор $A = \{a_0, a_1, \dots, a_n\}$, а n - целое число. Процесс шифрования этого вектора данных выполняется в соответствии с алгоритмом, который преобразует числа в полиномы, а секретный ключ представлен парой $(r, c(x))$. Таким образом, вычисление функции $f(A)$ сводится к вычислению $f(Gom(A))$.

Однако следует отметить важное ограничение данной схемы: если $Ea_i : a_i < r$, то последующее шифрование теряет смысл. Этот факт является одним из недостатков системы.

Для восстановления результатов вычислений необходимо решить уравнение $c(x) = r$, и оно имеет единственное действительное решение – sr . Данное решение можно вычислить на этапе шифрования и использовать его вместо пары $(r, c(x))$.

Особенность данной криптосистемы заключается в том, что шифрование открытого текста выполняется при помощи подстановочного полинома, а дешифрование – при помощи подстановки его корня и вычисления результата. Это делает данную криптосистему асимметричной системой шифрования.

Тем не менее, стоит отметить, что раскрытие "открытого ключа" может уменьшить криптостойкость алгоритма, так как, зная подстановочный полином, задача взлома шифра сводится к его факторизации.

В рамках определенных задач пользователь удаленной системы может решить не передавать открытый ключ серверу, если нет необходимости в шифровании на стороне сервера. Это позволяет управлять уровнем безопасности и гибко настраивать доступ к данным.

3 Реализация

3.1 Выбор программных средств

3.1.1 Выбор языка программирования

В качестве языка программирования для разработки программного комплекса был выбран язык Python (3.10). Данный выбор был сделан вследствие целого ряда причин:

- широкий выбор стандартных библиотек языка и инструментов языка, ускоряющих процесс разработки и отладки программы;
- большое число функциональных сторонних библиотек (модулей), которые расширяют возможности работы с математическими структурами данных и облегчают написание программного интерфейса приложения;
- объектно-ориентированная часть языка для написания собственных классов и структур;
- независимость языка от платформы, при наличии интерпретатора и пакетного менеджера;
- возможность быстрой контейнеризации собственного приложения для облегчения процесса установки конечным пользователем (или для запуска сервиса внутри отказоустойчивого кластера);
- удобство языка и личные предпочтения. Данный язык использовался для написания всех модулей системы, в том числе реализации математического аппарата криптографического модуля.

3.1.2 Используемые библиотеки

Для реализации программного комплекса использовалось много библиотек как самого языка Python, так и сторонних. Среди них можно выделить основные:

- NumPy [5], предназначенная для работы со сложными математическими структурами. Использовалась для написания криптографического модуля и выполнения некоторых математических операций на клиенте;

- psycopg2 [6], адаптер СУБД PostgreSQL. Использовалась для выполнения SQL-запросов к базе данных;
- flask [7], являющаяся WSGI фреймворком для создания приложений, работающих через веб. Программный интерфейс приложения исполняется внутри него;
- requests для работы по протоколу HTTP/HTTPS;
- unittest, встроенный микрофреймворк. Использовался для написания тестов для отладки криптографического модуля;
- colorama, используемая для визуального вывода удачных и неудачных запросов (окраска текста в консоли).

3.1.3 Реализация базы данных

Для реализации использовалась реляционная база данных работающая под управлением СУБД Postgres. Была выбрана из-за удобства инструмента psycopg2, упомянутого выше, который предоставлял методы для работы с базой данных.

Выбранная СУБД обладает естественным недостатком: создание экземпляра базы данных невозможно без установки громоздкого сервера системы Postgres. Для решения этой задачи было принято решение упаковать экземпляр базы данных в контейнер. В качестве платформы для контейнеризации был выбран Docker, как самый простой виртуализатор. Описание инструкции по сборке контейнера приведено ниже, на рисунке 14.

```
FROM postgres
ENV POSTGRES_PASSWORD 1234
ENV POSTGRES_DB secure_remote_memory
COPY secure_remote_memory_dump.sql /docker-entrypoint-initdb.d/
```

Рисунок 14

3.2 Клиентская библиотека

Клиентская библиотека представляет из себя класс class ClientConn(self, ip, port) с реализованными методами, находящимися в файле clientside/client.py.

Параметры класса:

- ip (str) – адрес сервера;
- port(str) – порт, прослушиваемый сервером.

Описание: класс для установления соединения с сервером и выполнения запросов клиента. Используется демонстрационной программой.

Методы:

- def sign_up(self) – метод для регистрации пользователя в системе;
- def sign_in(self) – метод для авторизации пользователя в системе;
- def sign_out(self) – метод для выхода из учетной записи;
- def init_new_variable(self) – метод для создания новой переменной в удаленной защищенной памяти;
- def get_memory(self) – метод, выводящий на экран все имеющиеся в распоряжении пользователя переменные;
- def get_variable(self) – метод, выводящий на экран значение переменной по ее названию;
- def edit_value(self) – метод для изменения значения уже созданной переменной;
- def delete_variable(self) – метод для удаления переменной;
- def calc_variable(self) – метод, реализующий набор операций по работе с переменными. Выводит на экран получившийся результат вычислений и сохраняет его в переменную, указанную пользователем;
- def update_keys(self) – метод, выполняющий генерацию новой пары ключей. При выполнении этой процедуры происходит создание локальной копии удаленной защищенной области с последующим перешифрованием и отправкой на сервер;
- def get_key(self, key) – параметр: key (str) – название выводимого на экран ключа метод для вывода на экран ключа пользователя по запросу.

Также реализован дополнительный функционал, который используется внутри класса ClientConn. Это набор утилитарных функций, выделенный в отдельный файл – client_utils.py.

Демонстрационная программа, использующая перечисленные выше функции и методы класса, находится в файле `clientside/demo_app.py`.

3.3 Описание сервера

Программный интерфейс сервера описан в файле `serverside/rest_api.py` представляет набор функций-обработчиков:

- `@app.route("/create_user", methods=["POST"]): def create_user()` – функция, вызывающая в обработчике метод для создания нового пользователя в базе данных, возвращает: ответ сервера;
- `@app.route("/get_user_info", methods=["GET"]): def get_user_info()` – функция, вызывающая в обработчике метод для возвращения записи о пользователе в базе данных по принятому логину, возвращает: упакованную в формат json запись о пользователе;
- `@app.route("/check_variable_exist", methods=["GET"]): def check_variable_exist()` – функция, вызывающая в обработчике метод для возвращения флага существования переменной по принятому названию, возвращает: упакованный в формат json флаг;
- `@app.route("/get_variable", methods=["GET"]): def get_variable()` – функция, вызывающая в обработчике метод для возвращения записи о переменной в базе данных по принятому названию, возвращает: упакованную в формат json запись о переменной;
- `@app.route("/get_memory", methods=["GET"]): def get_memory()` – функция, вызывающая в обработчике метод для возвращения всех записей из таблицы “memory”, возвращает: упакованные в формат json записи;
- `@app.route("/init_new_variable", methods=["POST"]): def init_new_variable()` – функция вызывающая в обработчике метод для создания новой записи в таблице “memory”, возвращает: ответ от сервера;
- `@app.route("/edit_value", methods=["POST"]): def edit_value()` – функция, вызывающая в обработчике метод для изменения атрибута value в таблице “memory” по принятому названию, возвращает: ответ от сервера;

– `@app.route("/delete_variable", methods=["GET"]): def delete_variable()` – функция, вызывающая в обработчике метод для удаления записи в таблице “memory” по принятому названию, возвращает: ответ от сервера;

– `@app.route("/calc_variables", methods=["POST"]): def calc_variables()` – функция, вызывающая в обработчике метод для выполнения над переменными по принятым значениям математические операции (кроме деления в силу специфики деления полиномов), возвращает: упакованный в формат json зашифрованный результат;

– `@app.route("/div_variables", methods=["POST"]): def div_variables()` – функция, вызывающая в обработчике метод для выполнения операции деления над переменными, возвращает: упакованный в формат json зашифрованный результат.

3.4 Инструкция по установке

Для установки необходимо выполнить следующие программные требования:

- 1) иметь на рабочей машине интерпретатор python >3.10 и пакетный менеджер pip версии >22.1.1;
- 2) иметь установленную платформу Docker >20.10.

Инструкция по установке python:

- 1) чтобы проверить установлен ли python на компьютере откройте терминал и выполните команду `python --version`, в случае если приведен результат, изображенный на рисунке 15;

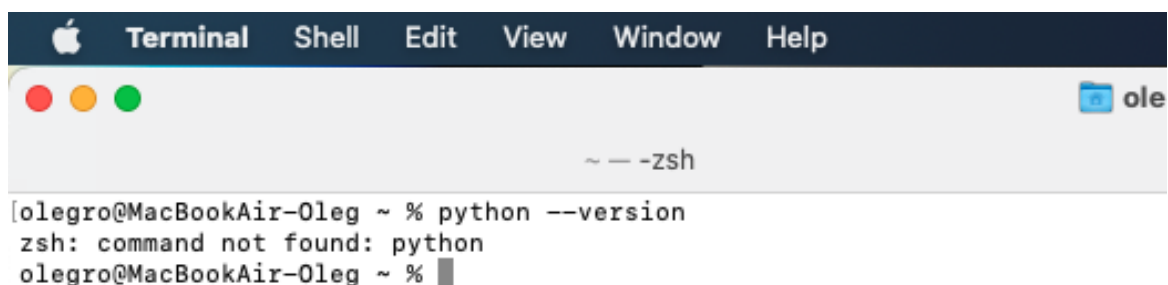


Рисунок 15
37

2) для установки Python перейдите на официальный сайт <https://www.python.org/>, выберите пункт «Downloads» и скачайте установщик последней версии Python для вашей операционной системы. Скриншот официального сайта представлен на рисунке 16;

