БЕЗОПАСНАЯ ПЕРЕДАЧА СООБЩЕНИЙ С РАЗДЕЛЕНИЕМ ДАННЫХ ЧЕРЕЗ ПОЧТОВЫЕ СЕРВЕРЫ

Степанов П. П.[1], Никонова Г. В.[2]

DOI: 10.21681/2311-3456-2024-2-120-129

Целью исследования *заключается в разработке методов, которые могут быть использованы для повышения безопасности передачи информации и повышения эффективности за счет избыточности современной сетевой инфраструктуры. Одной из таких задач является надежность пересылки электронной почты и сохранение конфиденциальности пересылаемых сообщений.*

Методом проведения *исследования является анализ состава и содержания задач, связанных с повышением эффективности существующих каналов связи, а также разработки приложений, обеспечивающих разбиение файлов для безопасной передачи по нескольким каналам.*

В результате исследования *предложено, что решением проблемы защиты пересылаемых данных может стать создание механизма, способного передавать сообщения электронной почты на основе стандартных почтовых протоколов с использованием разделения передаваемых данных и подбор ключа с использованием модификатора входа хэш-функции. Такие методы разделения файлов позволяют реализовать различные схемы передачи данных по нескольким каналам. В процессе работы было разработано программное обеспечение, которое выполняет разбиение файлов по нескольким алгоритмам. Предложенные методы разделения файлов позволяют реализовать различные схемы передачи данных по нескольким каналам. Файл ключа может передаваться отдельно, или как часть одного из блока данных. Приведены примеры программ разделения файлов на части для систем передачи данных с разделением пакетов. Рассмотрены алгоритмы разделения файла на симметричные и несимметричные части. Предложена последующая модификация таких алгоритмов, позволяющая ассиметрично разделять файлы на N частей. Приведен пример реализации интерфейса IFileSystemServices, который содержит методы создания ключа для симметричного и асимметричного разбиения. Также приведена реализация интерфейса ISplitServices, определяющего логику разбиения файла и вызов метода закрытия потоков, связанные с файлами.*

Практическая ценность *состоит в том, что представлен способ маркировки блоков разделенных данных методом последовательности псевдослучайных чисел, сгенерированных генераторами псевдослучайных чисел (ГПСЧ) на основе линейно-конгруэнтных алгоритмов. Разработан алгоритм синхронной генерации уникальных идентификаторов от отправителя и получателя сообщений, для реализации обмена информацией. Представленная методика является универсальным средством, позволяющим защищать от несанкционированного использования как программные продукты, так и другие объекты интеллектуальной собственности.*

Ключевые слова: *ключ, протокол, разбиение, интерфейс, генерация паролей, хеш-функции.*

SECURE TRANSMISSION OF MESSAGES WITH DATA SEPARATION VIA MAIL SERVERS

Stepanov P. P.[3], Nikonova G. V.[4]

The purpose of the study *is to develop methods that can be used to improve the security of information transmission and increase efficiency due to the redundancy of modern network infrastructure. One of these challenges is the reliability of email forwarding and maintaining the confidentiality of forwarded messages.*

1 Степанов Петр Петрович, старший преподаватель, федерального государственного автономного учреждения высшего образования «Омский государственный технический университет» (ОмГТУ), г. Омск, Россия, E-mail: omsk.petr@gmail.com

2 Никонова Галина Владимировна, кандидат технических наук, доцент, федерального государственного автономного учреждения высшего образования «Омский государственный технический университет» (ОмГТУ), г. Омск, Россия. E-mail: nikonova@omgtu.ru

3 Petr P. Stepanov, Senior Lecturer, Omsk State Technical University (Omsk State Technical University), Omsk, Russia, E-mail: omsk.petr@gmail.com

4 Galina V. Nikonova, Candidate of Technical Sciences, Associate Professor, Omsk State Technical University (Omsk State Technical University), Omsk, Russia. E-mail: nikonova@omgtu.ru

The method of research *is to analyze the composition and content of tasks related to increasing the efficiency of existing communication channels, as well as the development of applications that provide file splitting for secure transmission over several channels.*

