МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Анализ шифрования данных для хранения

АВТОРЕФЕРАТ
дипломной работы

студента 6 курса 631 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Мязина Александра Валерьевича

Научный руководитель
доцент к. ф.-м. н. _____ А. В. Жаркова
22.01.2024 г.

Заведующий кафедрой
д. ф.-м. н., доцент _____ М. Б. Абросимов
22.01.2024 г.

Саратов 2024

ВВЕДЕНИЕ

В наше время, когда цифровые технологии становятся неотъемлемой частью повседневной деятельности, объем информации, создаваемой и обрабатываемой организациями, стремительно увеличивается. Этот беспрецедентный рост информационных потоков ставит перед предприятиями не только задачу эффективного управления данными, но и обеспечения их полной безопасности.

Проблема утечек информации и нарушений конфиденциальности стала более острой в условиях цифровой трансформации. В этом контексте статистика, утверждающая, что большинство компаний подвергаются риску банкротства из-за утраты всего лишь небольшой части своей корпоративной информации, выглядит более чем настораживающей. Таким образом, стоит признать, что информация – не только ценный ресурс, но и объект повышенного внимания злоумышленников, готовых использовать любую возможность для ее несанкционированного доступа[1].

В условиях такой информационной угрозы становится ясным, что обеспечение безопасности данных – это не просто мера предосторожности, но и стратегическая необходимость. Многие компании переходят к электронному хранению данных, что, несомненно, предоставляет им множество преимуществ, но при этом усиливает проблемы, связанные с конфиденциальностью.

В этом контексте технология шифрования данных становится неотъемлемым элементом стратегии обеспечения безопасности. Шифрование, как метод искажения информации посредством специальных алгоритмов, имеет богатую историю использования для защиты конфиденциальности. И хотя существуют современные средства защиты, вроде межсетевых экранов

[1] 1 Чмора, А. Л. Современная прикладная криптография [Электронный ресурс] / А. Л. Чмора. – М. : Гелиос-АРВ, 2002. – 244 с. – Загл. с экрана. – Яз. рус.

антивирусов, шифрование остаётся неоспоримым инструментом для дополнительного слоя обороны[2].

Целью данной дипломной работы является изучение вопросов, связанных с шифрованием данных для хранения, в результате чего требуется написать программный продукт для надёжного хранения данных посредством их шифрования.

В процессе работы необходимо решить следующие задачи:

1) изучить основные положения задачи шифрования данных для хранения;

2) рассмотреть различные блочные и поточные алгоритмы шифрования данных и сравнить их эффективность на практике;

3) написать программный продукт, функционал которого позволит надёжно шифровать файлы пользователей и безопасно хранить их в системе.

Данная работа частично была представлена на студенческой научной конференции СГУ.

Дипломная работа состоит из введения, 5 разделов, заключения, списка использованных источников и 1 приложения. Общий объем работы – 82 страницы, из них 47 страниц – основное содержание, включая 35 рисунков, список использованных источников из 20 наименований.

[2] 2 Хачатурова, С. С. Хранение и защита информации [Электронный ресурс] / С. С. Хачатурова // Международный журнал прикладных и фундаментальных исследований [Электронный ресурс]. – 2016. – № 2-1. – С. 63–65. – URL: https://applied-research.ru/ru/article/view?id=8427 (дата обращения: 25.11.2023). – Загл. с экрана. – Яз. рус.

КРАТКОЕ СОДЕРЖАНИЕ

В дипломной работе в разделе 1 «Необходимые определения» приводятся основная теория по теме работы и понятия, которые в дальнейшем встречаются в тексте.

В разделе 2 работы «Блочное и поточное шифрование» рассматриваются блочные и поточные шифры, схемы их работы, примеры алгоритмов, их назначение, преимущества и недостатки.

Блочные шифры оперируют блоками фиксированной длины открытого текста. В процессе шифрования блочные шифры с использованием одного и того же ключа всегда преобразуют одинаковый блок открытого текста в один и тот же блок шифртекста.

Существует несколько режимов работы блочных шифров, объединяющих базовый шифр, обратную связь и различные операции:

1) режим простой замены (Electronic Codebook, ECB);

2) режим гаммирования (Counter, CTR);

3) режим гаммирования с обратной связью по выходу (Output Feedback, OFB);

4) режим простой замены с зацеплением (Cipher Block Chaining, CBC);

5) режим гаммирования с обратной связью по шифртексту (Cipher Feedback, CFB)[3, 4, 5].