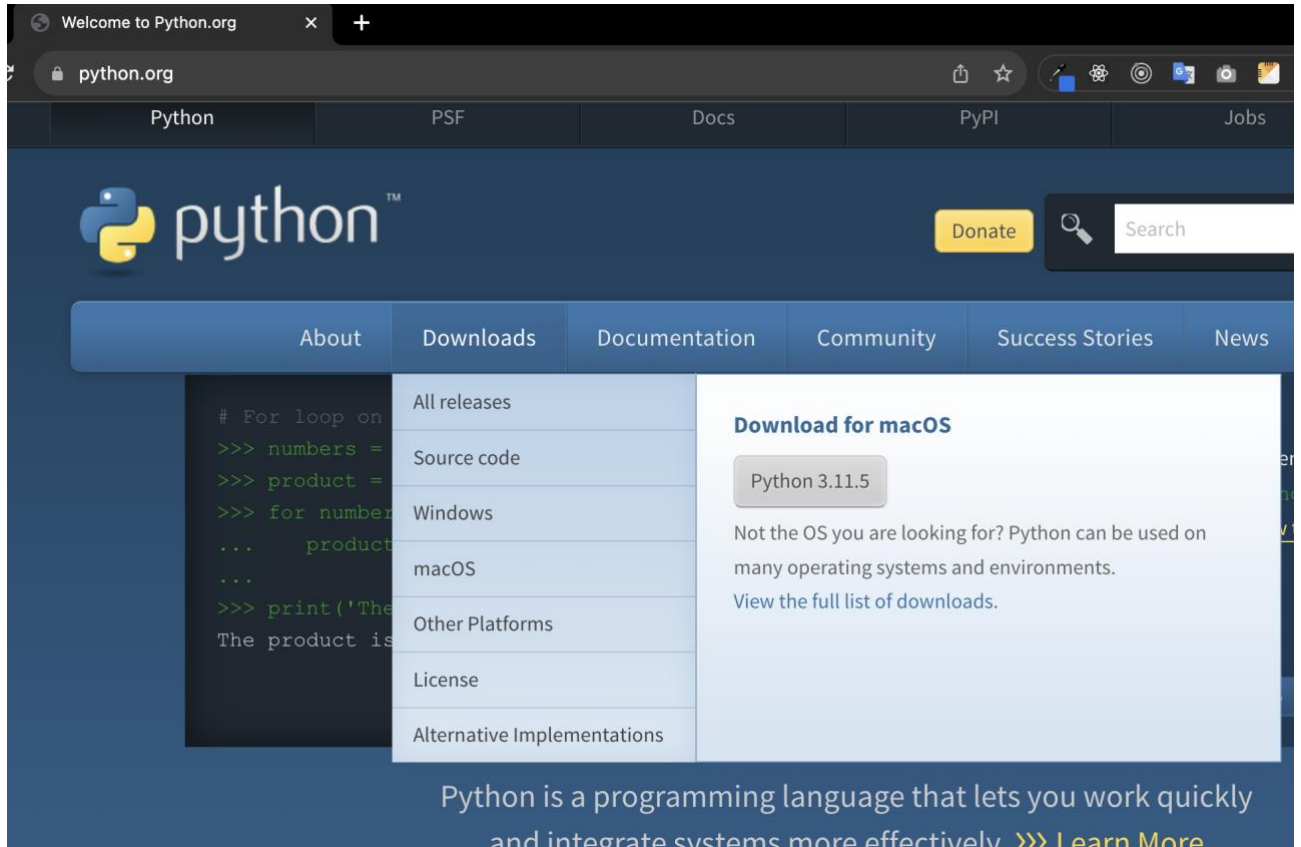


Рисунок 16

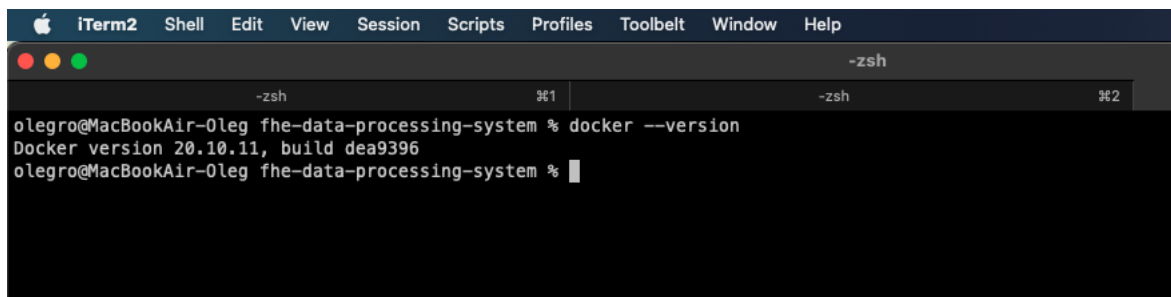
3) запустите скачанный файл и следуйте инструкциям;
4) проверьте, что установка прошла успешно, и установлена подходящая версия Python, выполнив команду `python --version`. Результат успешного выполнения представлен на рисунке 17.

```
~/Projects/fhe-data-processi
[olegro@MacBookAir-Oleg ~ % python3 --version
Python 3.11.5
olegro@MacBookAir-Oleg ~ %
```

Рисунок 17

Инструкция по установке docker:

1) чтобы проверить установлен ли Docker на вашем компьютере - откройте терминал и выполните команду `docker --version`. Результат выполнения при установленном Docker приведен на рисунке 18;



```
olegro@MacBookAir-0leg fhe-data-processing-system % docker --version
Docker version 20.10.11, build dea9396
olegro@MacBookAir-0leg fhe-data-processing-system %
```

Рисунок 18

2) для установки docker перейдите на официальный сайт <https://docs.docker.com/get-docker/> и скачайте установщик Docker для своей операционной системы. Скриншот страницы для скачивания представлен на рисунке 19;

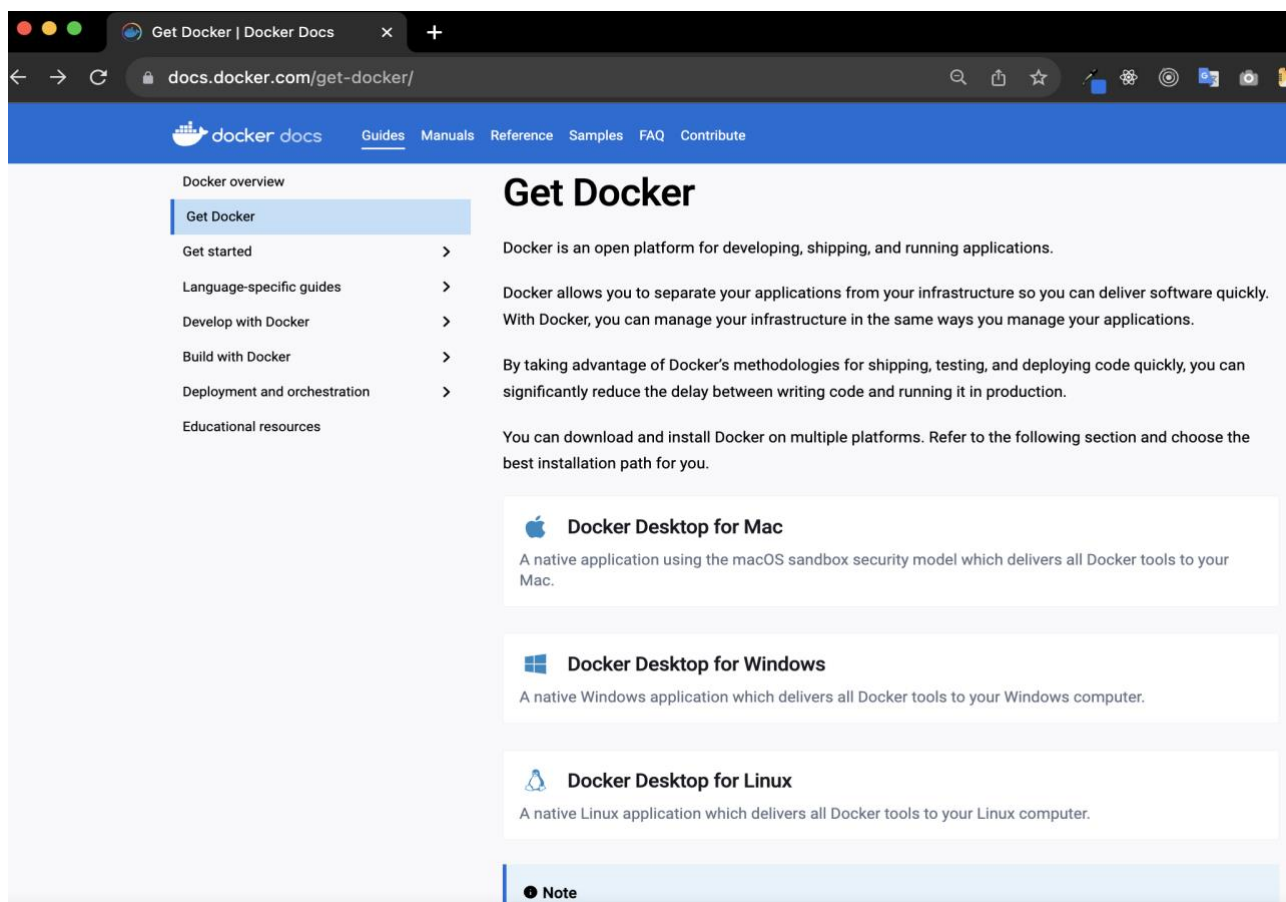
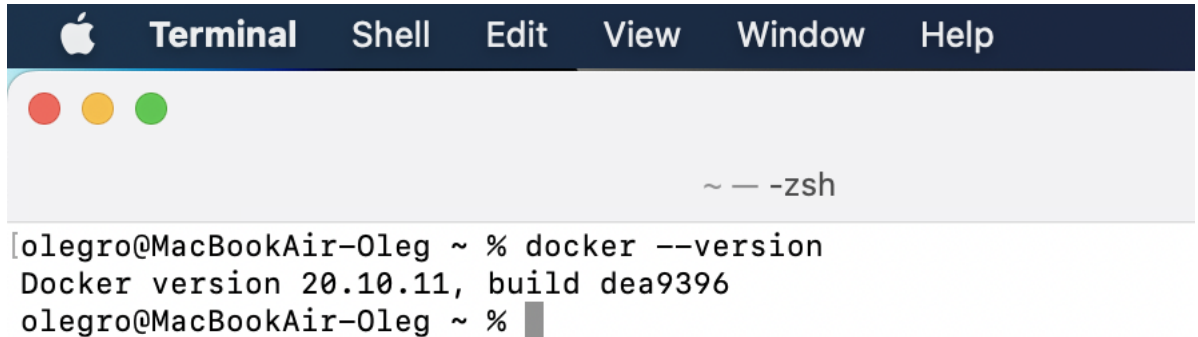


Рисунок 19

- 3) запустите установочный файл и следуйте инструкциям;
- 4) после установки выполните команду `docker --version`, чтобы убедиться, что установка прошла успешно. Результат успешной установки представлен на рисунке 20.

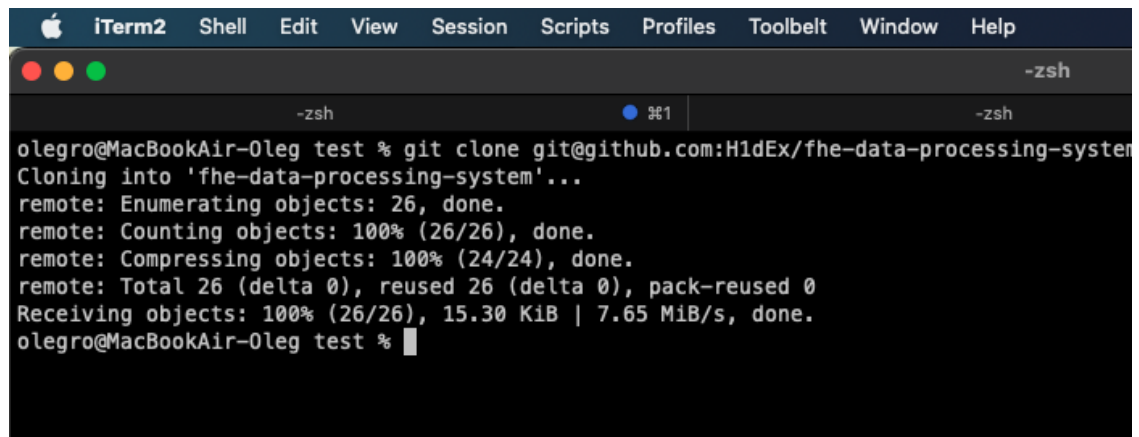


```
Terminal  Shell  Edit  View  Window  Help
~ — -zsh
[olegro@MacBookAir-Oleg ~ % docker --version
Docker version 20.10.11, build dea9396
olegro@MacBookAir-Oleg ~ %
```

Рисунок 20

После выполнения требований можно приступать к самой установке. Инструкция по установке и запуску приложения.