As a result of the study*, it was proposed that a solution to the problem of protecting transmitted data could be the creation of a mechanism capable of transmitting email messages based on standard mail protocols using separation of transmitted data and selection of a key using a hash function input modifier. Such file separation methods allow you to implement various data transfer schemes over multiple channels. In the process, software was developed that splits files using several algorithms. The proposed methods for separating files make it possible to implement various data transfer schemes over several channels. The key file can be transferred separately, or as part of one of the data blocks. Examples of programs for splitting files into parts for data transmission systems with packet separation are given. Algorithms for dividing a file into symmetrical and asymmetrical parts are considered. A subsequent modification of such algorithms is proposed, which makes it possible to asymmetrically divide files into N parts. An example implementation of the IFileSystemServices interface is provided, which contains methods for creating a key for symmetric and asymmetric partitioning. Also provided is an implementation of the ISplitServices interface, which defines the logic for splitting a file and calling a method for closing streams associated with files.*

Practical value *is that the method of labeling blocks of separated data by the method of sequence of pseudo-random numbers generated by the generators of pseudo-random numbers (GPSC) based on linear-conload algorithms. The algorithm of synchronous generation of unique identifiers from the sender and recipient of messages was developed to implement information exchange. The presented technique is a universal tool that allows you to protect both software products and other intellectual property from unauthorized use.*

Keywords: *key, protocol, partitioning, interface, password generation, hash functions.*

Введение

С развитием технологий передачи данных и глобализацией Интернета возникают различные задачи, связанные с повышением безопасности существующих каналов связи. Одной из таких задач является надежность пересылки электронной почты и сохранение конфиденциальности пересылаемых сообщений. Решением этой проблемы может стать создание механизма, способного передавать сообщения электронной почты на основе стандартных почтовых протоколов с использованием разделения передаваемых данных и избыточности сетевой инфраструктуры.

Сегодня распределенные технологии и реализующее их программное обеспечение приобретают все большее значение для решения такого рода задач [1, 2]. Современные реалии требуют постоянного повышения безопасности и эффективности этих систем и диктуют задачи, которые невозможно решить без использования научного подхода. В связи с тем, что современные линии связи могут иметь огромную протяженность, сейчас они тянутся на сотни тысяч километров, в результате можно реализовать физическое подключение к каналу, воздействовать на каналы, нарушая функционирование системы [3].

Наиболее распространены угрозы связанные с перехватом трафика, которые являются серьезной проблемой защиты данных от несанкционированного доступа, в частности такой вид вторжения, как подмена протокола разрешения адресов (ARP Spoofing) — разновидность сетевой атаки типа «человек посередине» MITM (англ. Man in the middle) [4, 5].

Подобные атаки относятся к довольно опасному типу, поскольку основаны на недостатках ARP протокола посредством отправки поддельного ARP пакета для осуществления DOS атаки [5–8].

Кроме того, существуют и другие виды атак «Man in the Middle» [8–11]:

- MAC Spoofing – здесь возможна подмена MAC адреса [8, 10];
- ICMP redirect. Связан с возможностью посылки с любого хоста в сегменте сети ложного redirect-сообщения от имени маршрутизатора на атакуемый хост [8, 11];
- DHCP Spoofig. Протокол DHCP осуществляет динамическое назначение IP-адреса компьютеру-клиенту, который временно подключается к сети, т.е. пакеты, предназначенные для атакуемого компьютера, будут приниматься и на компьютере с измененным MAC адресом [10, 11].

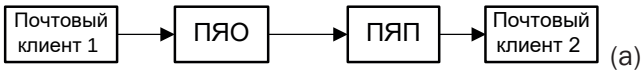
Взлом маршрутизатора – способ взлома роутера и переконфигурирование DHCP так чтобы адресом шлюза и DNS сервера был указан компьютер атакующего [10, 11].

В целом можно сказать, что проблема повышения безопасности и надежности передачи данных в настоящее время очень актуальна и важность проблемы возрастает прямо пропорционально развитию сетевых технологий [3].

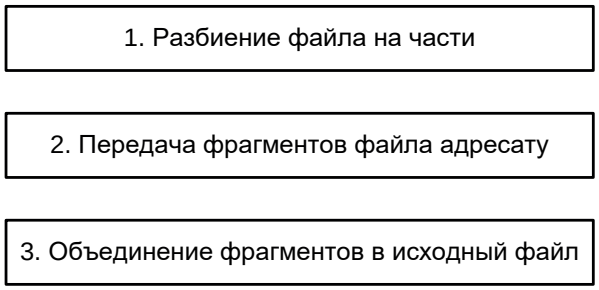
Решение (Передача сообщений электронной почты с разделением данных через почтовые серверы)

Рассмотрим классическую схему отправки сообщений электронной почты. На компьютере отправителя устанавливается программа почтового клиента, которая содержит информацию о почтовых ящиках отправителя, зарегистрированных на почтовых серверах (адреса, учетные данные доступа, протоколы обмена данными) [12]. Адресная книга также содержит адреса получателей, которым будут доставляться сообщения. Если необходимо переслать сообщение, почтовый клиент на стороне отправителя инициирует сеанс обмена информацией с сервером, на котором расположен почтовый ящик отправителя (ПЯО), и отправляет на него данные, предназначенные для отправки получателю. Почтовый сервер, в свою очередь, пересылает данные на почтовый сервер, на котором зарегистрирован почтовый ящик получателя (ПЯП), в соответствии с адресом электронной почты получателя. При проверке почты почтовым клиентом получателя пересылаемые данные загружаются на локальный компьютер получателя для дальнейшей работы с ними (рис. 1, а).

За основу разрабатываемой системы передачи сообщений возьмем мультиплексную систему [13]. Работа мультиплексной системы происходит в несколько этапов (рис. 1, б).



(а)



(б)

Рис. 1. Классическая схема передачи сообщений (а); мультиплексная передача данных (б)

Программа, осуществляющая передачу сообщения, имеет вид почтового клиента с функцией разбиения файла и рассылки фрагментов на различные почтовые сервера по определенному алгоритму. Этот же клиент осуществляет получение фрагментов и сборку сообщения на стороне получателя.

На рисунке 2 представлены варианты реализаций схем с рассылкой фрагментов сообщений:

а) с одного почтового ящика отправителя (ПЯО) на несколько почтовых ящиков получателя (ПЯП1 ... ПЯПn);
б) с нескольких почтовых ящиков (ПЯП1 ... ПЯПn) на один (ПЯО);
в) с нескольких почтовых ящиков (ПЯП1 ... ПЯПn) на несколько (ПЯП1 ... ПЯПn).

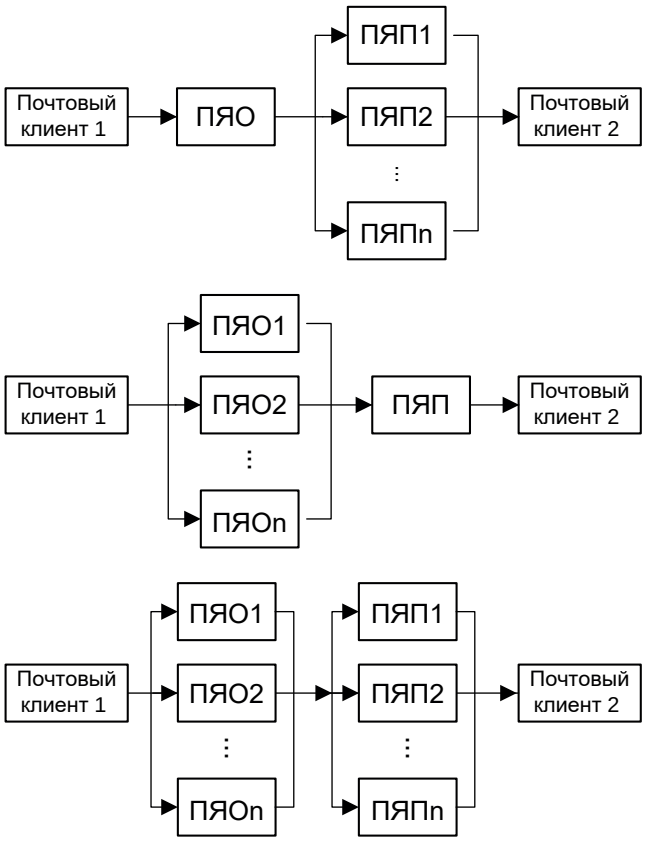


Рис. 2. Схема рассылки один – много, много – один, много – много

Для проведения эксперимента и определения эффективности данной схемы предлагается реализовать возможность использования в почтовом клиенте всех трех вариантов.

Программа почтовый клиент работает по стандартным почтовым протоколам POP3 и SMTP. SMTP используется для отправки фрагментов сообщения, POP3 – для получения [14, 15]. Протокол IMAP решено не использовать, так как дополнительные возможности этого протокола могут помешать корректной работе данного программного обеспечения.

Подготовка файла к передаче осуществляется почтовым клиентом по следующему алгоритму:

1) осуществляется инверсия файла;
2) производится разбиение файла, и перетасовка данных по определенному алгоритму;
3) осуществляется разбиение файла на фрагменты для передачи;
4) фрагменты маркируются идентификаторами.

После процедуры разделения фрагменты пересылаются в почтовые ящики получателя. Для этого в почтовом клиенте создается список адресов

почтовых ящиков отправителя и получателя, который можно использовать для мультиплексной передачи. Мы выбираем схему передачи фрагментов: один – много, много – один, много – много. В случае «много-много» случайным образом выбирается почтовый ящик отправителя, из которого будет отправлен первый фрагмент. Если необходимо отправить другой фрагмент из этого почтового ящика в текущем сеансе, он может быть использован только для передачи нечетного фрагмента. Затем случайным образом выбирается адрес почтового ящика, куда будет отправлен файл. Ящики для приема фрагментов также выбираются по принципу четности – нечетности. Если в почтовый ящик был отправлен нечетный фрагмент, то четные фрагменты не отправляются в него во время текущего сеанса. Далее выбирается пара полей для второго фрагмента и так далее.

Почтовый клиент получателя опрашивает почтовые ящики, считывает фрагменты и собирает переданный файл по умолчанию. Чтобы получить исходный файл, вам нужно применить обратное преобразование к результирующему файлу, то есть вернуть перетасованные фрагменты данных на их место и выполнить инверсию. Можно использовать единый алгоритм преобразования, но для повышения надежности передачи сообщений предлагается ввести временную метку, в зависимости от которой к файлу применяется определенная схема разделения и перетасовки. Это позволит вам использовать несколько изменяющихся схем разделения и перетасовки.

Описанная система позволит повысить надежность передачи конфиденциальных данных за счет использования общедоступных почтовых сервисов, не прибегая к криптографическим методам. Также эта схема позволяет изучать свойства сети во время мультиплексной передачи данных.

Использование генератора псевдослучаиных чисел для маркирования блоков разделенных данных

При передаче данных с разделением пакетов возникает вопрос о сборке передаваемого сообщения на стороне получателя. Даже если все пакеты получены правильно, необходимо знать порядок соединения полученных пакетов для правильной сборки исходного сообщения. Таким образом, для реализации сеанса передачи порядок пакетного соединения должен быть известен как отправителю, так и получателю.

Для реализации обмена информацией о порядке сборки сообщений между отправителем и получателем можно использовать нумерацию пакетов сообщений, в которой будет указан порядок сборки [16]. Существует несколько вариантов нумерации пакетов (рис. 3):

- прямая нумерация, когда посылке присваивается номер по порядку в качестве идентификатора;
- косвенная нумерация, когда в качестве идентификатора пакету присваивается условное значение (например, контрольная сумма), которое сопоставляется в ключевом файле с номером по порядку;
- присвоение пакету вычисляемого уникального идентификатора, который соответствует номеру в заказе.

При прямой нумерации недостатком является очевидность номера каждого пакета в случае перехвата. Вариант с косвенной нумерацией требует передачи информации о соответствии условного значения порядковому номеру в отдельном ключевом файле или в составе первого пакета сообщений. Для варианта с уникальным идентификатором требуется разработать алгоритм синхронной генерации уникальных идентификаторов от отправителя и получателя соответственно прямая нумерация.

Рис. 3. Варианты нумерации пакетов

Идентификатор должен обладать следующими качествами:

1) уникальностью (как минимум неповторяемостью);
2) отсутствие корреляционной зависимости;
3) воспроизводимость получателем.

Такими свойствами в частности обладают последовательности псевдослучайных чисел, сгенерированные генераторами псевдослучаиных чисел (ГПСЧ) на основе линейно-конгруэнтных алгоритмов [16, 17]. Линейный конгруэнтный метод заключается в вычислении членов линейной рекуррентной последовательности по модулю некоторого натурального числа m, задаваемой следующей формулой:

$$X_{n+1} = (aX_n + c) \bmod m, \qquad (1)$$

где a и c — некоторые целочисленные коэффициенты.

Получаемая последовательность зависит от выбора стартового числа и при разных его значениях получаются различные последовательности случайных чисел. В то же время многие свойства этой последовательности определяются выбором коэффициентов в формуле и не зависят от выбора стартового числа. Линейный конгруэнтный метод порождает статистически хорошую псевдослучайную последовательность чисел, но не является криптографически стойким [18, 19]. Генераторы на основе линейного конгруэнтного метода являются предсказуемыми. Впервые генераторы на основе линейного конгруэнтного метода были взломаны Джимом Ридсом (Jim Reeds), а затем Джоан Бояр (Joan Boyar). Ей удалось также вскрыть квадратические и кубические генераторы. Другие исследователи расширили идеи Бояр, разработав способы вскрытия любого полиномиального генератора. Таким образом, была доказана бесполезность генераторов на основе конгруэнтных методов для криптографии. Однако генераторы на основе линейного конгруэнтного метода сохраняют свою полезность для некриптографических приложений, например, для моделирования [19, 20]. Они эффективны и в большинстве используемых эмпирических тестах демонстрируют хорошие статистические характеристики [20, 21]. Для разрабатываемой системы важна воспроизводимость идентификатора на стороне получателя, что и обусловило выбор данного метода для разработки алгоритма генерации уникальных идентификаторов [20].

Для маркирования пакетов предлагается использовать последовательные выборки длиной n знаков, сделанные через интервал m (a – первая выборка, b – вторая выборка и т.д.) [20].

Таким образом со стороны отправителя мы получаем необходимое количество последовательных уникальных маркеров для отправляемых пакетов информации. Далее необходимо произвести процедуру распознавания идентификаторов со стороны получателя. Для этого предлагается использовать алгоритм генерации одноразовых паролей.

Симметричное разбиение

Алгоритм симметричного разбиения представлен на рисунке 4, ключ для которого может быть, как несимметричным, так и симметричным.

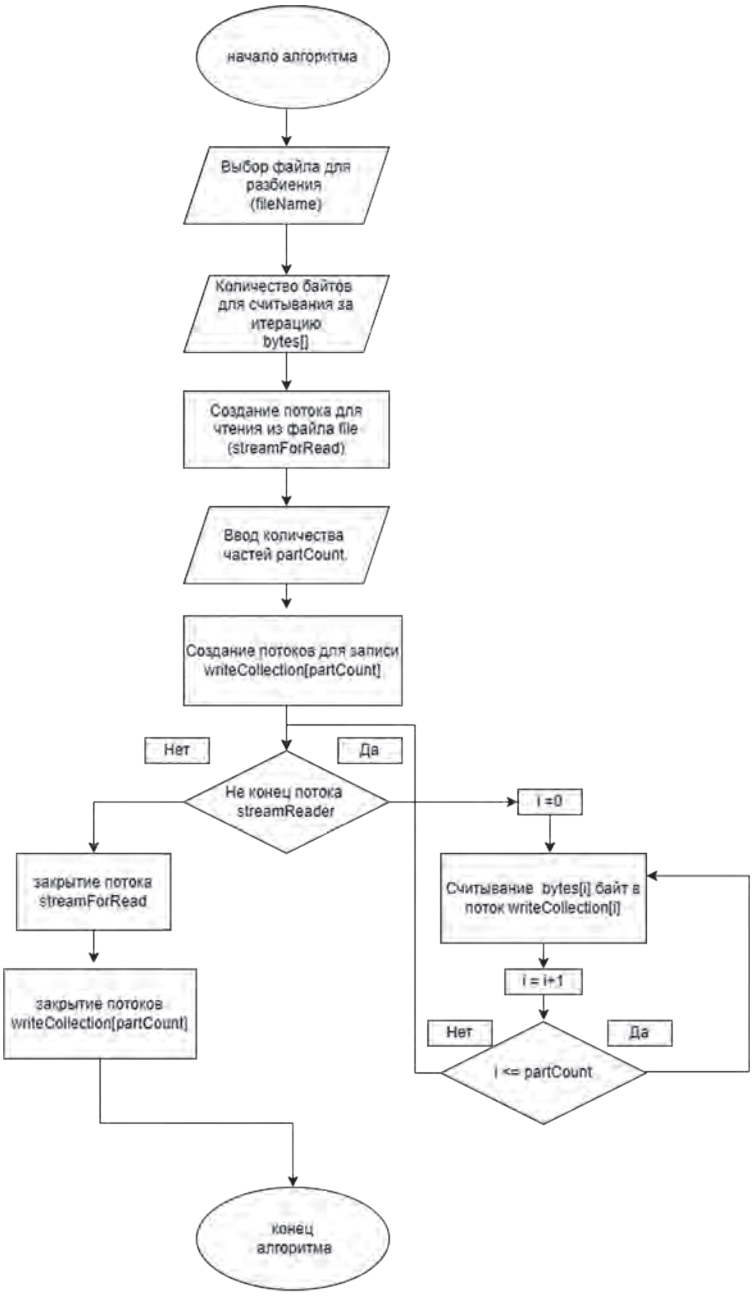


Рис. 4. Блок-схема алгоритма разбиения

На рисунке 5 представлена схема симметричного разбиения.

Данная схема разбиения может использовать части с одинаковым размером. При использовании такого способа разбиения, в случае если злоумышленник перехватит все файлы частей, он сможет легко восстановить передаваемый файл [16, 19, 20]. Суть ее в том, что исходный файл разбивается на несколько кусков одинакового размера (так как в одну итерацию в каждый кусок записывается одинаковое количество байт из исходного файла). Принцип симметричного разбиения представлен на рисунке 6.

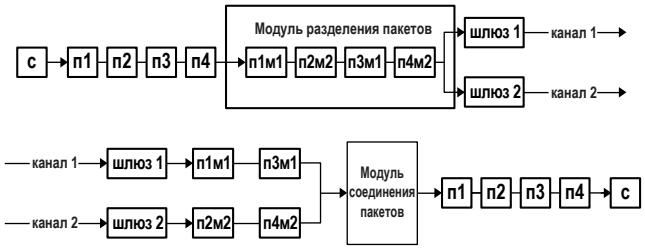


Рис. 5. Схема симметричного разбиения

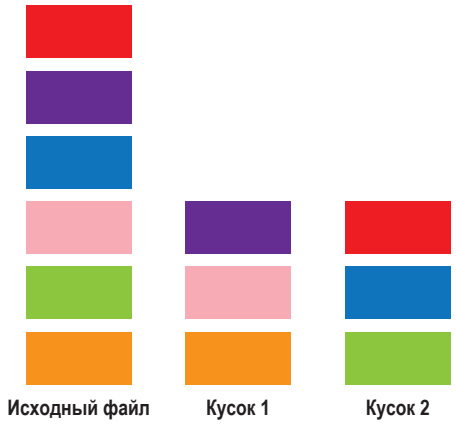


Рис. 6. Принцип симметричного разбиения

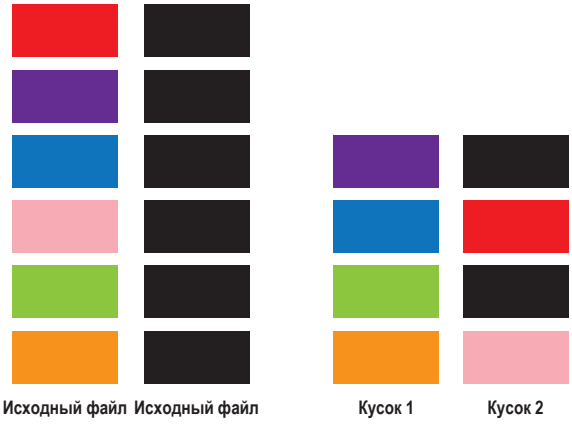


Рис. 7. Несимметричное разбиение с одинаковыми частями

Несимметричное разбиение. Несимметричное разбиение с одинаковыми частями

Суть этого метода состоит в том, чтобы при разбиении использовать модификатор входа хэш-функции («соль – строка данных, которая передаётся хеш-функции вместе с входным массивом данных – прообразом) для вычисления хэша – образа»), для того чтобы усложнить подбор ключа [19]. Атаки утечки хешей паролей приводят к катастрофическим последствиям из-за возможности быстрого подбора [20]. В качестве «соли» мы открываем файл еще в одном потоке и добавляем при забвении «мусорное байты» которые при использовании правильного ключа будут игнорироваться.

На рисунке 7 представлены алгоритмы несимметричного разбиения с одинаковыми частями. Идея метода состоит в том, что исходный файл открывается в 2 потоках. Перед началом разбиения ищется самый большой символ в ключе и в каждую часть на каждой итерации дописывается разность между максимальным символ ключа и текущим. Благодаря этому файлы получаются одинакового размера благодаря чему подобрать ключ возможно только полным перебором вариантов. При сборке файлов лишние байты игнорируются.

Программная реализация

На рисунке 8 и рисунке 9 представлены классы для разбиения файлов (Key и StreamToWrite) Key [19, 20].

Файл Key (рис. 8) представляет собой класс, содержащий в себе, информация о разбиваемом файле:

1. Строковое свойство FileName содержит в себе имя файла для разбиения.
2. Свойство StreamForRead представляет собой поток, открытый на чтение, связный с файлом, предназначенным для разбиения.
3. Свойство WriteCollection представляет собой коллекцию объектов SteramToWrite, содержащих в себе информации о потоках для записи.
4. Массив байт Bytes представляет собой массив байтов полочный из коллекции WriteCollection, содержащий в себе сколько байтов нужно считывать за одну итерацию.
5. Так же класс Key реализует интерфейс IDisposable, а именно метода Dispose в котором происходит закрытие потоков, связанных с файлами.

Файл StreamToWrite (рис. 9) представляет собой класс, содержащий в себе, информацию о файле, куда будут записываться данные:

1. Строковое свойство FileName содержит в себе имя файла.
2. Свойство Stream представляет собой поток, открытый на запись, связный с файлом.
3. Свойство IterationByteSize содержит в себе количество байт сохраняемых в файл за одну итерацию.
4. Массив байт Bytes представляет собой массив байтов, в который сохраняются байты, считанные за итерацию из потока StreamForRead.

Описание интерфейса IFileSystemServices и его реализации

На рисунке 10 представлен интерес IFileSystemServices Key [19, 20] который обязует класс, который его реализует содержать в себе 2 метода:

1. CreateSymmetricKey – Метод создания ключа для симметричного разбиения, который возвращает экземпляр класса KeyDto и принимает 3 параметра:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Newtonsoft.Json;

namespace FileBreaker.Common.Dto
{
    [Serializable]
    public class Key : IDisposable
    {
        public string FileName { get; set; }

        [JsonIgnore]
        public Stream StreamForRead { get; set; }

        [JsonIgnore]
        public IEnumerable<StreamToWrite> WriteCollection { get; set; }

        public int[] Bytes => WriteCollection.Select(w => w.IterationByteSize).ToArray();

        public void Dispose()
        {
            StreamForRead.Dispose();

            foreach (StreamToWrite? streamToWrite in WriteCollection) streamToWrite.Stream.Dispose();
        }
    }
}
```

Рис. 8. – Пример файла key

```
public class StreamToWrite
{
    public string FileName { get; set; }

    public Stream Stream { get; set; }

    public int IterationByteSize { get; set; }

    public byte[] Bytes { get; set; }
}
```

Рис. 9. – Пример файл StreamToWite

```
using FileBreaker.Common.Dto;

namespace FileBreaker.Common.Interfaces
{
    /// <summary> Service for working with the file system. </summary>
    public interface IFileSystemServices
    {
        /// <summary> Method for creating a key for asymmetric partitioning. </summary>
        /// <param name="fileName"> Name of file to split.</param>
        /// <param name="contParts"> Number of parts.</param>
        /// <param name="iterationByteSize"> Number of bytes to read per iteration.</param>
        /// <returns> Parameters for splitting </returns>
        Key CreateAsymmetricKey(string fileName, int contParts, int[] iterationByteSize);

        /// <summary> Method for creating a key for symmetric partitioning. </summary>
        /// <param name="fileName"> Name of file to split.</param>
        /// <param name="contParts"> