

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра прикладной информатики

ОТЧЕТ О ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

Кафедра прикладной информатики
Шифрование, дешифрование, цифровая подпись файлов C#

Руководитель от университета

подпись, дата

П.П. Дьячук

Студент КИ23-21Б, 032319560

подпись, дата

П.А. Казарез

Красноярск 2025

СОДЕРЖАНИЕ

Введение.....	3
Основная часть	4
1 Описание алгоритмов	4
1.1 RSA.....	4
1.2 Алгоритм MD5	4
2 Спецификация требований к ПО	5
2.1 Общее описание бизнес-процесса.....	5
2.2 Описание прецедентов и их интерфейсы	5
2.2.1 Зашифровать файл	5
2.2.2 Расшифровать файл	6
2.2.3 Создать цифровую подпись	7
2.2.4 Проверить подпись	7
3 Реализация ПО	9
3.1 Настройка среды разработки и подключение библиотек	9
3.1.1 Настройка среды разработки	9
3.1.2 Использование библиотек.....	9
3.1.3 Технологии и подходы	9
3.2 Тестирование ПО	10
3.2.1 Проверка шифрования и дешифрования.....	10
3.2.2 Проверка цифровой подписи.....	11
Заключение	12
Список использованных источников	13

ВВЕДЕНИЕ

Необходимо разработать консольное приложение, которое реализует алгоритм RSA для шифрования и дешифрования файлов различных форматов (DOC, DOCX, PDF, TXT, XLS и других), а также создание и проверку цифровой подписи с использованием алгоритма MD5. Приложение поможет обеспечить базовый уровень защиты данных и сможет быть использовано для безопасного хранения и передачи конфиденциальной информации.

Задачи:

- анализ требований и проектирование системы;
- настройка среды разработки и подключение библиотек;
- реализация шифрования и дешифрования;
- реализация цифровой подписи;
- работа с файловой системой;
- оформить отчет.

Приложение должно позволить безопасно шифровать файлы, проверять их целостность с помощью цифровой подписи и работает с разными форматами данных.

ОСНОВНАЯ ЧАСТЬ

1 Описание алгоритмов

1.1 RSA

RSA — это один из первых и наиболее известных асимметричных криптографических алгоритмов. Он был разработан в 1977 году Рональдом Ривестом, Ади Шамиром и Леонардом Адлеманом и получил своё название по первым буквам их фамилий. Основное отличие RSA от симметричных алгоритмов заключается в использовании двух ключей: открытого и закрытого.

Открытый ключ доступен всем и используется для шифрования данных или проверки цифровой подписи. Закрытый ключ хранится в секрете и применяется для расшифрования данных или создания подписи. Это позволяет обмениваться зашифрованной информацией даже между сторонами, которые ранее не договаривались о ключе.

Основа алгоритма RSA лежит в сложности разложения большого числа на два простых множителя. Генерация ключей начинается с выбора двух больших простых чисел, вычисления их произведения и дальнейшего построения пары ключей. Шифрование и расшифрование данных производятся с использованием операций возведения в степень по модулю.

RSA широко применяется в различных системах безопасности, включая протоколы SSL/TLS, электронную подпись и шифрование сообщений. Однако из-за своей вычислительной сложности RSA обычно не используется для прямого шифрования больших объёмов данных. Чаще всего он применяется для шифрования симметричного ключа, который затем используется для быстрого шифрования основных данных.

1.2 Алгоритм MD5

MD5 — это алгоритм хэширования, разработанный Рональдом Ривестом в 1991 году. Он предназначен для преобразования данных произвольной длины в уникальный 128-битный хэш, который обычно представляется в виде 32-символьной шестнадцатеричной строки. Хэш-функция позволяет эффективно проверять целостность данных, так как даже небольшое изменение исходных данных приводит к значительному изменению результата хэширования.