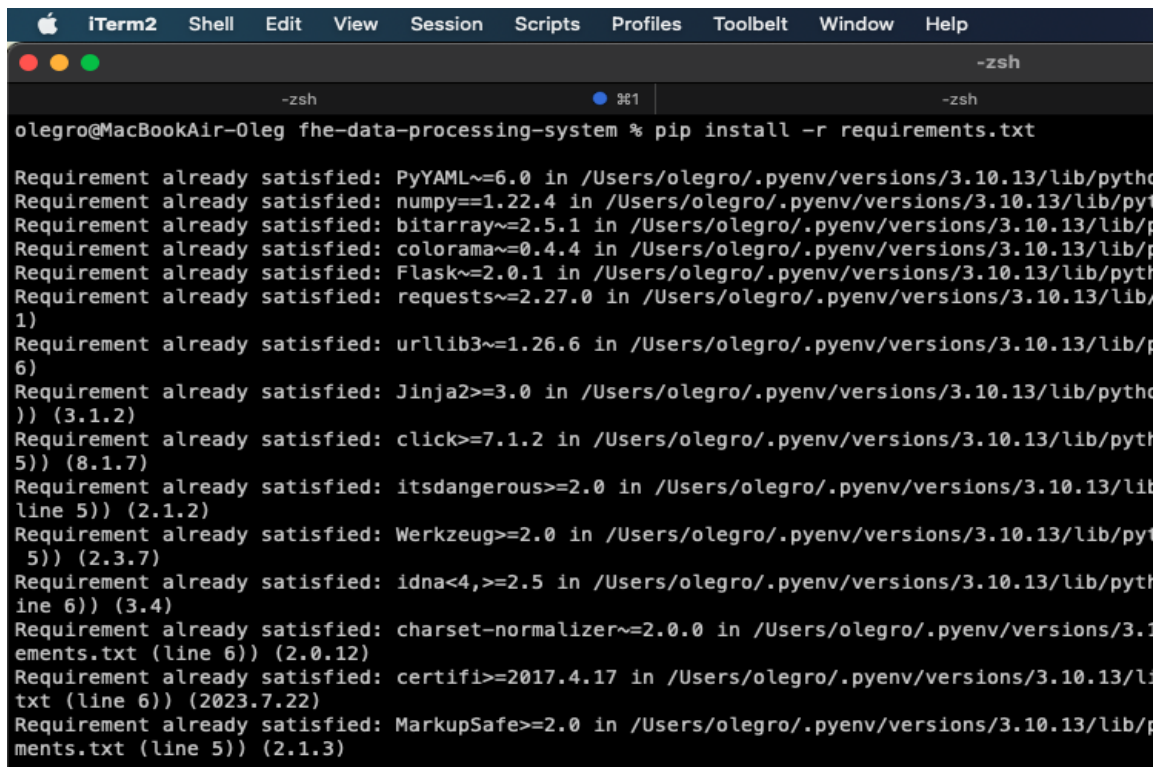
1. Выгрузить код приложения с репозитория <https://github.com/H1dEx/fhe-data-processing-system>. Результат выполнения представлен на рисунке 21.



```
iTerm2  Shell  Edit  View  Session  Scripts  Profiles  Toolbelt  Window  Help
~ — -zsh
olegro@MacBookAir-Oleg test % git clone git@github.com:H1dEx/fhe-data-processing-system
Cloning into 'fhe-data-processing-system'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 26 (delta 0), reused 26 (delta 0), pack-reused 0
Receiving objects: 100% (26/26), 15.30 KiB | 7.65 MiB/s, done.
olegro@MacBookAir-Oleg test %
```

Рисунок 21

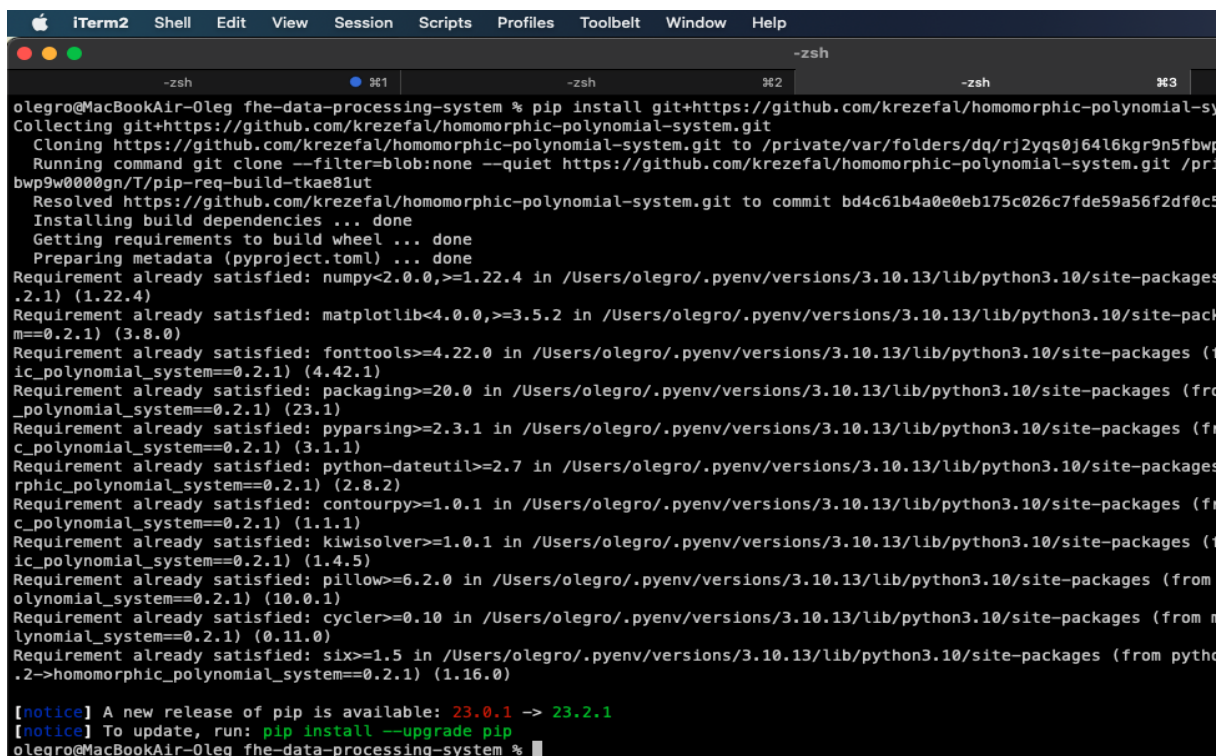
2. Далее необходимо установить зависимости и сторонние библиотеки, необходимые для работы системы. Запустите следующую команду из корня репозитория: `pip install -r requirements.txt`. Результат выполнения представлен на рисунке 22.



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
~zsh
olegro@MacBookAir-0leg fhe-data-processing-system % pip install -r requirements.txt
Requirement already satisfied: PyYAML~=6.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (6.0.1)
Requirement already satisfied: numpy==1.22.4 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (1.22.4)
Requirement already satisfied: bitarray~=2.5.1 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.5.1)
Requirement already satisfied: colorama~=0.4.4 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (0.4.4)
Requirement already satisfied: Flask~=2.0.1 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.0.1)
Requirement already satisfied: requests~=2.27.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.27.0)
Requirement already satisfied: urllib3~=1.26.6 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (1.26.6)
Requirement already satisfied: Jinja2>=3.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (3.1.2)
Requirement already satisfied: click>=7.1.2 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (8.1.7)
Requirement already satisfied: itsdangerous>=2.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.1.2)
Requirement already satisfied: Werkzeug>=2.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.3.7)
Requirement already satisfied: idna<4,>=2.5 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (3.4)
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.1.3)
```

Рисунок 22

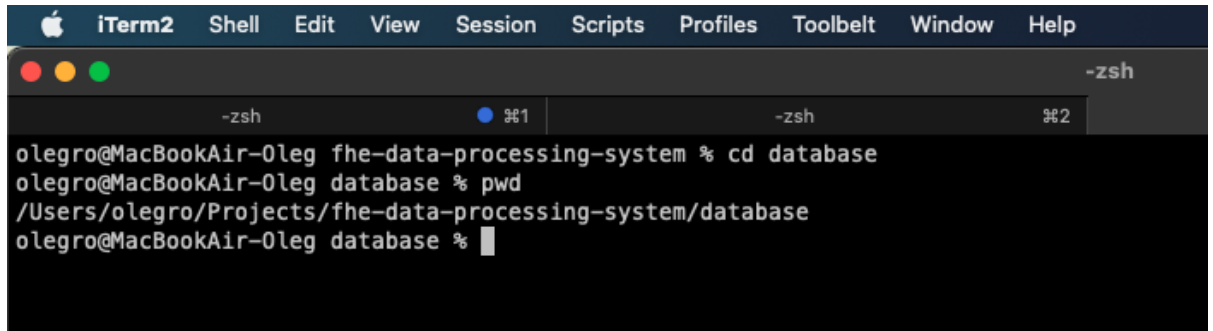
3. Для установки криптографического модуля введите следующую команду: `pip install git+https://github.com/H1dEx/homomorphic-polynomial-system.git`. Результат выполнения представлен на рисунке 23.



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
~zsh
olegro@MacBookAir-0leg fhe-data-processing-system % pip install git+https://github.com/krezefal/homomorphic-polynomial-system.git
Collecting git+https://github.com/krezefal/homomorphic-polynomial-system.git
  Cloning https://github.com/krezefal/homomorphic-polynomial-system.git to /private/var/folders/dq/rj2yqs0j64l6kgrn5f5bw9w0000gn/T/pip-req-build-tkae81ut
  Running command git clone --filter=blob:none --quiet https://github.com/krezefal/homomorphic-polynomial-system.git /private/var/folders/dq/rj2yqs0j64l6kgrn5f5bw9w0000gn/T/pip-req-build-tkae81ut
  Resolved https://github.com/krezefal/homomorphic-polynomial-system.git to commit bd4c61b4a0eb175c026c7fde59a56f2df0c3
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: numpy<2.0.0,>=1.22.4 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (1.22.4)
Requirement already satisfied: matplotlib<4.0.0,>=3.5.2 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (3.8.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (4.42.1)
Requirement already satisfied: packaging>=20.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (2.8.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (1.1.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (10.0.1)
Requirement already satisfied: cycycler>=0.10 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (0.11.0)
Requirement already satisfied: six>=1.5 in /Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages (1.16.0)
[notice] A new release of pip is available: 23.0.1 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
olegro@MacBookAir-0leg fhe-data-processing-system %
```