Поточные шифры преобразуют открытый текст в шифртекст не по блокам, а по одному биту (в программных реализациях – по одному байту). В поточном шифре используется генератор гаммы (бегущего ключа), который создаёт последовательность бит $k_1, k_2, \dots, k_n, \dots$, к которым применяется операция XOR с

---

[3] 5 ГОСТ Р 34.13–2015. Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров [Электронный ресурс] // Технический комитет по стандартизации «Криптографическая защита информации» [Электронный ресурс]. – URL: https://tc26.ru/standard/gost/GOST_R_3413-2015.pdf (дата обращения: 25.11.2023). – Загл. с экрана. – Яз. рус.

[4] 8 Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на Си / Б. Шнайер. – М. : Триумф, 2002. – 610 с.

[5] 9 Жаркова, А. В. О сравнении блочных и поточных шифров [Электронный ресурс] / А. В. Жаркова, М. Д. Проданов // Международная научная конференция [Электронный ресурс]. ¬– М. : Научная книга, 2021. – С. 65–68. – Загл. с экрана. – Яз. рус.

битами открытого текста $p_1, p_2, \dots, p_n, \dots$, результатом которой являются биты шифртекста $c_1, c_2, \dots, c_n, \dots$, то есть $c_i = p_i \oplus k_i$, при этом $p_i = c_i \oplus k_i$. В поточных шифрах используют ключи, которые определяют выход генератора гаммы[4, 5].

В дипломной работе в разделе 3 «О шифровании данных для хранения» приводятся схемы шифрования жёсткого диска.

Для шифрования жёсткого диска можно воспользоваться двумя схемами: шифрованием на уровне файлов и шифрованием на уровне драйверов. Шифрование на файловом уровне означает, что все файлы шифруются по отдельности. Для использования зашифрованного файла сначала нужно расшифровать файл, затем использовать его, а потом снова зашифровать.

Шифрование на уровне драйвера обеспечивает защиту для всего логического диска, где все данные шифруются. Этот метод, если реализован правильно, обеспечивает безопасность без необходимости дополнительных действий со стороны пользователя. Однако разработка драйвера требует более сложного подхода, так как он должен управлять установкой драйверов устройств, выделением новых секторов для файлов, утилизацией старых секторов и обработкой других операций, имеющих отношение к логическому диску.

Также в данном разделе описывается порядок взаимодействия файловой системы и жёсткого диска, выделяются уровни, на которых могут осуществляться операции, сопутствующие сохранению данных на диск в операционных системах. Далее рассматриваются некоторые примеры осуществления защиты хранимых данных на разных уровнях взаимодействия. Также в этой части работы уделяется особое внимание универсальности используемого подхода к обеспечению безопасности данных.

Вопрос обеспечения безопасности хранения данных представляет собой ярко выраженную практическую проблему, поэтому при изучении различных методов обеспечения безопасности необходимо уделять особое внимание их практическим характеристикам. Учитывая сложную иерархию и

многокомпонентность системы обработки данных, особенно важной является универсальность подхода, которая достигается тем лучше, чем больше компонентов, для которых модуль безопасности способен обеспечивать защитные свойства, и чем меньше дополнительных шагов (например, изменений в исходном коде), необходимых для достижения этой цели[6].

Кроме того, в этом разделе описываются этапы работы современных FDE-схем и некоторые актуальные аварийные ситуации, которые могут возникнуть во время работы FDE-модуля.

Шифрование томов диска использует уровень драйвера устройства для шифрования и расшифрования информации на физическом носителе. Шифрование томов позволяет зашифровать все логические разделы, что является удобным и интуитивно понятным для конечного пользователя, но не обеспечивает надёжного контроля за доступом к отдельным директориям или файлам. Основной принцип работы заключается в том, что модуль FDE перехватывает запросы от файловой системы, обрабатывает данные и взаимодействует с драйвером диска для обеспечения безопасности и записи данных на диск. Для реализации «прозрачной» защиты данных интерфейс FDE должен быть идентичен интерфейсу драйвера жёсткого диска в операционной системе, в которой он используется. Таким образом, файловая система «не знает» о том, что данные на используемом ею жёстком диске защищены[6].

В разделе 4 работы «Программная реализация» подробно описывается созданная программа, суть её работы.