Процесс работы MD5 включает несколько этапов. Сначала данные дополняются до определённой длины, затем разбиваются на блоки по 512 бит и последовательно обрабатываются с помощью ряда нелинейных операций. В результате формируется итоговый хэш, который может использоваться для проверки целостности файла или как часть цифровой подписи.

Несмотря на широкое распространение, MD5 считается криптографически небезопасным. Были найдены методы, позволяющие создать два различных набора данных с одинаковым хэшем — так называемые коллизии. Поэтому

алгоритм не рекомендуется для использования в задачах, требующих высокой безопасности, таких как хранение паролей или защита данных в финансовых системах.

2 Спецификация требований к ПО

2.1 Общее описание бизнес-процесса

Разрабатываемое программное обеспечение представляет собой консольное приложение, предназначенное для обеспечения базовой криптографической защиты файлов. Основной задачей приложения является предоставление пользователю возможности шифровать и расшифровывать файлы, а также создавать и проверять цифровые подписи. Приложение ориентировано на простоту использования и поддержку различных форматов файлов, включая текстовые документы, таблицы, PDF и другие.

Ниже на рисунке 1 приведена диаграмма прецедентов, описывающая основные функциональные возможности системы.

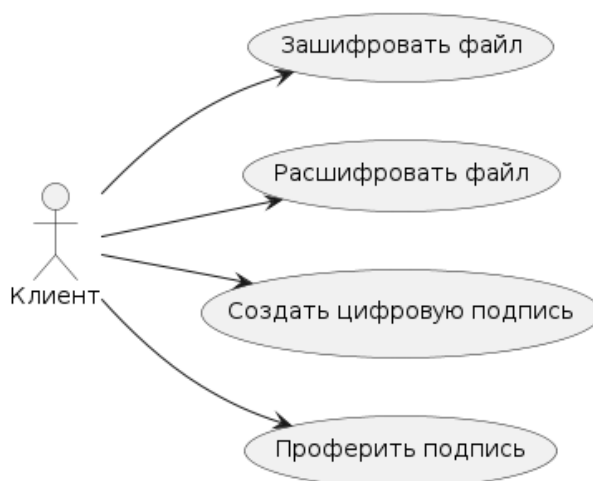


Рисунок 1 – Диаграмма прецедентов

Эти прецеденты полностью охватывают основные задачи, которые решает приложение.

2.2 Описание прецедентов и их интерфейсы

2.2.1 Зашифровать файл

Цель: зашифровать входящий файл и сохранить результат.

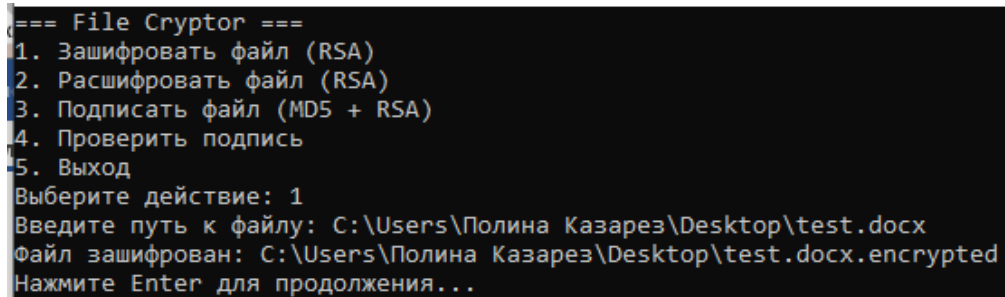
Предусловия: приложение запущено, пользователь выбрал операцию шифрования, файл, который необходимо зашифровать, существует на диске.

Основная последовательность:

1. Пользователь выбирает в меню операцию “Зашифровать файл”.
2. Приложение запрашивает путь к файлу.
3. Пользователь вводит путь к исходному файлу.

4. Приложение читает файл блоками.
5. Каждый блок шифруется с использованием открытого ключа RSA.
6. Зашифрованные блоки записываются в новый файл с расширением .encrypted.
7. Приложение информирует пользователя о завершении информации.