Рисунок 23

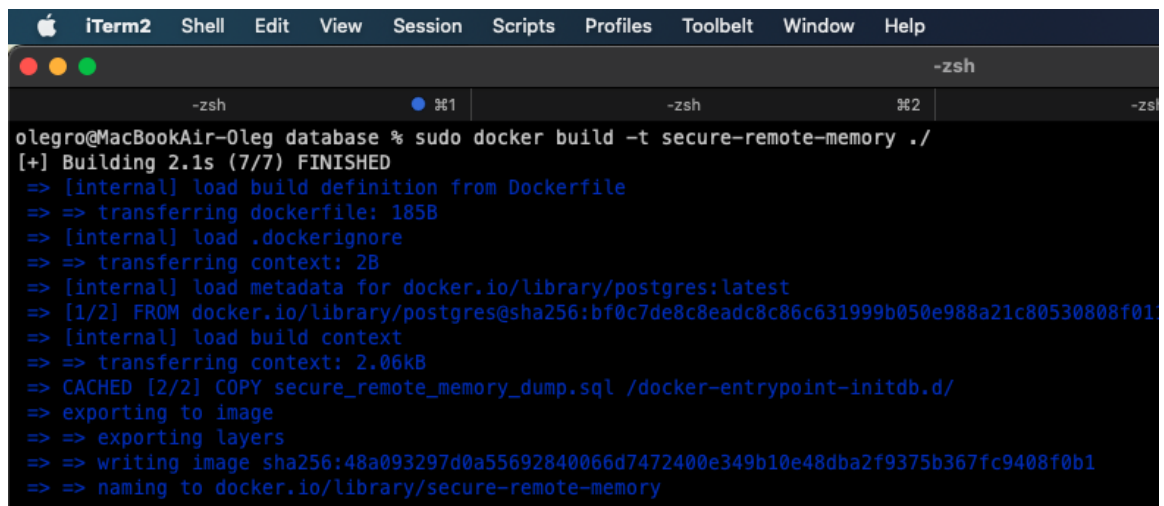
4. Далее необходимо перейти в директорию database выполнив команду `cd database`. Результат выполнения представлен на рисунке 24.



```
iTerm2  Shell  Edit  View  Session  Scripts  Profiles  Toolbelt  Window  Help
-zsh
-zsh  %1  -zsh  %2
olegro@MacBookAir-Oleg fhe-data-processing-system % cd database
olegro@MacBookAir-Oleg database % pwd
/Users/olegro/Projects/fhe-data-processing-system/database
olegro@MacBookAir-Oleg database %
```

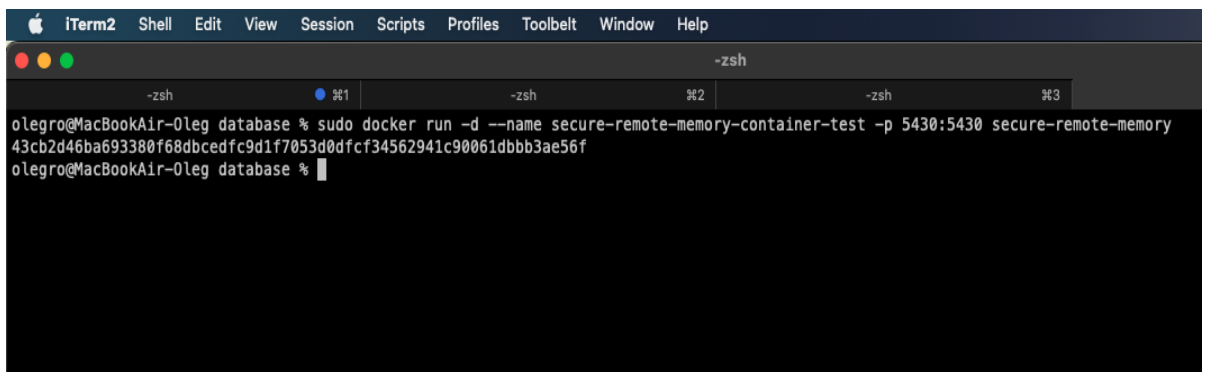
Рисунок 24

5. Далее необходимо собрать и запустить докер-контейнер. Чтобы собрать и запустить контейнер выполните: `sudo docker build -t secure-remote-memory ./` `sudo docker run -d --name secure-remote-memory-container -p 5432:5432 secure-remote-memory`. Результат выполнения представлен на рисунках 25 и 26.



```
iTerm2  Shell  Edit  View  Session  Scripts  Profiles  Toolbelt  Window  Help
-zsh
-zsh  %1  -zsh  %2  -zsh
olegro@MacBookAir-Oleg database % sudo docker build -t secure-remote-memory ./
[+] Building 2.1s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 185B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/postgres:latest
=> [1/2] FROM docker.io/library/postgres@sha256:bf0c7de8c8eadc8c86c631999b050e988a21c80530808f01
=> [internal] load build context
=> => transferring context: 2.06kB
=> CACHED [2/2] COPY secure_remote_memory_dump.sql /docker-entrypoint-initdb.d/
=> exporting to image
=> => exporting layers
=> => writing image sha256:48a093297d0a55692840066d7472400e349b10e48dba2f9375b367fc9408f0b1
=> => naming to docker.io/library/secure-remote-memory
```

Рисунок 25



```
iTerm2  Shell  Edit  View  Session  Scripts  Profiles  Toolbelt  Window  Help
-zsh
-zsh  %1  -zsh  %2  -zsh  %3
olegro@MacBookAir-Oleg database % sudo docker run -d --name secure-remote-memory-container-test -p 5430:5430 secure-remote-memory
43cb2d46ba693380f68dbcedfc9d1f7053d0dfcf34562941c90061dbbb3ae56f
olegro@MacBookAir-Oleg database %
```

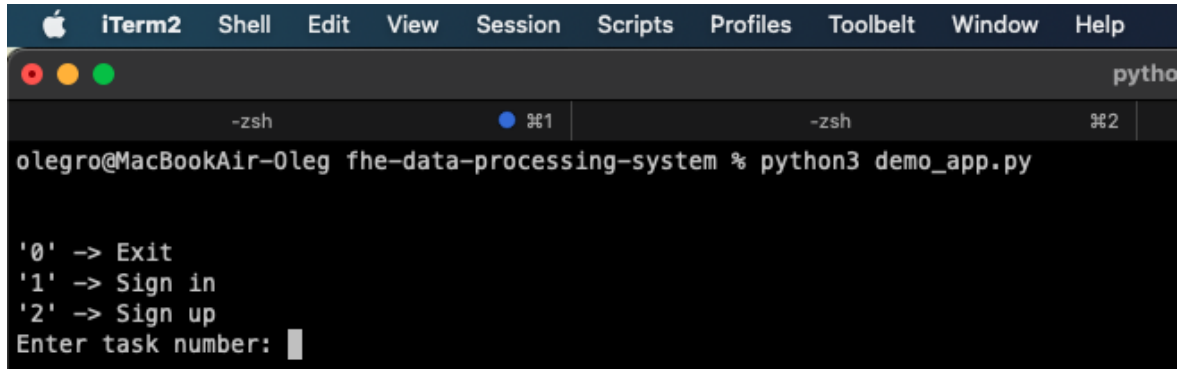
Рисунок 26

[illegible]

7. Теперь можно запустить сервер. Для этого необходимо перейти в корневую папку выполнив команду `cd ..` и, находясь в корне проекта запустить сервер, выполнив команду `python3 ./serverside/rest_api.py`. Пример выполнения команды представлен на рисунке 28.

Рисунок 28

8. Далее необходимо запустить демонстрационное приложение. Для этого нужно открыть второе окно терминала, перейти в корневую директорию проекта и выполнить команду `python3 demo_app.py`. Результат выполнения представлен на рисунке 29.



```
iTerm2  Shell  Edit  View  Session  Scripts  Profiles  Toolbelt  Window  Help
-zsh  %1  -zsh  %2
olegro@MacBookAir-Oleg fhe-data-processing-system % python3 demo_app.py

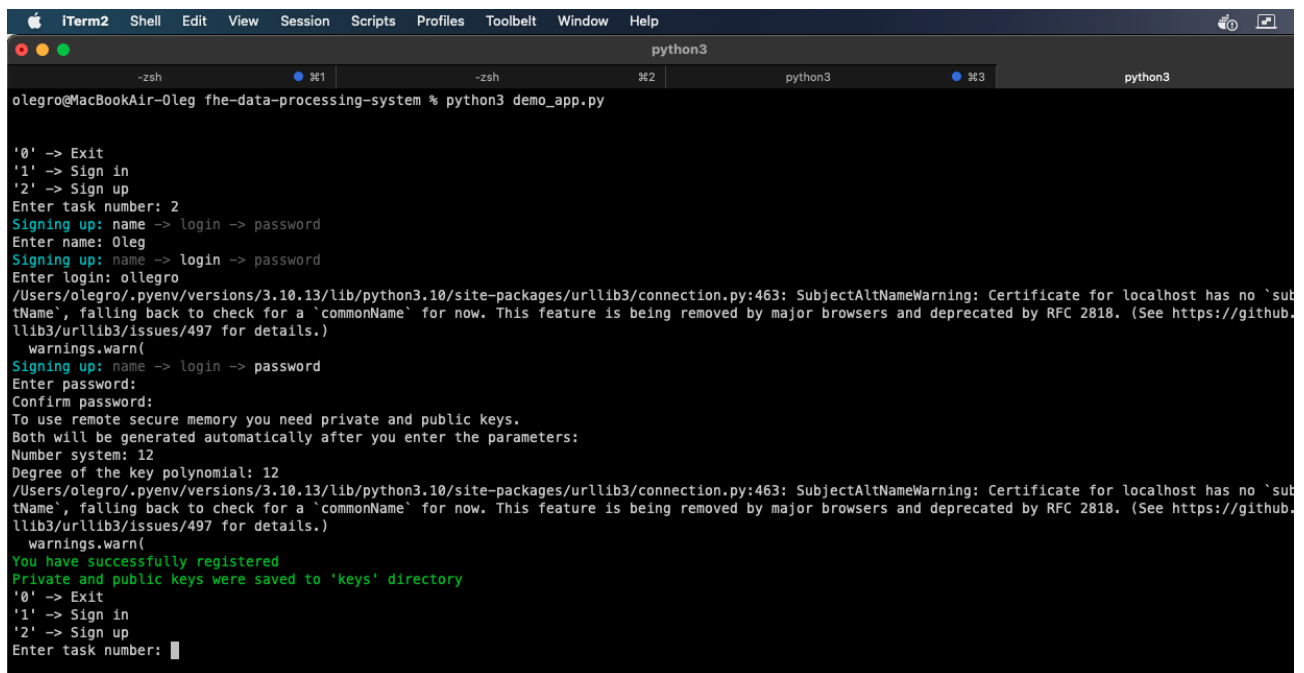
'0' -> Exit
'1' -> Sign in
'2' -> Sign up
Enter task number: 
```

Рисунок 29

4 Тестирование

Разработанный программный комплекс прошел функциональное тестирование. Функциональное тестирование – это тестирование программного обеспечения с целью проверки соблюдения функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, поставленные перед ним на этапе проектирования.

Была проведена проверка на создание пользователя. Результаты тестирования представлены на рисунке 30.



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
python3
-zsh #1 -zsh #2 python3 #3 python3
olegro@MacBookAir-Oleg fhe-data-processing-system % python3 demo_app.py

'0' -> Exit
'1' -> Sign in
'2' -> Sign up
Enter task number: 2
Signing up: name -> login -> password
Enter name: Oleg
Signing up: name -> login -> password
Enter login: ollegro
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning: Certificate for localhost has no `subjectName`, falling back to check for a `commonName` for now. This feature is being removed by major browsers and deprecated by RFC 2818. (See https://github.com/urllib3/urllib3/issues/497 for details.)
  warnings.warn(
Signing up: name -> login -> password
Enter password:
Confirm password:
To use remote secure memory you need private and public keys.
Both will be generated automatically after you enter the parameters:
Number system: 12
Degree of the key polynomial: 12
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning: Certificate for localhost has no `subjectName`, falling back to check for a `commonName` for now. This feature is being removed by major browsers and deprecated by RFC 2818. (See https://github.com/urllib3/urllib3/issues/497 for details.)
  warnings.warn(
You have successfully registered
Private and public keys were saved to 'keys' directory
'0' -> Exit
'1' -> Sign in
'2' -> Sign up
Enter task number: █
```

Рисунок 30

В ходе данного этапа успешно выполнена регистрация нового пользователя ollegro. Основываясь на данную проверку, был сделан вывод, что функционал регистрации работает корректно. Таким образом, проверка пройдена успешно.

Была проведена проверка на авторизацию пользователя. Результаты тестирования представлены на рисунке 31.

