|  |  |
| --- | --- |
|  | Автономное учреждение  профессионального образования  Ханты-Мансийского автономного округа – Югры  «Нефтеюганский политехнический колледж» |

ДОПУЩЕН К СДАЧЕ ГИА

Заместитель директора

по учебно-производственной работе

\_\_\_\_\_\_\_\_\_О.А. Миляр

**ДИПЛОМНАЯ РАБОТА**

«Создание криптографического приложения»

09.02.07 Информационные системы и программирование

Выполнил(а) работу

обучающийся 4 курса \_\_\_\_\_\_\_\_\_\_\_\_ Кузьмин А.С.

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ Козырева В.В.

(подпись)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_ Козырева В.В.

(подпись)

г. Нефтеюганск, 2024

Содержание

[Введение 2](#_Toc1925152403)

[1. Теоретическая часть 4](#_Toc646158950)

[1.1 Описание предметной области 4](#_Toc915498260)

[1.2 Обзор основных принципов криптографии 5](#_Toc1209607505)

[1.3 Рассмотрение существующих криптографических алгоритмов 6](#_Toc1106759367)

[1.4 Аналоги криптографических приложений 8](#_Toc1084802447)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 9](#_Toc585493349)

[2.1. Выбор средства автоматизации 9](#_Toc39069297)

[2.2. Требования к программе 11](#_Toc1723889710)

[2.3. Проектирование программного продукта 12](#_Toc335452675)

[2.4. Текст программы с описанием 12](#_Toc928869769)

[2.5. Тестирование и отладка 17](#_Toc925206157)

[Список литературы 18](#_Toc1427437452)

# **Введение**

В современном информационном обществе безопасность данных и приватность стали одними из основных приоритетов. Криптография - наука о защите информации с помощью математических алгоритмов, играет важную роль в обеспечении безопасности данных. В последние годы возрос интерес к разработке криптографических приложений, предоставляющих возможность шифрования и расшифрования информации с использованием различных алгоритмов и протоколов.

**Актуальность** данной темы заключается в том, что в современном информационном обществе защита данных и конфиденциальность стали важными аспектами. Криптография - это наука, которая занимается защитой информации с помощью различных методов и алгоритмов шифрования.

В процессе разработки приложения будут рассмотрены основные принципы криптографии, включая симметричные и асимметричные алгоритмы шифрования, хэш-функции, цифровые подписи и другие криптографические методы. Будет проведен анализ существующих криптографических библиотек и фреймворков, а также выбраны наиболее подходящие для реализации функций приложения.

Ожидаемыми результатами дипломной работы являются создание работающего криптографического приложения с удобным интерфейсом, адекватной производительностью и высокой безопасностью. Будут проведены тестирование и оценка эффективности разработанного приложения, а также выполнена его сравнительная характеристика с аналогичными программами.

Создание криптографического приложения позволяет обеспечить безопасную передачу и хранение информации. Такие приложения часто используются в банковских системах, электронной коммерции, мобильных приложениях и других областях, где важна конфиденциальность информации.

Криптографические приложения также помогают защищать данные от несанкционированного доступа и атак со стороны злоумышленников. Они обеспечивают аутентификацию пользователей, целостность данных и защиту от подделки информации.

Кроме того, в настоящее время все больше людей осознают важность личной конфиденциальности и защиты своих данных. Популярность мессенджеров с шифрованием, таких как WhatsApp и Signal, свидетельствует о повышенном интересе к криптографическим приложениям.

Таким образом, создание криптографического приложения актуально для обеспечения безопасности, защиты данных и конфиденциальности в современном информационном обществе.

**Объект исследования:** криптографическое приложение.

**Предмет исследования:** разработка криптографического приложения для обеспечения безопасности, защиты данных и конфиденциальности.

**Цель** данной дипломной работы состоит в создании криптографического приложения, которое предоставит пользователю возможность безопасного обмена сообщениями, файлами и другими данными. Программа будет базироваться на современных алгоритмах шифрования и протоколах, обеспечивающих конфиденциальность, аутентификацию и целостность информации.

Определены следующие задачи:

1. Разработка и реализация алгоритмов шифрования и дешифрования данных.

2. Создание пользовательского интерфейса для приложения.

3. Разработка функций для генерации и управления ключами.

4. Реализация функций для обмена зашифрованными данными.

5. Обеспечение безопасности и защиты данных.

6. Тестирование приложения.

7. Обновление и поддержка приложения.

Методы: поиск информации, анализ информации, проектировка

разработки.

**Структура дипломной работы:** дипломная работа состоит из введения, теоретической части, практической части, заключения и списка использованной литературы.

# **1. Теоретическая часть**

# **1.1 Описание предметной области**

Криптография – это метод защиты информации путем использования закодированных алгоритмов, хэшей и подписей. Информация может находиться на этапе хранения (например, файл на жестком диске), передачи (например, электронная связь между двумя или несколькими сторонами) или использования (при применении для вычислений).