Программа была написана на языке C# в среде разработки Microsoft Visual Studio 2022. Для хранения учётных данных пользователей была выбрана СУБД Microsoft SQL Server 2022. Для доступа к базе данных применялось стандартное средство подключения данных Visual Studio 2022. В качестве средства для создания интерфейса использовалась платформа Windows Forms. В программе

---

[6] 13 Защищенное хранение данных и полнодисковое шифрование [Электронный ресурс] / Е. К. Алексеев, Л. Р. Ахметзянова, А. А. Бабуева, С. В. Смышляев // Прикладная дискретная математика [Электронный ресурс]. – 2020. – № 49. – С. 78–97 – URL: http://vital.lib.tsu.ru/vital/access/manager/Repository/vtls:000723929 (дата обращения: 28.11.2023). – Загл. с экрана. – Яз. рус.

применяется библиотека BouncyCastle с реализованными в ней алгоритмами шифрования.

В программной реализации для защиты данных пользователю доступны следующие блочные алгоритмы:

1) RC2;

2) Blowfish;

3) Cast6;

4) AES;

5) ГОСТ Р 34.12–2015;

6) Cast5;

7) RC5;

8) SM4.

Среди поточных можно использовать реализации алгоритмов Salsa и ChaCha из BouncyCastle.

Для хранения учётных данных пользователя в зашифрованном виде используется реализация хэш-функции SHA256 из пространства имён System.Security.Cryptography.

В этом разделе рассматриваются такие доступные администратору функции, как управление пользовательскими учётными записями, генерация ключей и векторов инициализации для выдачи пользователям и использование средства для анализа работы алгоритмов шифрования, что выполняется в соответствии с рисунками 4.4, 4.8, 4.10. По умолчанию в программе предусмотрен только один администратор с логином «admin» и паролем «admin», но могут быть созданы новые пользователи с правами администратора.

На управление учётными данными пользователя существуют ограничения: запрещено добавлять пользователей с одинаковыми логинами, удалять можно любых пользователей, кроме пользователя с именем admin, он остаётся в системе при любых условиях.