```

'0' -> Exit
'1' -> Sign in
'2' -> Sign up
Enter task number: 1
Login: olegro
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463:
tName', falling back to check for a `commonName` for now. This feature is being removed by ma
llib3/urllib3/issues/497 for details.)
  warnings.warn(
Password:
You have successfully authorize
Specify the path to private key file or press enter to use default location:
Specify the path to public key file or press enter to use default location:
Keys has been added

```

Рисунок 31

В ходе данного этапа успешно выполнена авторизация для зарегистрированного пользователя olegro. Основываясь на данную проверку, был сделан вывод, что функционал авторизации работает корректно. Таким образом, проверка пройдена успешно.

Была проведена проверка на создание переменной. Результат представлен на рисунке 32.

```

'0' -> Sign out
'1' -> Initialize new variable
'2' -> Get variable list
'3' -> Get variable
'4' -> Edit value
'5' -> Delete variable
'6' -> a + b
'7' -> a - b
'8' -> a * b
'9' -> a / b
'10' -> Get private key (root)
'11' -> Get public key (polynomial)
'12' -> Update private and public keys

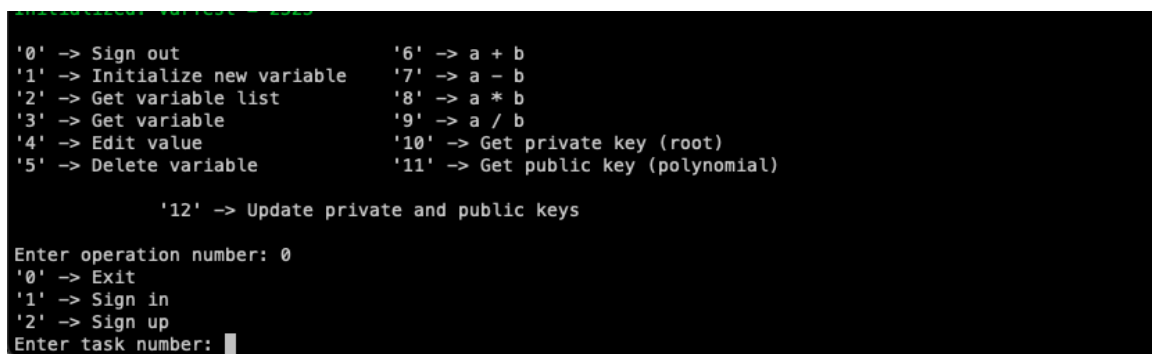
Enter operation number: 1
Enter a variable name: testOne
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3
tName', falling back to check for a `commonName` for now. This feature is
llib3/urllib3/issues/497 for details.)
  warnings.warn(
Enter a value: 1111
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3
tName', falling back to check for a `commonName` for now. This feature is
llib3/urllib3/issues/497 for details.)
  warnings.warn(
Initialized: testOne = 1111

```

Рисунок 32

В ходе данного этапа успешно выполнено создание переменной testOne. Основываясь на данную проверку, был сделан вывод, что функционал создания переменных работает корректно. Таким образом, проверка пройдена успешно.

Был протестирован выход из учетной записи. Результат тестирования представлен на рисунке 33.



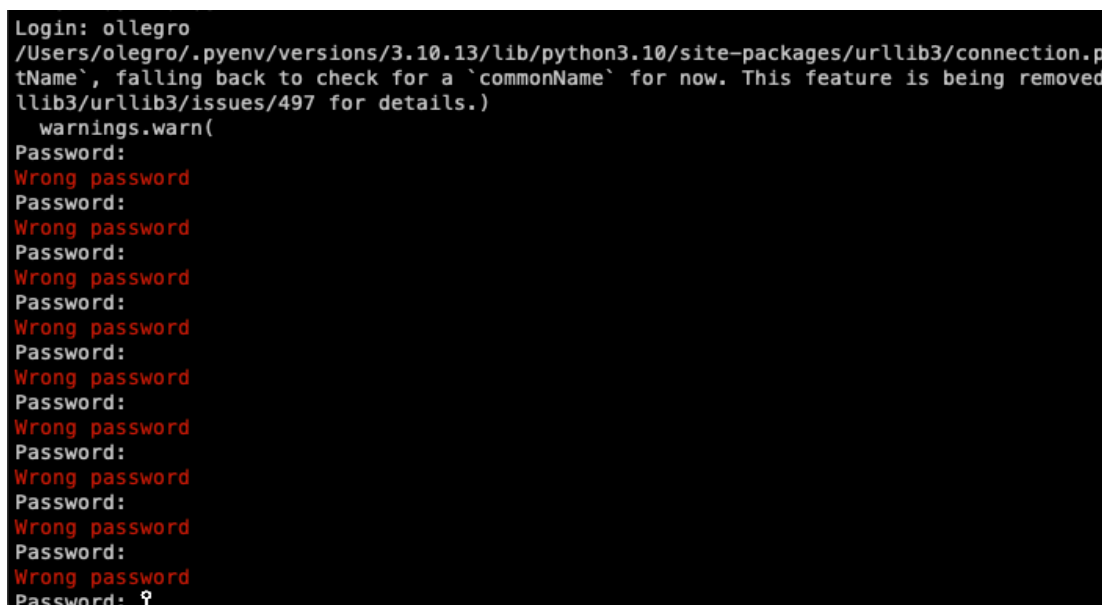
```
'0' -> Sign out
'1' -> Initialize new variable
'2' -> Get variable list
'3' -> Get variable
'4' -> Edit value
'5' -> Delete variable
'6' -> a + b
'7' -> a - b
'8' -> a * b
'9' -> a / b
'10' -> Get private key (root)
'11' -> Get public key (polynomial)
'12' -> Update private and public keys

Enter operation number: 0
'0' -> Exit
'1' -> Sign in
'2' -> Sign up
Enter task number: 1
```

Рисунок 33

В ходе данного этапа был успешно выполнен выход из учетной записи. Основываясь на результаты данной проверки, был сделан вывод, что выход из учетной записи работает корректно. Таким образом, проверка пройдена успешно.

Протестированы попытки несанкционированного доступа – с логином, существующего пользователя и с несуществующим логином. Результаты представлены на рисунках 34 и 35.



```
Login: ollegro
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.p
tName`, falling back to check for a `commonName` for now. This feature is being removed
llib3/urllib3/issues/497 for details.)
warnings.warn(
Password:
Wrong password
Password:
Wrong password
Password:
Wrong password
Password:
Wrong password
Password:
Wrong password
Password:
Wrong password
Password:
Wrong password
Password:
Wrong password
Password:
Wrong password
Password: ?
```

Рисунок 34

В ходе данной проверки была произведена попытка авторизации в учетную запись, используя правильный логин и неверный пароль. Система не выполнила авторизацию, что является корректным поведением. Таким образом, проверка пройдена успешно.

```
olegro@MacBookAir-Oleg fhe-data-processing-system % python3 ./demo_app.py
'0' -> Exit
'1' -> Sign in
'2' -> Sign up
Enter task number: 1
Login: test
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning:
urllib3/urllib3/issues/497 for details.)
  warnings.warn(
User not found
Login: user
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning:
urllib3/urllib3/issues/497 for details.)
  warnings.warn(
User not found
Login: 
```

Рисунок 35

В ходе данной проверки была произведена попытка авторизации в учетную запись с несуществующим логином. Система не выполнила авторизацию, что является корректным поведением. Таким образом, проверка пройдена успешно.

Протестирована попытка получения переменной с несуществующим именем. Результат представлен на рисунке 36.

```
'0' -> Sign out
'1' -> Initialize new variable
'2' -> Get variable list
'3' -> Get variable
'4' -> Edit value
'5' -> Delete variable
'6' -> a + b
'7' -> a - b
'8' -> a * b
'9' -> a / b
'10' -> Get private key (root)
'11' -> Get public key (polynomial)
'12' -> Update private and public keys

Enter operation number: 3
Enter a variable name: notfound
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning:
urllib3/urllib3/issues/497 for details.)
  warnings.warn(
No variable with name 'notfound'
```

Рисунок 36

В ходе данной проверки была произведена попытка получения значения переменной с несуществующим именем. Система вернула ошибку, соответствующую корректному поведению. Таким образом, проверка пройдена успешно.

Протестирована попытка создания переменной с уже используемым именем. Результат представлен на рисунке 37.

```
Enter operation number: 1
Enter a variable name: testwo
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/alembic/util/sqlalchemy.py:125: SAWarning: Can't
alling back to check for a `commonName` for now. This feature is
es/497 for details.)
  warnings.warn(
Value 'testwo' already initialized
```

Рисунок 37

На данном этапе была произведена попытка создания переменной с уже используемым именем. Система вернула ошибку, соответствующую ожидаемому поведению. Таким образом, проверка пройдена успешно.

Протестировано получение списка переменных пользователя. Результат представлен на рисунке 38.

```
'0' -> Sign out
'1' -> Initialize new variable
'2' -> Get variable list
'3' -> Get variable
'4' -> Edit value
'5' -> Delete variable

'6' -> a + b
'7' -> a - b
'8' -> a * b
'9' -> a / b
'10' -> Get private key (root)
'11' -> Get public key (polynomial)

'12' -> Update private and public keys

Enter operation number: 2
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/compat.py:497: DeprecationWarning: Calling back to check for a `commonName` for now. This feature is being removed in 497 for details.)
  warnings.warn(
#1: testOne = 38929497.00765468
#2: testtwo = 5572813.310378334
#3: test3 = 21222.13191008295
#4: new = 2366.68132334255

'0' -> Sign out
'1' -> Initialize new variable
'2' -> Get variable list
'3' -> Get variable
'4' -> Edit value
'5' -> Delete variable

'6' -> a + b
'7' -> a - b
'8' -> a * b
'9' -> a / b
'10' -> Get private key (root)
'11' -> Get public key (polynomial)

'12' -> Update private and public keys
```

Рисунок 38

На данном этапе было выполнено получение списка переменных пользователей. Операция выполнена успешно, были получены все переменные текущего пользователя в расшифрованном виде. Таким образом, проверка пройдена успешно.

Протестирован функционал получения значения переменной по ключу. Результат представлен на рисунке 39.

```
'0' -> Sign out          '6' -> a + b
'1' -> Initialize new variable  '7' -> a - b
'2' -> Get variable list      '8' -> a * b
'3' -> Get variable          '9' -> a / b
'4' -> Edit value            '10' -> Get private key (root)
'5' -> Delete variable        '11' -> Get public key (polynomial)

'12' -> Update private and public keys

Enter operation number: 3
Enter a variable name: new
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3
alling back to check for a `commonName` for now. This feature is being
es/497 for details.)
  warnings.warn(
new = 2366.68132334255
```

Рисунок 39

На данном этапе было выполнено получение значения переменной по ее ключу. Операция выполнена успешно, было получено значение переменной new в расшифрованном виде. Таким образом, проверка пройдена успешно.

Протестирован функционал изменения переменной. Результат представлен на рисунке 40.