Постусловия: зашифрованный файл сохранен на диске. В случае ошибки приложение информирует пользователя и удаляет частично создан. Ниже на рисунке 2 изображен процесс пользовательского ввода.



```
=== File Cryptor ===
1. Зашифровать файл (RSA)
2. Расшифровать файл (RSA)
3. Подписать файл (MD5 + RSA)
4. Проверить подпись
5. Выход
Выберите действие: 1
Введите путь к файлу: C:\Users\Полина Казарез\Desktop\test.docx
Файл зашифрован: C:\Users\Полина Казарез\Desktop\test.docx.encrypted
Нажмите Enter для продолжения...
```

Рисунок 2 – Интерфейс программы (шифрование)

2.2.2 Расшифровать файл

Цель: восстановить исходное содержимое зашифрованного файла с использованием закрытого ключа.

Предусловия: приложение запущено, пользователь выбрал операцию расшифрования, существует зашифрованный файл, созданный с помощью данного приложения.

Основная последовательность:

1. Пользователь выбирает в меню операцию "Расшифровать файл".
2. Приложение запрашивает путь к зашифрованному файлу.
3. Пользователь вводит путь к файлу с расширением .encrypted.
4. Приложение читает файл блоками.
5. Каждый блок расшифровывается с использованием закрытого ключа RSA.
6. Расшифрованные блоки записываются в новый файл, который получает исходное имя без расширения .encrypted.
7. Приложение информирует пользователя о завершении операции.

Постусловия: расшифрованный файл сохранен на диске, в случае ошибки (например, повреждение файла или неверный ключ) приложение информирует пользователя и удаляет частично созданный файл. Ниже на рисунке 3 изображен процесс пользовательского ввода.

```

=== File Cryptor ===
1. Зашифровать файл (RSA)
2. Расшифровать файл (RSA)
3. Подписать файл (MD5 + RSA)
4. Проверить подпись
5. Выход
Выберите действие: 2
Введите путь к зашифрованному файлу: C:\Users\Полина Казапез\Desktop\test.docx.encrypted
Файл расшифрован: C:\Users\Полина Казапез\Desktop\test.docx

```

Рисунок 2 – Интерфейс программы (дешифрование)

2.2.3 Создать цифровую подпись

Цель: сформировать цифровую подпись для файла, чтобы в дальнейшем можно было проверить его целостность и подлинность.

Предусловия: приложение запущено, пользователь выбрал операцию создания подписи, существует файл, для которого необходимо создать подпись, доступен закрытый ключ RSA.

Основная последовательность:

1. Пользователь выбирает в меню операцию "Создать цифровую подпись".
2. Приложение запрашивает путь к файлу.
3. Пользователь вводит путь к файлу.
4. Приложение вычисляет хэш-сумму файла с использованием алгоритма MD5.
5. Хэш подписывается закрытым ключом RSA.
6. Подпись сохраняется в отдельный файл с расширением .signature.
7. Приложение информирует пользователя о завершении операции.

Постусловия: файл подписи создан и сохранен рядом с оригиналом, в случае ошибки приложение информирует пользователя и не сохраняет частично сформированную подпись.

Ниже на рисунке 3 приведен процесс пользовательского ввода при создании цифровой подписи.

```

=== File Cryptor ===
1. Зашифровать файл (RSA)
2. Расшифровать файл (RSA)
3. Подписать файл (MD5 + RSA)
4. Проверить подпись
5. Выход
Выберите действие: 3
Введите путь к файлу: C:\Users\Полина Казапез\Desktop\test.docx
Введите путь для сохранения (по умолчанию: C:\Users\Полина Казапез\Desktop\test.docx.signature):
Файл успешно сохранён: C:\Users\Полина Казапез\Desktop\test.docx.signature
Подпись создана успешно

```

Рисунок 3 – Интерфейс программы (создание подписи)

2.2.4 Проверить подпись

Цель: убедиться в целостности и подлинности файла, проверив его цифровую подпись.