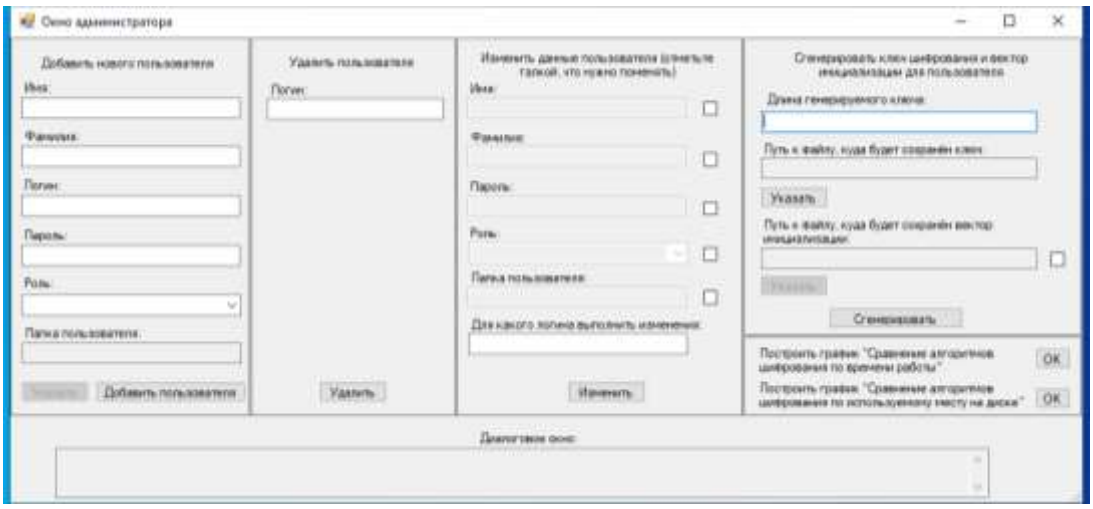

Рисунок 4.4 – Окно администрирования

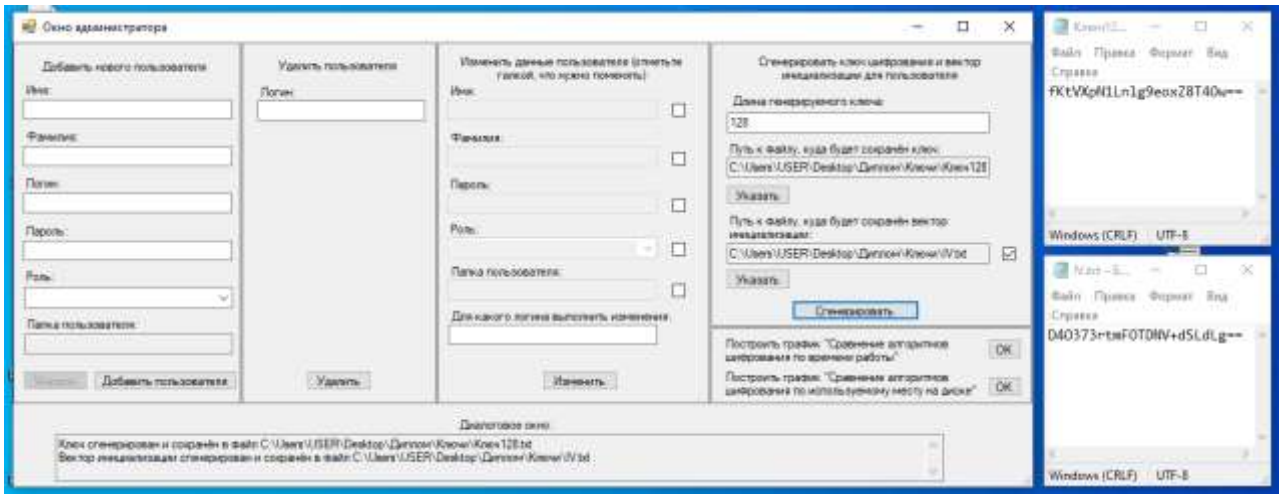

Рисунок 4.8 – Генерация ключа шифрования и вектора инициализации

Рисунок 4.10 – Окно с графиком работы алгоритмов

Для генерации ключа шифрования следующим образом используется библиотека BouncyCastle. Создается новый объект класса CipherKeyGeneration. После его инициализации с использованием в качестве параметров объекта класса KeyGenerationParameters и заданной длины ключа вызывается метод GenerateKey. В результате получаем ключ шифрования выбранной длины. Аналогичным образом генерируется вектор инициализации. Стоит отметить, что ключи необходимо хранить безопасно.

Также в данном разделе указывается, как пользователь может применить программу, чтобы безопасно хранить свои данные в зашифрованном виде, что выполняется в соответствии с рисунком 4.12.

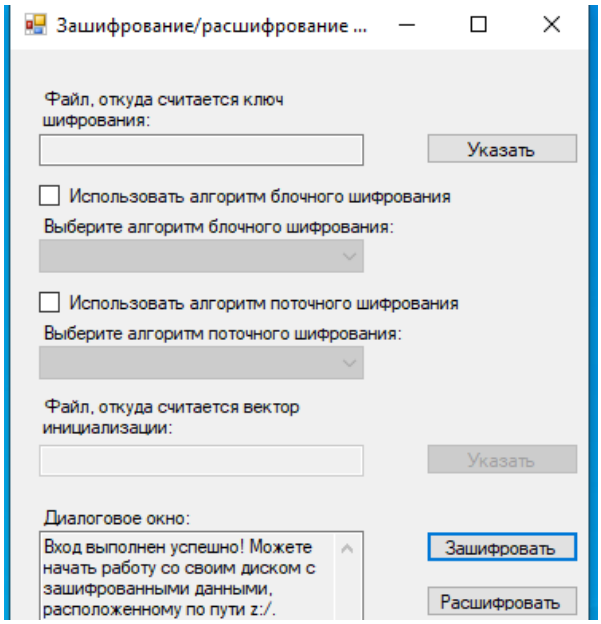

Рисунок 4.12 – Окно для работы пользователя

После указания файла с ключом пользователь должен выбрать, каким алгоритмом будет зашифровывать и расшифровывать свои данные – блочным или поточным. Причём в выпадающих списках доступных алгоритмов шифрования появятся только те из них, которые поддерживают работу с длиной выбранного ключа. Сначала надо поставить отметку рядом с используемым типом алгоритма шифрования, затем нужно выбрать конкретный алгоритм из выпадающего списка. В случае поточного алгоритма шифрования необходимо также указать файл с вектором инициализации.