```

'0' -> Sign out          '6' -> a + b
'1' -> Initialize new variable '7' -> a - b
'2' -> Get variable list   '8' -> a * b
'3' -> Get variable        '9' -> a / b
'4' -> Edit value          '10' -> Get private key (
'5' -> Delete variable     '11' -> Get public key (p

                '12' -> Update private and public keys

Enter operation number: 4
Enter a variable name: new
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-p
alling back to check for a `commonName` for now. This featu
es/497 for details.)
    warnings.warn(
Enter a value: 555
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-p
alling back to check for a `commonName` for now. This featu
es/497 for details.)
    warnings.warn(
Edited: new = 555

```

Рисунок 40

На данном этапе было выполнено изменение значения переменной. Операция выполнена успешно, было изменено значение переменной new. Таким образом, проверка пройдена успешно.

Протестирован функционал удаления переменной. Результат представлен на рисунке 41.

```

Enter operation number: 5
Enter a variable name: new
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib
alling back to check for a `commonName` for now. This feature is being re
es/497 for details.)
    warnings.warn(
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib
alling back to check for a `commonName` for now. This feature is being re
es/497 for details.)
    warnings.warn(
Variable 'new' was deleted

```

Рисунок 41

На данном этапе было выполнено удаление переменной. Операция выполнена успешно, была удалена переменная new. Таким образом, проверка пройдена успешно.

Протестирован функционал сложения переменных. Результат представлен на рисунке 42.

```
Enter operation number: 6
Enter 1st variable name: testOne
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/497 for details.)
warnings.warn(
Enter 2nd variable name: testtwo
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/497 for details.)
warnings.warn(
Enter variable name in which to write the result: testOne
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/497 for details.)
warnings.warn(
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/497 for details.)
warnings.warn(
testOne + testtwo = 44502310.0
```

Рисунок 42

На данном этапе было выполнено сложение значений переменных testOne и testTwo, полученный результат в расшифрованном виде выведен на экран. Таким образом, проверка пройдена успешно.

Протестирован функционал вычитания переменных. Результат представлен на рисунке 43.

```
Enter operation number: 7
Enter 1st variable name: testOne
/Users/olegro/.pyenv/versions/3.10.13/lib/
alling back to check for a `commonName` fo
es/497 for details.)
    warnings.warn(
Enter 2nd variable name: testtwo
/Users/olegro/.pyenv/versions/3.10.13/lib/
alling back to check for a `commonName` fo
es/497 for details.)
    warnings.warn(
Enter variable name in which to write the
/Users/olegro/.pyenv/versions/3.10.13/lib/
alling back to check for a `commonName` fo
es/497 for details.)
    warnings.warn(
/Users/olegro/.pyenv/versions/3.10.13/lib/
alling back to check for a `commonName` fo
es/497 for details.)
    warnings.warn(
testOne - testtwo = 33356684.0
```

Рисунок 43

На данном этапе было выполнено вычитание значения переменной testOne из значения переменной testTwo, полученный результат в расшифрованном виде выведен на экран. Таким образом, проверка пройдена успешно.

Протестирован функционал умножения переменных. Результат представлен на рисунке 44.

```
Enter operation number: 8
Enter 1st variable name: testOne
/Users/olegro/.pyenv/versions/3.10.13/lib/p
alling back to check for a `commonName` for
es/497 for details.)
    warnings.warn(
Enter 2nd variable name: testtwo
/Users/olegro/.pyenv/versions/3.10.13/lib/p
alling back to check for a `commonName` for
es/497 for details.)
    warnings.warn(
Enter variable name in which to write the r
/Users/olegro/.pyenv/versions/3.10.13/lib/p
alling back to check for a `commonName` for
es/497 for details.)
    warnings.warn(
/Users/olegro/.pyenv/versions/3.10.13/lib/p
alling back to check for a `commonName` for
es/497 for details.)
    warnings.warn(
testOne * testtwo = 216946819090592.0
```

Рисунок 44

На данном этапе было выполнено умножение значения переменной testOne на значение переменной testTwo, полученный результат в расшифрованном виде выведен на экран. Таким образом, проверка пройдена успешно.

Протестирован функционал деления переменных. Результат представлен на рисунке 45.

```

Enter operation number: 9
Enter 1st variable name: testOne
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning: Calling back to check for a `commonName` for now. This feature is being removed by major browsers and deprecated in 497 for details.)
  warnings.warn(
Enter 2nd variable name: testtwo
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning: Calling back to check for a `commonName` for now. This feature is being removed by major browsers and deprecated in 497 for details.)
  warnings.warn(
Enter variable name in which to write the result: result
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning: Calling back to check for a `commonName` for now. This feature is being removed by major browsers and deprecated in 497 for details.)
  warnings.warn(
/Users/olegro/.pyenv/versions/3.10.13/lib/python3.10/site-packages/urllib3/connection.py:463: SubjectAltNameWarning: Calling back to check for a `commonName` for now. This feature is being removed by major browsers and deprecated in 497 for details.)
  warnings.warn(
testOne / testtwo = 6.9856093932
Attention! Cryptosystem doesn't allow to encrypt floats. So calculated value will be stored as rounded: (7.0)

```

Рисунок 45

На данном этапе было выполнено деление значения переменной testOne на значение переменной testTwo, полученный результат в расшифрованном виде выведен на экран. Таким образом, проверка пройдена успешно.

Протестирован функционал получения открытого ключа. Результат представлен на рисунке 46.

```

'0' -> Sign out
'1' -> Initialize new variable
'2' -> Get variable list
'3' -> Get variable
'4' -> Edit value
'5' -> Delete variable

'6' -> a + b
'7' -> a - b
'8' -> a * b
'9' -> a / b
'10' -> Get private key (root)
'11' -> Get public key (polynomial)

'12' -> Update private and public keys

Enter operation number: 10
-0.5263157894736842

```

Рисунок 46

На данном этапе был получен открытый ключ пользователя, таким образом, проверка выполнена успешно.

Протестирован функционал получения закрытого ключа. Результат представлен на рисунке 47.

```

'0' -> Sign out
'1' -> Initialize new variable
'2' -> Get variable list
'3' -> Get variable
'4' -> Edit value
'5' -> Delete variable
'6' -> a + b
'7' -> a - b
'8' -> a * b
'9' -> a / b
'10' -> Get private key (root)
'11' -> Get public key (polynomial)
'12' -> Update private and public keys

Enter operation number: 11
      57      56      55      54      31      30      29
1064 x + 560 x + 9576 x + 5040 x + 1330 x + 700 x + 1.273e+04 x
      28      27      26      3      2
+ 6700 x + 6840 x + 3600 x + 950 x + 500 x + 8550 x + 4505

```

Рисунок 47

На данном этапе был получен закрытый ключ пользователя, таким образом, проверка выполнена успешно.

В данном этапе было проведено функциональное тестирование с целью проверки соблюдения функциональных требований к программному комплексу. В результате тестирования сделан вывод, что требования к программному комплексу были удовлетворены в полном объеме.

5 Вредные психофизические факторы, влияющие на нервную систему специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

В современном информационном обществе, где обработка и защита цифровых данных играют ключевую роль, специалисты по защите информации сталкиваются с множеством задач и вызовов. Они обязаны не только обеспечивать безопасность информации, но и подвергаются воздействию различных факторов, которые могут оказать негативное влияние на их нервную систему и общее физическое состояние [8].

Работа в области информационной безопасности требует высокой концентрации внимания, постоянного мониторинга и быстрого принятия решений. Специалисты по защите информации часто сталкиваются с непрерывным стрессом, переутомлением, долгими рабочими часами и высокими требованиями к результатам. Эти факторы могут негативно сказываться на их физическом и психическом здоровье, что, в свою очередь, может привести к снижению производительности труда и качества выполняемой работы.

Вредные психофизические факторы могут оказать серьезное воздействие на нервную систему специалистов по защите информации. Постоянный стресс и тревожность могут привести к негативным эмоциональным состояниям, таким как раздражительность, усталость и бессонница. Длительные периоды монотонной работы могут вызвать чувство одиночества и утомления [9]. Все это может сказаться на эффективности работы и общем физическом и психическом состоянии специалистов по защите информации.

Кроме того, малоподвижный образ жизни, связанный с длительным пребыванием в неподвижном состоянии, может способствовать развитию проблем, связанных с циркуляцией крови в организме и ухудшением общей физической формы. Это может привести к ощущению общей усталости, снижению энергии и ухудшению работоспособности специалиста. Еще одними не менее значимыми негативными последствиями, связанными с

малоподвижной деятельностью специалиста по безопасности, являются возможные проблемы с нарушением правильной циркуляции крови в организме человека, а также общее снижение физической формы.

Физическая активность играет важную роль в поддержании физического и психического здоровья специалистов по защите информации. Регулярные физические упражнения помогают укрепить иммунную систему, улучшить кровообращение и повысить общую выносливость. Рекомендуется включить в ежедневный режим зарядку, прогулки на свежем воздухе и другие виды физической активности.

Для достижения максимальной эффективности следует придерживаться ряда рекомендаций по применению физической культуры:

1) регулярно выполнять физические упражнения, включая разминку и растяжку. Эти упражнения помогут расслабить напряженные мышцы и улучшить общее физическое состояние;

2) поддерживать активный образ жизни, в том числе прогулками на свежем воздухе. Прогулки помогут не только физически, но и психологически расслабиться, освежить ум и улучшить настроение;

3) соблюдать режим сна и отдыха для восстановления нервной системы. Регулярный и качественный сон играет важную роль в восстановлении энергии и снижении стресса;

4) питаться сбалансированно и учитывать потребности организма в витаминах и минералах. Здоровое питание помогает поддерживать физическое и психическое здоровье на должном уровне.

Исследование вредных психофизических факторов, воздействующих на нервную систему специалистов по защите информации, и разработка методов и средств физической культуры для снижения их воздействия имеют большое значение. Психологическая поддержка, регулярная физическая активность и внимание к общему физическому и психическому здоровью помогут

специалистам эффективно справляться с нагрузкой и поддерживать высокую работоспособность.

Более того, стоит подчеркнуть, что забота о физическом и психическом благополучии специалистов по защите информации не только улучшает их качество работы, но и содействует снижению вероятности возникновения болезней, связанных с долгосрочным воздействием вредных факторов.

Кроме того, важно отметить, что физическая активность способствует повышению общего уровня энергии и мотивации, что в свою очередь способствует более эффективному решению сложных задач, которые ставятся перед специалистами по информационной безопасности.

Таким образом, соблюдение рекомендаций по физической культуре и забота о физическом и психическом здоровье становятся неотъемлемой частью успешной карьеры специалистов по защите информации в современном мире. Они позволяют поддерживать стабильную работоспособность и повышать уровень безопасности в цифровой среде.

Заключение

В данной выпускной квалификационной работе была разработан и реализован прототип программного комплекса по скрытию информации при помощи гомоморфного шифрования.

Была рассмотрена значимость сервисов, предоставляющих ресурсы пользователям в качестве вычислительных мощностей. Была сформулирована проблема защиты данных при в подобных системах. Была сформулирована модель нарушителя “извне” и показано, что при такой задаче шифровать данные на стороне сервиса не будет решение. Был представлен и подробно рассмотрен инструмент, обеспечивающий обработку зашифрованных данных. Описана проблематика и поставлены задачи для выполнения в данной выпускной квалификационной работе.

В ходе разработки было подробно изучено гомоморфное шифрование, его математическая формулировка, типы гомоморфных систем и области его применения. Также было рассмотрено его применение в целях реализации набора операций, заявленных в техническом задании.

Исходя из изученной информации была спроектирована общая архитектура программного комплекса, включающая в себя клиентскую часть, серверную часть, базу данных и криптографический модуль. Также были выбраны протоколы для передачи информации и спроектирован криптографический модуль.

В соответствии с проведенным проектированием произведен выбор программных средств и разработан программный комплекс: написана клиентская библиотека, разработана серверная часть, спроектирована база данных. Также разработан криптографический модуль, обеспечивающий полностью гомоморфное шифрование и реализована демонстрационная программа.

При проектировании защищенной системы обработки данных были детально описаны все компоненты и модули программного комплекса.

Подробно описан процесс разработки архитектуры, приведена структура, описание и значимость каждого модуля системы.

В последнем разделе данной выпускной квалификационной приведен и обоснован выбор тех или иных технических решений реализации. Приведен и разобран код реализации системы: клиентской и серверной частей. Приведены шаги инструкции по установке.

Помимо этого, в работе были рассмотрены вредные психофизические факторы, влияющие на нервную систему специалиста по защите информации. Были приведены методы и средства физической культуры, снижающие их воздействие.

Таким образом, все пункты технического задания ВКР выполнены в полном объеме.

Список использованных источников

1. International Data Group (Foundry). Исследование. [Электронный ресурс] URL: <https://foundryco.com/tools-for-marketers/research-cloud-computing/> (дата обращения: 21.01.23).
2. Arif Mohamed. A history of cloud computing Computer Weekly. [Электронный ресурс] URL: <https://www.computerweekly.com/feature/Ahistory-of-cloud-computing> (дата обращения: 21.01.23).
3. Craig Gentry. A fully homomorphic encryption scheme // PhD thesis. – Stanford University. – 2009. – [Электронный ресурс] URL: <http://crypto.stanford.edu/craig> (дата обращения: 21.01.23).
4. RFC 7414 – A Roadmap for TCP Specification Documents. [Электронный ресурс] URL: <https://datatracker.> (дата обращения: 21.01.23).
5. Numpy docs. [Электронный ресурс] URL: <https://numpy.org/doc/> (дата обращения: 21.01.23).
6. Psycopg2 docs. [Электронный ресурс] URL: <https://www.psycopg.org/docs/> (дата обращения: 21.01.23).
7. Flask documentation [Электронный ресурс] URL: <https://flask.palletsprojects.com/en/2.1.x/> (дата обращения: 21.01.23).
8. Вредные и опасные факторы при работе с компьютером. URL: <http://www.grandars.ru/shkola/bezopasnost-zhiznedeyatelnosti/vrednye-factoryprirabote-na-pk.html/> (дата обращения: 21.01.23).
9. Иванов П. С. Стресс и его влияние на работу специалистов в области информационной безопасности. — Конференция по информационной безопасности, 2018. — С. 45-56.

Приложение А

(обязательное)

Исходный код криптографического модуля