Предусловия: приложение запущено, пользователь выбрал операцию создания цифровой подписи, существует оригинальный файл и файл его цифровой подписи, доступен открытый ключ RSA.

Основная последовательность:

1. Пользователь выбирает в меню операцию "Проверить подпись".
2. Приложение запрашивает путь к оригинальному файлу.
3. Пользователь вводит путь к файлу.
4. Приложение запрашивает путь к файлу подписи.
5. Пользователь указывает путь к файлу с расширением .signature.
6. Приложение считывает подпись и вычисляет хэш оригинального файла.
7. Выполняется проверка подписи с использованием открытого ключа RSA.
8. Приложение выводит результат проверки: подпись действительна или недействительна.

Постусловия: оригинал проверки и подпись остаются без изменений.

Ниже на рисунке 4 приведен интерфейс программы при проверке валидной цифровой подписи.

```
=== File Cryptor ===
1. Зашифровать файл (RSA)
2. Расшифровать файл (RSA)
3. Подписать файл (MD5 + RSA)
4. Проверить подпись
5. Выход
Выберите действие: 4
Введите путь к файлу: C:\Users\Полина Казарез\Desktop\test.docx
Введите путь к подписи: C:\Users\Полина Казарез\Desktop\test.docx.signature
Подпись валидна
```

Рисунок 4 – Интерфейс программы (проверка валидной подписи)

При изменении файла signature, программа оповещает о недействительности цифровой подписи. На рисунке 5 ниже приведен интерфейс программы при проверке недействительной подписи.

```
=== File Cryptor ===
1. Зашифровать файл (RSA)
2. Расшифровать файл (RSA)
3. Подписать файл (MD5 + RSA)
4. Проверить подпись
5. Выход
Выберите действие: 4
Введите путь к файлу: C:\Users\Полина Казарез\Desktop\test.docx
Введите путь к подписи: C:\Users\Полина Казарез\Desktop\test.docx.signature
Подпись недействительна
```

Рисунок 5 – Интерфейс программы (проверка недействительной подписи)

3 Реализация ПО

3.1 Настройка среды разработки и подключение библиотек

3.1.1 Настройка среды разработки

Для реализации функционала шифрования, дешифрования и работы с цифровой подписью было разработано консольное приложение на языке C# с использованием интегрированной среды разработки Microsoft Visual Studio и платформы .NET 8.0.

3.1.2 Использование библиотек

Для реализации криптографических функций и работы с файлами в проекте были подключены следующие пространства имён:

- System.IO — используется для работы с файловой системой: чтение и запись файлов, обработка потоков данных. Благодаря этому модулю реализована возможность считывать файлы по блокам, сохранять зашифрованные данные и работать с цифровыми подписями.

- System.Security.Cryptography — содержит реализацию криптографических алгоритмов, включая RSA для асимметричного шифрования и MD5 для вычисления хэш-суммы. Именно с помощью этого пространства имён реализованы основные функции приложения: шифрование, дешифрование, создание и проверка цифровой подписи.

Эти библиотеки входят в состав стандартного пакета .NET и не требуют дополнительной установки, что делает приложение простым в сборке и запуске на любой системе с установленной средой .NET 8.0.

3.1.3 Технологии и подходы

Проект реализован с использованием принципов объектно-ориентированного программирования. Функционал разделён на отдельные классы, отвечающие за криптографические операции (CryptoService) и работу с файлами (FileService). Такая структура упрощает поддержку кода, позволяет легко расширять функционал и тестирует отдельные части приложения независимо друг от друга.

Консольное приложение взаимодействует с пользователем через текстовое меню, что делает его удобным для тестирования и демонстрации работы основных функций.

Особенностью алгоритма RSA является то, что он работает с данными ограниченной длины. Это ограничение делает невозможным шифрование больших файлов целиком.