Когда пользователь будет готов завершить работу, ему нужно будет нажать на кнопку «Зашифровать», чтобы все файлы на логическом диске снова стали защищены с помощью шифрования выбранным алгоритмом. Более того, пользователь может последовательно применить несколько разных алгоритмов шифрования, как поточных, так и блочных.

На момент развёртывания программы доступна только одна учётная запись с логином «admin» и паролем «admin». То есть начало работы программы начинается с того, что администратор создаёт учётные записи остальных пользователей, указывает пути к их папкам с данными. После этого добавленные пользователи смогут работать в программе.

При запуске программы открывается окно авторизации. Когда пользователь заполняет поля «Логин» и «Пароль» и пытается авторизоваться, приложение отправляет запрос к базе данных, где сравниваются введённые логин и пароль в хэшированном виде с теми, что хранятся в базе. При любом неудачном входе в систему пользователь получит сообщение об ошибке.

Во время того, как пользователь работает в программе, монтируется логический диск, который по сути и является папкой с данными пользователя. Любые изменения с файлами на логическом диске будут применены на самом деле к файлам в папке пользователя.

Для монтирования диска написан код, по функционалу аналогичный команде subst в терминале Windows. Команда subst позволяет создать виртуальный диск, содержимым которого, будет заданный в команде каталог файловой системы.

Любые возникающие исключения будут отображаться в специально предназначенном для этого диалоговом окне.

В разделе 5 работы «Результаты работы программы» проведён анализ работы разных алгоритмов, используемых в написанной программной реализации. Чтобы упростить сбор статистики в программе предусмотрено ведение журнала событий. В журнале записывается подробная информация о действиях администратора по изменению учётных данных, а также фиксируется,

какой пользователь с какими параметрами зашифровал свои данные и за какое время.

Также есть возможность на основе журнала событий построить графики работы алгоритмов, в которых показано время их работы и используемое ими место на диске в ходе различных экспериментов. Также возможно масштабировать график по вертикальной оси.

Тестирование программы проводилось на компьютере со следующими характеристиками: процессор – Intel Core i3-12100F, 3300 МГц, оперативная память – 32,0 ГБ, тип накопителя – SSD, видеокарта – GeForce GTX 950.

При анализе полученных данных сделаны следующие выводы:

1) чем больше длина ключа, тем дольше выполняется шифрование/расшифрование данных, что особенно заметно в случае использования AES;

2) поточные алгоритмы шифрования работают быстрее, чем блочные;

3) самыми быстрыми алгоритмами среди представленных в программе стали ChaCha и Salsa, самым медленным оказался RC2;

4) при шифровании поточными алгоритмами данные пользователя не увеличиваются в размере, в то время как файлы, зашифрованные алгоритмами блочного шифрования, занимают на диске больше места.

Насчёт реализованного программного продукта стоит добавить, что пользователи должны надёжно хранить выданные им администратором ключи и никому их не передавать. Иначе доступ к данным пользователя могут получить посторонние лица.

ЗАКЛЮЧЕНИЕ

Подводя итоги, следует отметить, что никто в мире не может полностью гарантировать защиту от кражи и утраты данных, независимо от размера, отрасли и местоположения организации. Несмотря на наличие многочисленных эффективных алгоритмов шифрования, использование которых значительно повышает уровень безопасности, они не исключают возможности несанкционированного доступа к данным. Применение различных методов шифрования для хранения информации, по крайней мере, серьезно усложняет задачу злоумышленникам и, возможно, заставляет их отказаться от своих намерений. Таким образом, основной целью в области защиты информации на сегодняшний день остаётся создание надёжных барьеров, способных максимально затруднить несанкционированный доступ[7].

В ходе данной работы изучены основные положения задачи шифрования данных для хранения, рассмотрены различные алгоритмы шифрования данных, написан программный продукт, функционал которого позволяет надежно шифровать файлы пользователей и безопасно хранить их в системе, причём при запусках программы с различными параметрами выявлена большая эффективность поточных алгоритмов по сравнению с блочными в рамках этой задачи.

Таким образом, все поставленные задачи решены, цель работы достигнута.

[7] 14 Иванов, К. К. Алгоритмы шифрования данных [Электронный ресурс] / К. К. Иванов, Р. Н. Юрченко, А. С. Ярмонов // Молодой ученый [Электронный ресурс]. – 2016. – № 29 (133). – С. 18–20 – URL: https://moluch.ru/archive/133/37180/ (дата обращения: 28.11.2023). – Загл. с экрана. – Яз. рус.