```
from typing import Tuple
import secrets
import numpy as np
from .enc_num import EncryptedNumber
from .utils import generate_obfuscating_multiplier
from .vars import *

class AbramovPrivateKey:
    def __init__(self, root: float):
        self._root = root

    def decrypt(self, encrypted_number: EncryptedNumber) -> float:
        decrypted_number = np.polyval(encrypted_number.polynomial,
self._root)
        return float(decrypted_number)

    def get_root(self):
        return self._root

class AbramovPublicKey:
    def __init__(self, base: int, key_polynomial: np.poly1d):
        self._base = base
        self._key_polynomial = key_polynomial

    def encode(self, number: int) -> np.poly1d:
        in_new_numbering_system = np.base_repr(number, self._base)
```

```

        polynomial_representation = np.poly1d([int(digit) for digit in
str(in_new_numbering_system)])

        return polynomial_representation

    def encrypt(self, number: int) -> EncryptedNumber:
        encoded_number = self.encode(int(number))
        encrypted_number_to_wrap = np.polyval(encoded_number,
self._key_polynomial)
        encrypted_number = EncryptedNumber(encrypted_number_to_wrap)
        return encrypted_number

    def get_base(self):
        return self._base

    def get_polynomial(self):
        return self._key_polynomial

    def generate_abramov_keypair(base: int, polynomial_degree: int) ->
Tuple[AbramovPrivateKey, AbramovPublicKey]:
        coef_at_first_deg = secrets.randbelow(MAX_COEFFICIENT_VALUE) *
secrets.randbelow(MAX_COEFFICIENT_VALUE) + 1
        coef_at_zero_deg = secrets.randbelow(MAX_COEFFICIENT_VALUE) *
secrets.randbelow(MAX_COEFFICIENT_VALUE) + 1
        key_polynomial = np.poly1d([coef_at_first_deg, coef_at_zero_deg])

        if polynomial_degree % 2 == 0:
            degree = polynomial_degree
        else:
            degree = polynomial_degree - 1

```

```

while degree != 0:
    tmp_degree = secrets.randbelow(int(degree / 2)) + 1
    obfuscating_polynomial = generate_obfuscating_multiplier(2 *
tmp_degree)
    key_polynomial = key_polynomial * obfuscating_polynomial
    degree -= tmp_degree * 2

key_polynomial += base
root = - coef_at_zero_deg / coef_at_first_deg
private_key = AbramovPrivateKey(root)
public_key = AbramovPublicKey(base, key_polynomial)
return private_key, public_key

```

Приложение Б
(обязательное)
Исходный код сервера

```
import sys

from flask import Flask, jsonify, request, Response
from serverside.request_handler import DBConn
from serverside.server_utils import parse_db_creds


app = Flask(__name__)
PORT = 8282


@app.route("/create_user", methods=["POST"])
def create_user():
    data = request.json
    db.create_user(data=data.values())
    return Response(status=200)


@app.route("/get_user_info", methods=["GET"])
def get_user_info():
    login: str = request.args.get("login")
    response = db.get_user_info(login=login)
    return jsonify(response)


@app.route("/check_variable_exist", methods=["GET"])
def check_variable_exist():
    login: str = request.args.get("login")
    var_name: str = request.args.get("var_name")
    response = db.check_variable_exist(login=login, name=var_name)
    return jsonify(response)
```

```
@app.route("/get_variable", methods=["GET"])
def get_variable():
    login: str = request.args.get("login")
    var_name: str = request.args.get("var_name")
    response = db.get_variable(login=login, name=var_name)
    return jsonify(response)
```

```
@app.route("/get_memory", methods=["GET"])
def get_memory():
    login: str = request.args.get("login")
    response = db.get_memory(login=login)
    return jsonify(response)
```

```
@app.route("/init_new_variable", methods=["POST"])
def init_new_variable():
    data = request.json
    db.init_new_variable(var_data=data.values())
    return Response(status=200)
```

```
@app.route("/edit_value", methods=["POST"])
def edit_value():
    data = request.json
    login = data["login"]
    name = data["var_name"]
    value = data["value"]
    db.edit_value(login=login, var_name=name, value=value)
    return Response(status=200)
```

```

@app.route("/delete_variable", methods=["GET"])
def delete_variable():
    login: str = request.args.get("login")
    var_name: str = request.args.get("var_name")
    db.delete_variable(login=login, var_name=var_name)

@app.route("/calc_variables", methods=["POST"])
def calc_variables():
    data = request.json
    login = data["login"]
    var_name1 = data["var_name1"]
    var_name2 = data["var_name2"]
    var_name_res = data["var_name_res"]
    op = data["op"]
    response = db.calc_variables(login=login, var_name1=var_name1,
var_name2=var_name2, var_name_res=var_name_res, op=op)
    return jsonify(response)

@app.route("/div_variables", methods=["POST"])
def div_variables():
    data = request.json
    login = data["login"]
    var_name1 = data["var_name1"]
    var_name2 = data["var_name2"]
    response = db.div_variables(login=login, var_name1=var_name1,
var_name2=var_name2)
    return jsonify(response)

if __name__ == "__main__":

```

```
db_creds = parse_db_creds(sys.argv[1:])
db = DBConn(*db_creds)
context = ('./serverside/openssl_certs_example/cert.pem',
'./serverside/openssl_certs_example/key.pem')
app.run(host='localhost', port=PORT, debug=False, ssl_context=context)
db.close_connection()
```

Приложение В

(обязательное)

Исходный код клиентской библиотеки

```
import numpy as np

from homomorphic_polynomial_system import keygen

from homomorphic_polynomial_system.utils import ROUND_TO_INT,
ROUND_TO_REAL_NUMBERS

from homomorphic_polynomial_system.enc_num import serialize, deserialize
from .client_utils import *

class ClientConn:

    def __init__(self, ip: str, port: str):
        self.address = f"https://{ip}:{port}"
        self.login = None
        self.private_key = None
        self.public_key = None

    def sign_up(self):
        print(f"{Fore.CYAN}Signing up:{Style.RESET_ALL} "
              f"name{Fore.LIGHTBLACK_EX} -> login ->
password{Style.RESET_ALL}")
        name = input("Enter name: ")
        print(f"{Fore.CYAN}Signing up:{Fore.LIGHTBLACK_EX} "
              f"name -> {Style.RESET_ALL}login{Fore.LIGHTBLACK_EX} ->
password{Style.RESET_ALL}")
        login = signup_login(self.address)
        print(f"{Fore.CYAN}Signing up:{Fore.LIGHTBLACK_EX} "
              f"name -> login -> {Style.RESET_ALL}password")
        password = signup_password()
```

```

        print("To use remote secure memory you need private and public keys.")
        print("Both will be generated automatically after you enter the
parameters:")

        base, degree = enter_alg_params()
        self.private_key, self.public_key =
keygen.generate_abramov_keypair(base, degree)

        user_data = {"login": login, "name": name, "passwd_hash":
md5(password.encode('utf-8')).hexdigest()}
        requests.post(f"{self.address}/create_user", json=user_data,
verify='./clientside/openssl_cert/cert.pem')

        with open('./clientside/keys/private_key.fhe', 'wb') as file_private_key, \
            open('./clientside/keys/public_key.fhe', 'wb') as file_public_key:
            file_private_key.write(pickle.dumps(self.private_key))
            file_public_key.write(pickle.dumps(self.public_key))

        print(f"{Fore.GREEN}You have successfully
registered{Style.RESET_ALL}")
        print(f"{Fore.GREEN}Private and public keys were saved to 'keys'
directory{Style.RESET_ALL}")

    def sign_in(self):
        login, resp = response_login(self.address)
        correct_password = resp[0][PASSWORD_HASH]
        signin_password(correct_password)
        self.login = login

```

```

        print(f"{Fore.GREEN}You have successfully
authorized{Style.RESET_ALL}")

        self.private_key = enter_key(key_type='private')
        self.public_key = enter_key(key_type='public')
        print(f"{Fore.GREEN}Keys have been added{Style.RESET_ALL}")

    def sign_out(self):
        self.login = None
        self.private_key = None
        self.public_key = None

    def init_new_variable(self):
        var_name = str(input("Enter a variable name: "))
        exist =

requests.get(f"{self.address}/check_variable_exist?login={self.login}&var_name={v
ar_name}", verify='./clientside/openssl_cert/cert.pem')

        if len(exist.json()) != 0:
            print(f"{Fore.RED}Value '{var_name}' already
initialized{Style.RESET_ALL}")
            return

        enc_value, value = enter_value(self.public_key)
        serialized_oj = serialize(enc_value)
        variable = {"var_name": var_name, "value": serialized_oj, "login":
self.login}
        requests.post(f"{self.address}/init_new_variable", json=variable,
verify='./clientside/openssl_cert/cert.pem')

```

```

        print(f"{Fore.GREEN}Initialized: {var_name} =
{value}}{Style.RESET_ALL}")

    def get_memory(self):
        response = requests.get(f"{self.address}/get_memory?login={self.login}",
verify='./clientside/openssl_cert/cert.pem')
        memory_dump = response.json()

        if len(memory_dump) != 0:
            for idx, variable in enumerate(memory_dump):
                var_name = variable[NAME]
                serialized_obj = variable[VALUE]
                enc_value = deserialize(serialized_obj)
                value = self.private_key.decrypt(enc_value)
                print(f"{Fore.GREEN}#{idx + 1}: {var_name} =
{value}}{Style.RESET_ALL}")
            else:
                print(f"{Fore.GREEN}No initialized variables{Style.RESET_ALL}")

    def get_variable(self):
        var_name = str(input("Enter a variable name: "))
        response =
requests.get(f"{self.address}/get_variable?login={self.login}&var_name={var_name
}", verify='./clientside/openssl_cert/cert.pem')

        if len(response.json()) != 0:
            serialized_obj = response.json()[0][VALUE]
            enc_value = deserialize(serialized_obj)
            value = self.private_key.decrypt(enc_value)

```

```

        print(f"{Fore.GREEN}{var_name} = {value}{Style.RESET_ALL}")
    else:
        print(f"{Fore.RED}No variable with name
'{var_name}'{Style.RESET_ALL}")

    def edit_value(self):
        var_name = str(input("Enter a variable name: "))
        exist =
requests.get(f"{self.address}/check_variable_exist?login={self.login}&var_name={v
ar_name}", verify='./clientside/openssl_cert/cert.pem')

        if len(exist.json()) == 0:
            print(f"{Fore.RED}No variable with name
'{var_name}'{Style.RESET_ALL}")
            return

        enc_value, value = enter_value(self.public_key)
        serialized_oj = serialize(enc_value)
        variable = {"var_name": var_name, "value": serialized_oj, "login":
self.login}
        requests.post(f"{self.address}/edit_value", json=variable,
verify='./clientside/openssl_cert/cert.pem')
        print(f"{Fore.GREEN}Edited: {var_name} =
{value}{Style.RESET_ALL}")

    def delete_variable(self):
        var_name = str(input("Enter a variable name: "))

```

```

        exist =
requests.get(f"{self.address}/check_variable_exist?login={self.login}&var_name={v
ar_name}", verify='./clientside/openssl_cert/cert.pem')

```

```

        if len(exist.json()) == 0:
            print(f"{Fore.RED}No variable with name
'{var_name}'{Style.RESET_ALL}")
            return

```

```

requests.get(f"{self.address}/delete_variable?login={self.login}&var_name={var_na
me}", verify='./clientside/openssl_cert/cert.pem')

```

```

            print(f"{Fore.GREEN}Variable '{var_name}' was
deleted{Style.RESET_ALL}")

```

```

def calc_variables(self, op):
    var_name1 = str(input("Enter 1st variable name: "))
    exist =
requests.get(f"{self.address}/check_variable_exist?login={self.login}&var_name={v
ar_name1}", verify='./clientside/openssl_cert/cert.pem')

```

```

        if len(exist.json()) == 0:
            print(f"{Fore.RED}No variable with name
'{var_name1}'{Style.RESET_ALL}")
            return

```

```

    var_name2 = str(input("Enter 2nd variable name: "))

```

```

        exist =
requests.get(f"{self.address}/check_variable_exist?login={self.login}&var_name={v
ar_name2}", verify='./clientside/openssl_cert/cert.pem')

```

```

        if len(exist.json()) == 0:
            print(f"{Fore.RED}No variable with name
'{var_name2}'{Style.RESET_ALL}")
            return

```

```

        var_name_res = str(input("Enter variable name in which to write the
result: "))

```

```

        exist =
requests.get(f"{self.address}/check_variable_exist?login={self.login}&var_name={v
ar_name_res}", verify='./clientside/openssl_cert/cert.pem')

```

```

        if len(exist.json()) == 0:
            print(f"{Fore.RED}No variable with name
'{var_name_res}'{Style.RESET_ALL}")
            return

```

```

        if op == "+":
            calc_data = {"login": self.login, "var_name1": var_name1,
"var_name2": var_name2, "var_name_res": var_name_res, "op": "+"}
            response = requests.post(f"{self.address}/calc_variables",
json=calc_data, verify='./clientside/openssl_cert/cert.pem')
            serialized_obj = response.json()
            enc_result = deserialize(serialized_obj)
            result = self.private_key.decrypt(enc_result)
            rounded_result = np.round(result, ROUND_TO_INT)

```

```

        print(f"{Fore.GREEN}{var_name1} + {var_name2} =
{rounded_result}{Style.RESET_ALL}")

    if op == "-":
        calc_data = {"login": self.login, "var_name1": var_name1,
"var_name2": var_name2, "var_name_res": var_name_res, "op": "-"}
        response = requests.post(f"{self.address}/calc_variables",
json=calc_data, verify='./clientside/openssl_cert/cert.pem')
        serialized_obj = response.json()
        enc_result = deserialize(serialized_obj)
        result = self.private_key.decrypt(enc_result)
        rounded_result = np.round(result, ROUND_TO_INT)
        print(f"{Fore.GREEN}{var_name1} - {var_name2} =
{rounded_result}{Style.RESET_ALL}")

    if op == "*":
        calc_data = {"login": self.login, "var_name1": var_name1,
"var_name2": var_name2, "var_name_res": var_name_res, "op": "*"}
        response = requests.post(f"{self.address}/calc_variables",
json=calc_data, verify='./clientside/openssl_cert/cert.pem')
        serialized_obj = response.json()
        enc_result = deserialize(serialized_obj)
        result = self.private_key.decrypt(enc_result)
        rounded_result = np.round(result, ROUND_TO_INT)
        print(f"{Fore.GREEN}{var_name1} * {var_name2} =
{rounded_result}{Style.RESET_ALL}")

```

DIVISION IS DIFFERENT FROM OTHER OPERATIONS

```

if op == "/":

```

```

try:
    calc_data = {"login": self.login, "var_name1": var_name1,
"var_name2": var_name2}
    response = requests.post(f"{self.address}/div_variables",
json=calc_data, verify='./clientside/openssl_cert/cert.pem')
    response_result = response.json()
    enc_whole_part, enc_remains, enc_divider = \
        deserialize(response_result[WHOLE_PART]), \
        deserialize(response_result[REMAINS]), \
        deserialize(response_result[DIVIDER])
    whole_part = self.private_key.decrypt(enc_whole_part)
    remains = self.private_key.decrypt(enc_remains)
    divider = self.private_key.decrypt(enc_divider)
    result = whole_part + remains / divider
    rounded_result = np.round(result, ROUND_TO_REAL_NUMBERS)
    print(f"{Fore.GREEN}{var_name1} / {var_name2} =
{rounded_result} {Style.RESET_ALL}")
    rounded_result_to_int = np.round(result, ROUND_TO_INT)
    print(f"{Fore.RED}Attention! Cryptosystem doesn't allow to encrypt
floats. "
        f"So calculated value will be stored as rounded:
({rounded_result_to_int}) {Style.RESET_ALL}")
    enc_rounded_result = self.public_key.encrypt(rounded_result_to_int)
    serialized_obj = serialize(enc_rounded_result)
    variable = {"var_name": var_name_res, "value": serialized_obj,
"login": self.login}
    requests.post(f"{self.address}/edit_value", json=variable,
verify='./clientside/openssl_cert/cert.pem')
except ZeroDivisionError:

```

```

        print(f"{Fore.RED}Division by zero{Style.RESET_ALL}")

def get_key(self, key):
    if key == "private":
        print(self.private_key.get_root())
    if key == "public":
        print(self.public_key.get_polynomial())

def update_keys(self):
    pass
    decision = input(f"{Fore.RED}Updating the keys will lead to temporarily
downloading the entire "
                    f"remote memory on the local host to perform the re-encryption
operation with new keys."
                    f" Do you want to continue? [y/n];{Style.RESET_ALL}")

    if decision == 'n' or decision == 'no':
        return

    base, degree = enter_alg_params()
    new_private_key, new_public_key =
keygen.generate_abramov_keypair(base, degree)
    response = requests.get(f"{self.address}/get_memory?login={self.login}",
verify='./clientside/openssl_cert/cert.pem')
    memory_dump = response.json()

    if len(memory_dump) != 0:
        for idx, variable in enumerate(memory_dump):
            var_name = variable[NAME]

```

```

        serialized_obj = variable[VALUE]
        enc_value = deserialize(serialized_obj)
        value = self.private_key.decrypt(enc_value)
        new_enc_value = new_public_key.encrypt(value)
        serialized_obj = serialize(new_enc_value)
        variable = {"var_name": var_name, "value": serialized_obj, "login":
self.login}

        requests.post(f"{self.address}/edit_value", json=variable,
verify='./clientside/openssl_cert/cert.pem')

self.private_key, self.public_key = new_private_key, new_public_key

with open('./clientside/keys/private_key.fhe', 'wb') as file_private_key, \
    open('./clientside/keys/public_key.fhe', 'wb') as file_public_key:
    file_private_key.write(pickle.dumps(self.private_key))
    file_public_key.write(pickle.dumps(self.public_key))

print(f"{Fore.GREEN}Private and public keys were updated "
      f"and saved to 'keys' directory{Style.RESET_ALL}")

```

Приложение Д

Исходный код демонстрационной программы

```
import sys

from clientside.client_utils import parse_server_info, enter_option, print_error
from clientside.client import ClientConn

def main(argv):
    ip, port = parse_server_info(argv)
    _client = ClientConn(ip, port)

    while True:
        print("'0' -> Exit")
        print("'1' -> Sign in")
        print("'2' -> Sign up")

        sign_choice = enter_option("Enter task number: ")

        match sign_choice:
            case 0:
                del _client
                return
            case 1:
                _client.sign_in()
                while True:
                    print()
                    print("'0' -> Sign out")
                    print("'6' -> a + b")
                    print("'1' -> Initialize new variable")
                    print("'7' -> a - b")
```

```

print("""2' -> Get variable list")
print("""8' -> a * b")
print("""3' -> Get variable")
print("""9' -> a / b")
print("""4' -> Edit value")
print("""10' -> Get private key (root)")
print("""5' -> Delete variable")
print("""11' -> Get public key (polynomial)")
print()
print("""12' -> Update private and public keys")
print()

```

```

opt_choice = enter_option("Enter operation number: ")

```

```

match opt_choice:
    case 0:
        _client.sign_out()
        break
    case 1:
        _client.init_new_variable()
    case 2:
        _client.get_memory()
    case 3:
        _client.get_variable()
    case 4:
        _client.edit_value()
    case 5:
        _client.delete_variable()
    case 6:

```

```

        _client.calc_variables("+")
    case 7:
        _client.calc_variables("-")
    case 8:
        _client.calc_variables("*")
    case 9:
        _client.calc_variables("/")
    case 10:
        _client.get_key("private")
    case 11:
        _client.get_key("public")
    case 12:
        _client.update_keys()
    case _:
        print_error("Undefined option")

case 2:
    _client.sign_up()

case _:
    print_error("Invalid action")

if __name__ == "__main__":
    main(sys.argv[1:])

```