Криптография использует некоторые низкоуровневые криптографические алгоритмы для достижения одной или нескольких из этих целей информационной безопасности. Среди этих инструментов – алгоритмы шифрования, алгоритмы цифровой подписи, алгоритмы хеширования и другие функции.

Для реализации криптографического приложения потребуется:

1. Выбрать подходящие криптографические алгоритмы, такие как AES, RSA или SHA-256, и реализовать их в приложении.

2. Создание пользовательского интерфейса для приложения включает в себя создание формы или окна, где пользователь может вводить данные для шифрования или дешифрования, а также кнопки для запуска процесса шифрования или дешифрования.

3. При шифровании данных требуется использование ключа, который используется для защиты информации. Приложение должно предоставлять возможность генерации, сохранения и загрузки ключей для использования в шифровании и дешифровании данных.

4. Необходимо обеспечить безопасность данных, хранящихся на устройстве пользователя или передаваемых через сеть, путем реализации дополнительных мер безопасности, таких как защита паролем или шифрование хранилища данных.

5. Не менее важной задачей является тестирование приложения для обнаружения и устранения возможных уязвимостей или ошибок в его работе.

6. Криптографические алгоритмы и технологии постоянно меняются и обновляются. Необходимо проследить за последними разработками в области криптографии и, при необходимости, обновить приложение, чтобы обеспечить его соответствие современным стандартам безопасности.

# **1.2 Обзор основных принципов криптографии**

Криптография - это наука, посвященная методам защиты информации, используя различные методы и принципы шифрования.

Основные принципы криптографии включают:

1. Конфиденциальность: Главный принцип криптографии состоит в обеспечении конфиденциальности передаваемой информации. Шифрование используется для преобразования исходного текста в шифрованный вид таким образом, что только авторизованный получатель сможет его расшифровать.

2. Целостность: Криптография также обеспечивает целостность информации. Это означает, что данные не могут быть изменены без предварительного обнаружения. Методы контроля целостности, такие как хэширование и цифровые подписи, используются для проверки, не была ли информация изменена в процессе передачи.

3. Аутентификация: Криптография помогает обеспечить аутентичность информации и подлинность участников взаимодействия. Для этого могут использоваться цифровые подписи и сертификаты. Аутентификация обеспечивает уверенность в том, что отправитель и получатель являются теми, за кого они себя выдают.

4. Неотказуемость: Криптография также помогает в решении проблемы отказа отдельных участников взаимодействия от подписания или расшифровки информации. Цифровые подписи и другие методы позволяют третьей стороне доказать, что конкретный участник совершил какое-то действие.

5. Ключевая безопасность: Криптография оперирует с использованием ключей - значений, используемых для шифрования и расшифровки данных. Ключевая безопасность особенно важна, поскольку любое разглашение или компрометация ключа может привести к нарушению безопасности и конфиденциальности информации.

Основные принципы криптографии играют важную роль в обеспечении безопасности информации в различных сферах, включая защиту персональных данных, интернет-банкинг, электронную коммерцию и другие.

# **1.3 Рассмотрение существующих криптографических алгоритмов**

Криптографический алгоритм - это математический алгоритм, который применяется для обеспечения конфиденциальности, целостности и аутентичности информации. В настоящее время существует множество криптографических алгоритмов, которые используются в различных областях, таких как защита данных, электронная коммерция, цифровые подписи и т. д. Ниже рассмотрены некоторые из наиболее распространенных криптографических алгоритмов.

1. AES (Advanced Encryption Standard) - это симметричный алгоритм шифрования, который широко используется для защиты данных. Он был выбран в качестве стандарта США и рекомендован для использования во всем мире. AES основан на методе замены и перестановки байтов в блоках данных.

2. RSA - это асимметричный алгоритм шифрования, который используется для защиты данных и цифровых подписей. RSA основан на сложности факторизации больших простых чисел и использует пару ключей - открытый и закрытый. Открытый ключ используется для шифрования сообщений, а закрытый ключ - для расшифровки.

3. DES (Data Encryption Standard) - это старый симметричный алгоритм шифрования, который был разработан в 1970-х годах. Он использует 56-битные ключи и блоки данных размером 64 бита. Несмотря на свою надежность в прошлом, DES сейчас считается слабым из-за ограниченной длины ключа.

4. SHA (Secure Hash Algorithm) - это семейство хэш-функций, которые используются для обеспечения целостности данных. Они принимают произвольно длинный вход и генерируют фиксированную длину хэш-кода. SHA-1 является наиболее широко используемым в прошлом, но сейчас рекомендуется использовать более безопасный SHA-256 или SHA-3.

5. ECC (Elliptic Curve Cryptography) - это асимметричный алгоритм шифрования, который основан на сложных математических проблемах в области эллиптических кривых. ECC предлагает ту же криптографическую безопасность с более короткими ключами, что делает его более эффективным для встроенных систем и мобильных устройств.

6. TLS/SSL - это протоколы безопасной передачи данных по сети. Они используют комбинацию различных криптографических алгоритмов, включая симметричное и асимметричное шифрование, для обеспечения конфиденциальности и целостности данных при передаче через открытую сеть.

Это лишь некоторые из множества криптографических алгоритмов, которые используются в современной криптографии. В зависимости от потребностей и требований безопасности конкретного приложения или системы, может использоваться различный набор алгоритмов.

# **1.4 Аналоги криптографических приложений**

Ниже приведены некоторые аналоги криптографических приложений:

1. ProtonMail (приложение 1) - это приложение для безопасной электронной почты. Его аналогом может быть Gmail с использованием дополнительных плагинов и расширений для шифрования писем.

2. Signal (приложение 1) - это приложение для защищенных сообщений и звонков, основанное на протоколе шифрования end-to-end (отправитель-получатель). Его аналогами могут быть Telegram с использованием режима "секретных чатов" или WhatsApp с включенной функцией "end-to-end" шифрования.

3. VeraCrypt (приложение 1) - это программное обеспечение для шифрования дисков и создания зашифрованных контейнеров. Его аналогом может быть BitLocker, встроенный в некоторые версии Windows, или Cryptomator, который предоставляет шифрование в облаке.

4. Tails (приложение 1) - это операционная система, предназначенная для анонимности и безопасности связи. Аналогами могут быть Qubes OS или Whonix, которые также обеспечивают высокий уровень безопасности и анонимности в Интернете.

5. LastPass (приложение 1) - это менеджер паролей, который хранит и автоматически заполняет пароли для пользователей. Его аналогами могут быть KeePass или 1Password, которые также предлагают функции шифрования и защиты паролей.

Нужно отметить, что каждое из перечисленных аналогов имеет свои особенности и некоторые могут быть более подходящими для конкретных потребностей пользователей, в зависимости от их требований к безопасности и удобству использования.

# **2. ПРАКТИЧЕСКАЯ ЧАСТЬ**

# **2.1. Выбор средства автоматизации**

Данный проект необходимо реализовать в среде программирования Visual Studio Code с помощью языка JavaScript.

Visual Studio Code (VS Code) — текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Преимущества Visual Studio Code:

1. Открытый и бесплатный исходный код: VS Code разработан и поддерживается Microsoft как бесплатный и открытый исходный код с поддержкой GitHub. Это означает, что он доступен для всех, а его исходный код может быть изменен и улучшен со всеми по мере необходимости.
2. Настраиваемый интерфейс: VS Code имеет настраиваемый интерфейс, который позволяет пользователям настраивать его в соответствии с его индивидуальными потребностями. С помощью него можно сделать очень удобный и интуитивно понятный интерфейс программирования.
3. Поддержка многих языков программирования: VS Code поддерживает множество языков программирования, что делает его очень полезным для различных программистов, включая PHP, C++, Java, Python, Ruby, SQL и многие другие.
4. Отличные инструменты для отладки: VS Code предоставляет широкий выбор отладочных инструментов, которые делают процесс отладки кода проще и быстрее.
5. Терминал внутри программы: Программа имеет командную строку, что позволяет вам работать с файловой системой и выполнить любые команды прямо внутри программы.
6. Расширяемость: VS Code расширяем и может быть дополнен различными плагинами, которые добавляют больше функциональности в процесс программирования.
7. Хорошее сообщество поддержки: Существует большое сообщество пользователей, которые активно развивают и поддерживают VS Code. Это означает, что вы можете получить помощь, если у вас возникнут проблемы, или задать вопрос, если есть необходимость.

JavaScript – это интерпретируемый язык программирования, который обычно используется для создания интерактивных веб-страниц. Он поддерживает многие функции, которые обычно присутствуют в объектно-ориентированных языках программирования, таких как наследование, инкапсуляция и полиморфизм.

JavaScript можно использовать как на стороне клиента в браузере, так и на стороне сервера благодаря Node.js. Синтаксис языка основан на C, у него есть своя стандартная библиотека, которая содержит множество полезных функций и методов для работы с различными типами данных и объектами. В JavaScript можно использовать различных фреймворки и библиотеки, такие как jQuery, React, Vue.js и другие, которые значительно упрощают работу с языком.

В качестве вспомогательных языков программирования были выбраны:

HTML (Hypertext Markup Language) является языком разметки, который используется для создания веб-страниц и придания им структурированности. Он определяет элементы веб-страницы и их расположение на странице, такие как заголовки, параграфы, ссылки, изображения и т.д. Кроме того, HTML позволяет веб-разработчику включать другие типы мультимедиа, такие как видео и аудио, в свои веб-страницы.

CSS (Cascading Style Sheets) – это язык стилей, который используется для изменения внешнего вида веб-страниц, созданных на HTML. CSS позволяет веб-разработчикам задавать цвета, шрифты, размеры и многое другое для элементов HTML. Он позволяет создавать красивые макеты и управлять внешним видом и расположением элементов веб-страницы. Одно из главных преимуществ в использовании CSS заключается в том, что он позволяет снизить объем HTML-кода, что улучшает производительность веб-страниц.

# **2.2. Требования к программе**

Моя программа должна выполнять такие функции как:

* Иметь удобный и интуитивно понятный пользовательский интерфейс, позволяющий легко выполнять операции шифрования и дешифрования.
* Иметь надежный алгоритм шифрования, который считается безопасным и хорошо зарекомендовал себя.
* Использовать безопасные методы генерации ключей.
* Предоставлять механизмы для безопасного хранения и управления ключами шифрования.
* Поддерживать шифрование и дешифрование данных в различных форматах файлов (например, текстовые файлы, изображения, архивы).

USE-CASE схема программы (приложение 2)

# **2.3. Проектирование программного продукта**

При создании приложения «Криптографическое приложение» был использован следующий прием – модульное программирование.

Модульное программирование — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Модуль — функционально законченный фрагмент программы. Во многих языках (но далеко не обязательно) оформляется в виде отдельного файла с исходным кодом или поименованной непрерывной её части. Некоторые языки предусматривают объединение модулей в пакеты.

# **2.4. Текст программы с описанием**

«Криптографическое приложение» позволяет шифровать и дешифровать данные с использованием различных криптографических алгоритмов и ключей.

В своей программе я использовал:

* Язык программирования JavaScript
* Язык разметки HTML
* Язык описания внешнего вида CSS
* Текстовый редактор – Visual Studio Code

Плагины:

Live Server – позволяет вам видеть изменения кода, отраженные в браузере. Он запускает локальный сервер разработки с функцией перезагрузки в реальном времени как для статических, так и для динамических страниц. Каждый раз, когда вы сохраняете свой код, вы мгновенно видите изменения, отраженные в браузере.

Prettier – Code Formatter – самоуверенный форматировщик кода. Он обеспечивает согласованный стиль, анализируя ваш код и перепечатывая его с помощью собственных правил, учитывающих максимальную длину строки, при необходимости перенося код.

(Приложение 2)

Размер - 2,91 МБ (3 054 221 байт)

На диске - 4,18 МБ (4 386 816 байт)

Код index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

<link rel="stylesheet" href="style.css">

<title>Криптографическое приложение</title>

</head>

<body>

<div class="container">

<h3>Криптографическое приложение</h3>

<input type="text" value="Hello, World!" class="form-control">

<div class="btn-box">

<button class="btn btn-primary btn-send">Send message</button>

<button class="btn btn-success btn-get" disabled>Get message</button>

</div>

<output></output>

</div>

<script src="client.js"></script>

</body>

</html>

Код style.css:

h3,

.btn-box {

margin: .5em;

text-align: center;

}

input,

output {

display: block;

margin: 1em auto;

text-align: center;

}

Код client.js:

const generateKey = async () =>

window.crypto.subtle.generateKey({

name: 'AES-GCM',

length: 256,

}, true, ['encrypt', 'decrypt'])

const encode = data => {

const encoder = new TextEncoder()

return encoder.encode(data)

}

const generateIv = () =>

window.crypto.getRandomValues(new Uint8Array(12))

const encrypt = async (data, key) => {

const encoded = encode(data)

const iv = generateIv()

const cipher = await window.crypto.subtle.encrypt({

name: 'AES-GCM',

iv

}, key, encoded)

return {

cipher,

iv

}

}

const pack = buffer => window.btoa(

String.fromCharCode.apply(null, new Uint8Array(buffer))

)

const unpack = packed => {

const string = window.atob(packed)

const buffer = new ArrayBuffer(string.length)

const bufferView = new Uint8Array(buffer)

for (let i = 0; i < string.length; i++) {

bufferView[i] = string.charCodeAt(i)

}

return buffer

}

const decode = byteStream => {

const decoder = new TextDecoder()

return decoder.decode(byteStream)

}

const decrypt = async (cipher, key, iv) => {

const encoded = await window.crypto.subtle.decrypt({

name: 'AES-GCM',

iv

}, key, cipher)

return decode(encoded)

}

// поле для ввода сообщения, которое будет зашифровано

const input = document.querySelector('input')

// контейнер для вывода результатов

const output = document.querySelector('output')

// ключ

let key

const encryptAndSendMsg = async () => {

const msg = input.value

// шифрование

key = await generateKey()

const {

cipher,

iv

} = await encrypt(msg, key)

// упаковка и отправка

await fetch('http://localhost:3000/secure-api', {

method: 'POST',

body: JSON.stringify({

cipher: pack(cipher),

iv: pack(iv)

})

})

output.innerHTML = `Сообщение <span>"${msg}"</span> зашифровано.<br>Данные отправлены на сервер.`

}

const getAndDecryptMsg = async () => {

const res = await fetch('http://localhost:3000/secure-api')

const data = await res.json()

// выводим данные в консоль

console.log(data)

// распаковка и расшифровка

const msg = await decrypt(unpack(data.cipher), key, unpack(data.iv))

output.innerHTML = `Данные от сервера получены.<br>Сообщение <span>"${msg}"</span> расшифровано.`

}

document.querySelector('.btn-box').addEventListener('click', e => {

if (e.target.classList.contains('btn-send')) {

encryptAndSendMsg()

e.target.nextElementSibling.removeAttribute('disabled')

} else if (e.target.classList.contains('btn-get')) {

getAndDecryptMsg()

}

})

Интерфейс программы (приложение 3)

# **2.5. Тестирование и отладка**

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

Для тестирования и отладки криптографического приложения можно использовать различные подходы и техники.

1. Функциональное тестирование. Проверка корректности работы каждой функции приложения и достижения ожидаемого результата. Напри мер, проверка правильности шифрования и дешифрования данных с использованием различных алгоритмов и ключей.
2. Тестирование производительности. Проверка скорости работы приложения и его отклика на запросы пользователя. Например, измерение времени шифрования и дешифрования данных различных размеров и форматов.
3. Тестирование безопасности. Тестирование устойчивости к известным криптографическим атакам. Например, атакам на основе известного открытого текста.

В заключении данной курсовой работы можно подвести итоги, которые отражают актуальность данной темы. Разработка криптографических приложений требует глубокого понимания криптографических алгоритмов, методов управления ключами и мер безопасности. Следует помнить, что криптография — это постоянно развивающаяся область, и разработчики должны быть в курсе последних тенденций и угроз безопасности, чтобы гарантировать, что их приложения остаются безопасными и надежными.

Выводы проделанной мной работы:

Был проведен анализ приложений, имеющихся в общем доступе и соответствующих тематике криптографических приложений.

Я сделал приложение, которое реализует следующие функции:

* Имеет удобный и интуитивно понятный пользовательский интерфейс, позволяющий легко выполнять операции шифрования и дешифрования.
* Имеет надежный алгоритм шифрования, который считается безопасным и хорошо зарекомендовал себя.
* Использует безопасные методы генерации ключей.
* Предоставляет механизмы для безопасного хранения и управления ключами шифрования.
* Поддерживает шифрование и дешифрование данных в различных форматах файлов (например, текстовые файлы, изображения, архивы).

# **Список литературы**

1. Википедия-свободная энциклопедия [Электронный ресурс] - https://ru.wikipedia.org/wiki/

2. Фримен, Робсон – Изучаем программирование на JavaScript, 2014 – 640 с.

3. Кит Мартин - Криптография. Как защитить свои данные в цифровом пространстве, 2023 – 378 с.

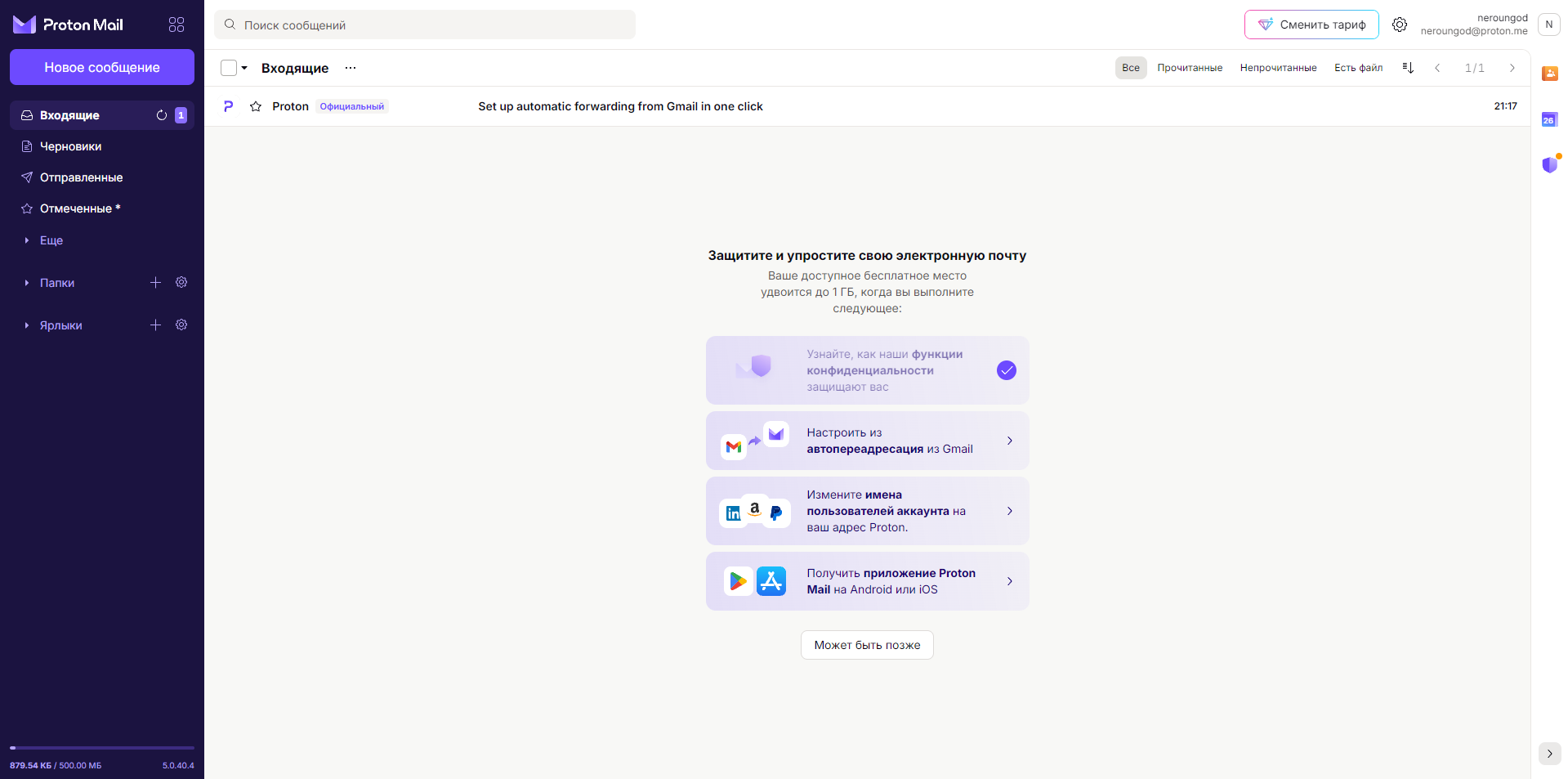
4. Скляров - Искусство защиты и взлома информации, 2004 – 289 с.

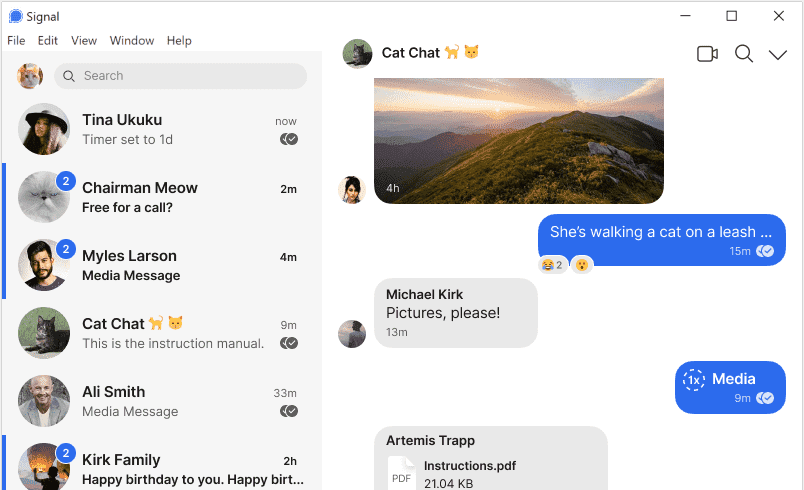
5. Metanit.com – сайт о программировании [Электронный ресурс] - https://metanit.com/sharp/

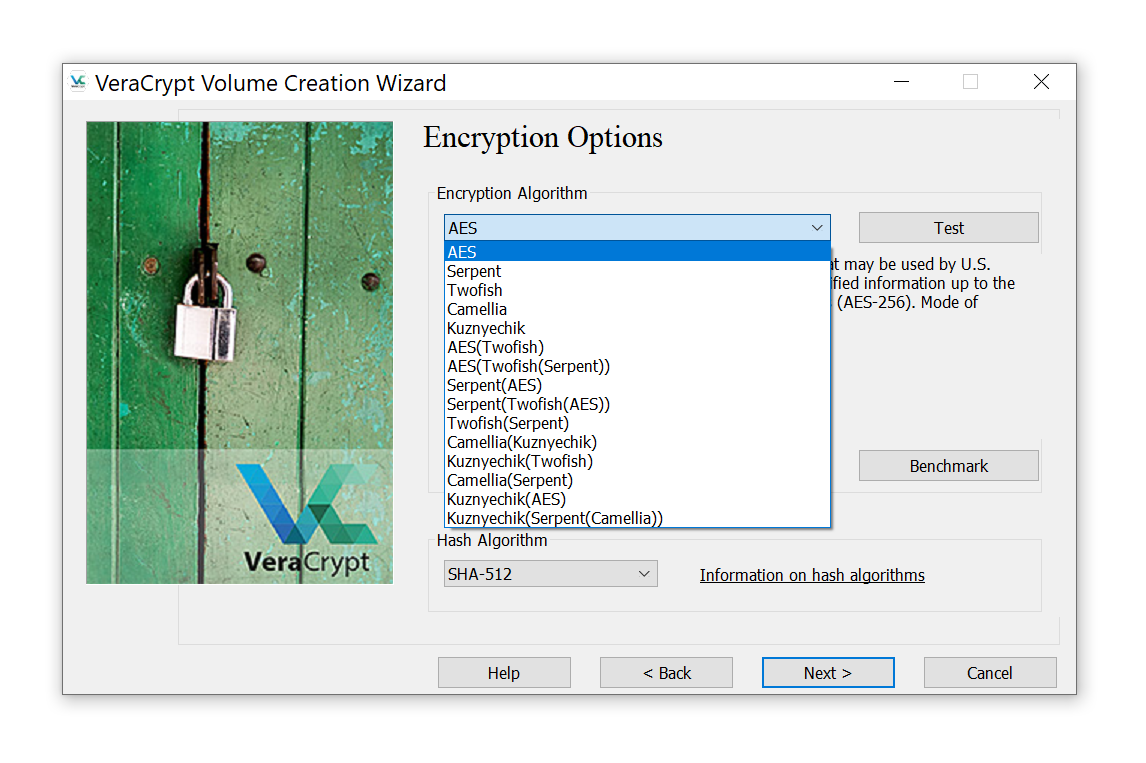
6. Дакетт Джон HTML и CSS. Разработка и дизайн вебсайтов. Эксмо 2020 - 480 с.

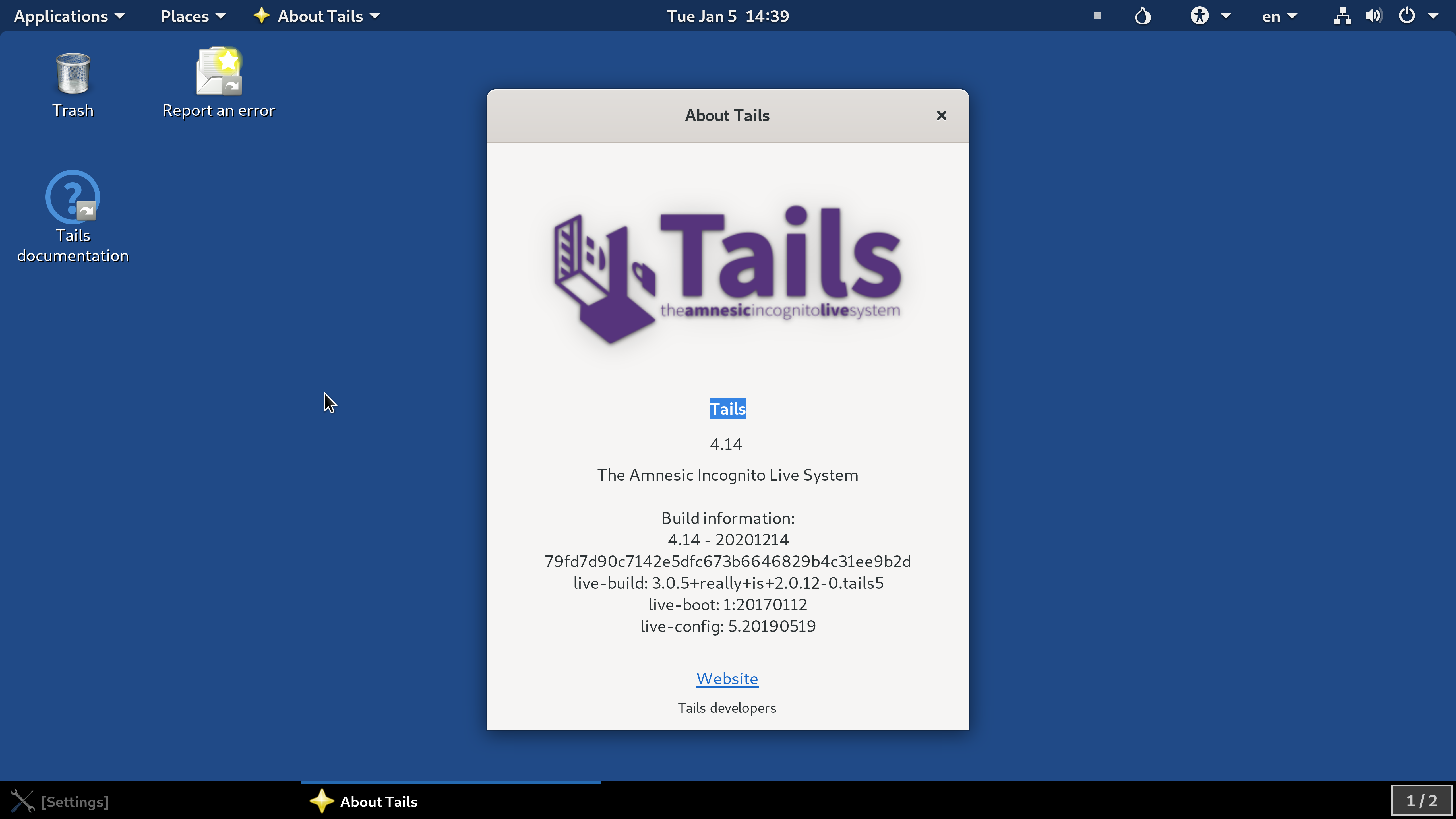
**Приложения**

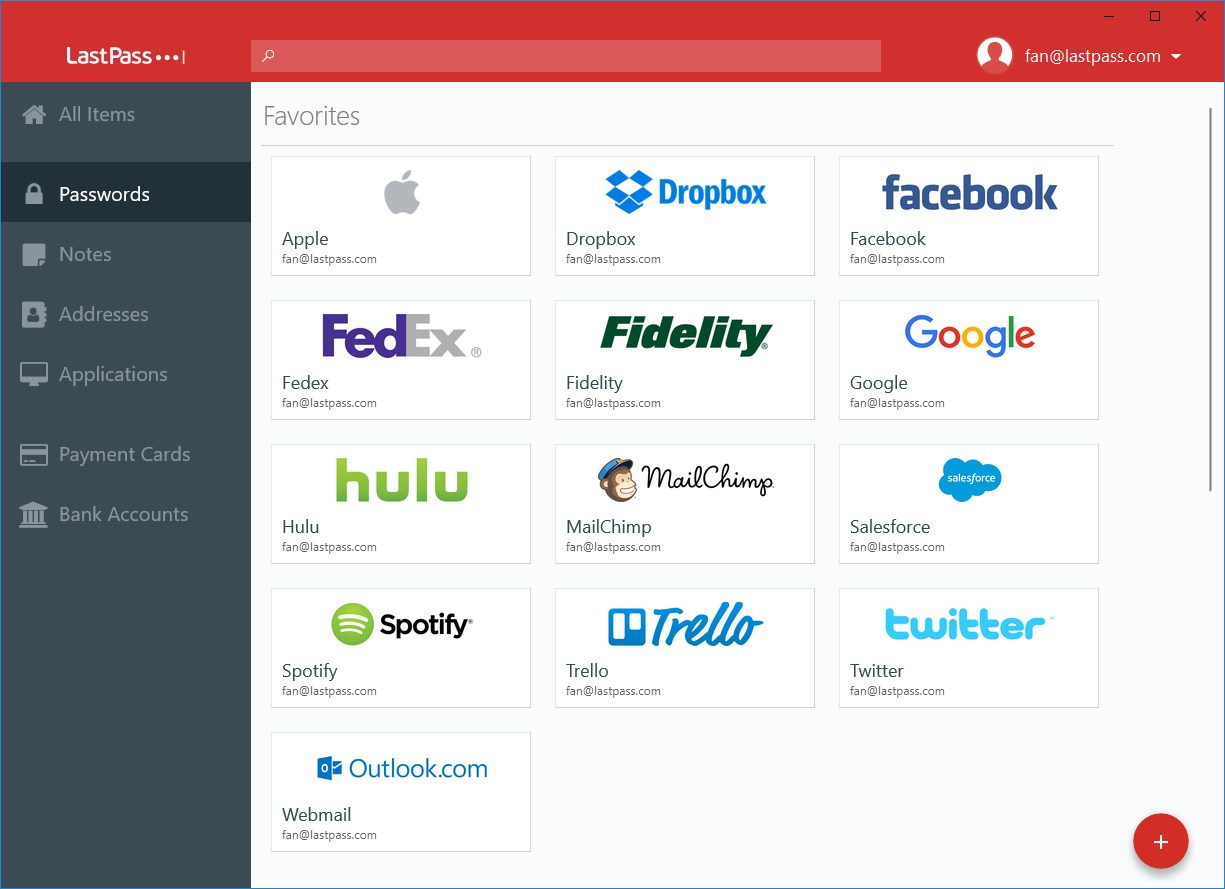
**Приложение 1**



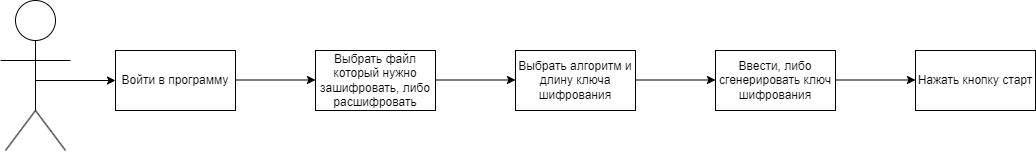


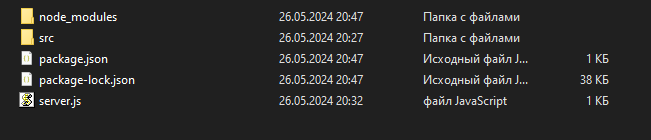






**Приложение 2**





**Приложение 3**