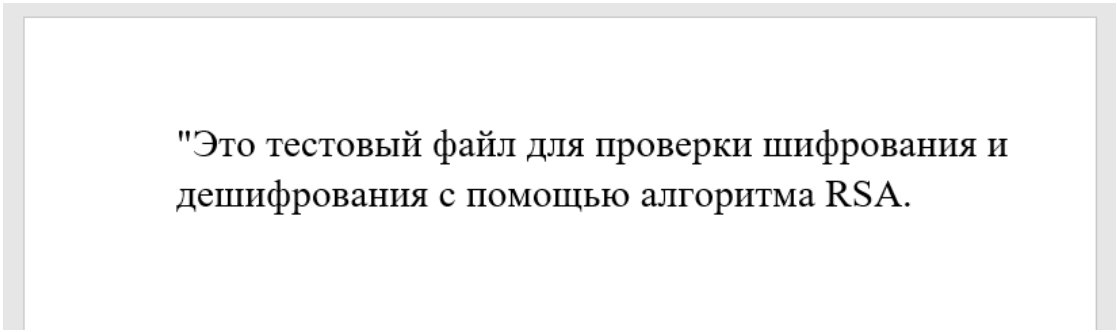
Поэтому в приложении реализована обработка файлов по блокам фиксированного размера. Исходный файл считывается порциями, каждый блок шифруется отдельно и записывается в выходной файл. При дешифровании процесс повторяется в обратном порядке: файл считывается блоками, каждый из которых расшифровывается и записывается в восстановленный файл.

3.2 Тестирование ПО

После реализации всех функциональных модулей приложения была проведена серия тестов, направленных на проверку корректности работы шифрования, дешифрования и создания/проверки цифровой подписи. Тестирование проводилось на файлах различных форматов, включая текстовые документы (.txt), документы Microsoft Office (.docx), таблицы (.xlsx) и PDF-файлы.

3.2.1 Проверка шифрования и дешифрования

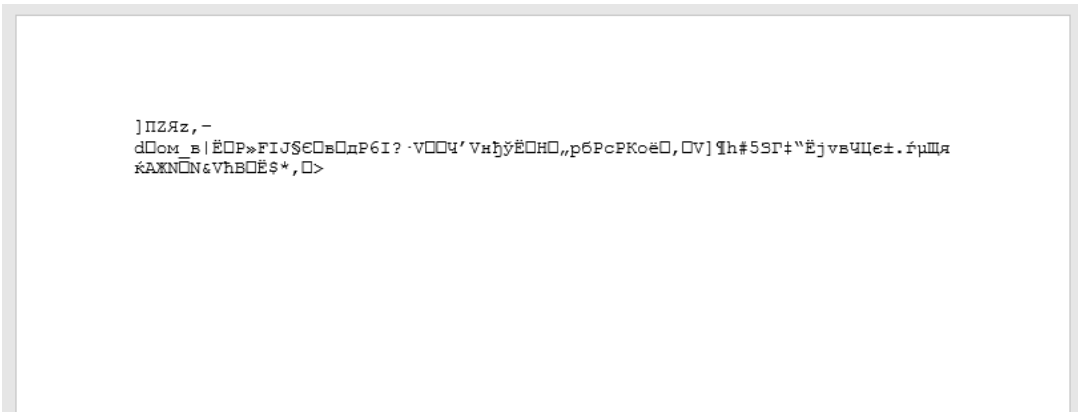
Для проверки функции шифрования был выбран текстовый файл test.txt, содержащий следующий текст, изображенный ниже на рисунке 6.



"Это тестовый файл для проверки шифрования и дешифрования с помощью алгоритма RSA.

Рисунок 6 – Исходный файл

После выбора в меню приложения функции "Зашифровать файл", программа запросила путь к файлу и выполнила его шифрование. Результатом стало появление нового файла test.txt.encrypted. Данный файл представляет собой последовательность зашифрованных блоков, которые невозможно прочитать обычным текстовым редактором. Файл изображен ниже на рисунке 7.



]ПЗЯz,-
dОом_в|ЁОр»FIJ\$€ОвОдР6I?·V□□Ч'VнђўЁ□Н□,,рбРсРКоеО,□V]¶h#55Г±"ЁjvвЧЩе±.±рЩя
КАЖН□N&VhB□ES*,□>

Рисунок 7 – Зашифрованный файл

Далее была выполнена операция дешифрования. При выборе файла test.txt.encrypted программа запросила путь для сохранения расшифрованного файла, после чего создала файл test.txt (без расширения .encrypted). Открытие

файла показало, что содержимое полностью совпадает с оригиналом. Результат дешифрования представлен ниже на рисунке 8.

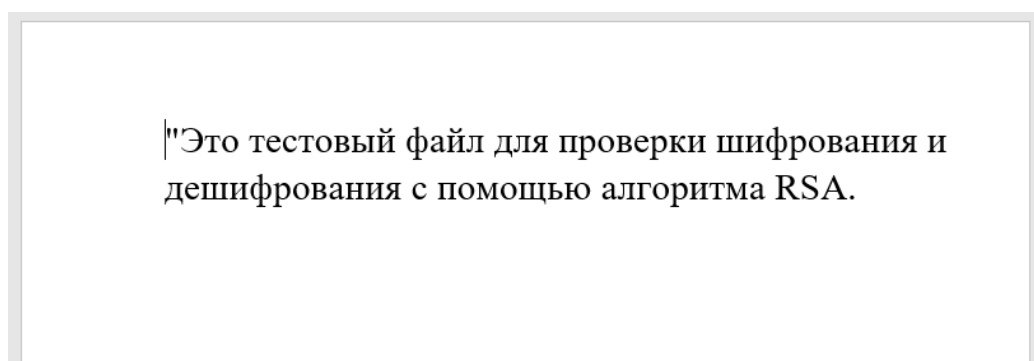


Рисунок 8 – Результат дешифрования

3.2.2 Проверка цифровой подписи

Для проверки работы цифровой подписи был создан файл testcat.jpg. С помощью функции "Создать цифровую подпись" было сформировано два файла: оригинальный документ и файл подписи testcat.jpg.signature.

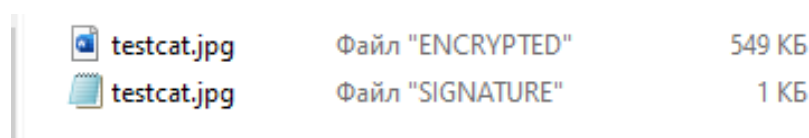


Рисунок 9 – Создание файла с подписью

Для проверки устойчивости системы к изменениям, содержимое файла document.docx было изменено после создания подписи. При повторной проверке цифровой подписи программа корректно определила несоответствие и вывела сообщение: "Подпись недействительна".

ЗАКЛЮЧЕНИЕ

В ходе практики было разработано консольное приложение на языке C#, реализующее шифрование и дешифрование файлов с помощью алгоритма RSA, а также создание и проверку цифровой подписи с использованием хэш-функции MD5.

Поставленные задачи успешно выполнены: изучены принципы криптографии, разработана архитектура приложения, реализованы все функциональные возможности и проведено тестирование, подтвердившее корректную работу программы.

Применение принципов объектно-ориентированного программирования позволило организовать код структурированно и модульно, что обеспечивает простоту сопровождения и расширения приложения. Обработка файлов по блокам позволила обойти ограничения RSA и работать с файлами любого размера.

Разработанное приложение может использоваться для базовой защиты данных и служить основой для дальнейшего улучшения, например, с переходом на более безопасные алгоритмы или добавлением графического интерфейса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Microsoft. Официальная документация по .NET [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/>
2. Microsoft. Документация по классу RSA в .NET [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/api/system.security.cryptography.rsa>
3. Microsoft. Документация по хэш-функции MD5 [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/api/system.security.cryptography.md5>
4. PlantUML. Онлайн-редактор для создания UML-диаграмм [Электронный ресурс] – Режим доступа: <https://www.plantuml.com/plantuml>
5. Microsoft. Официальная документация по C# [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/>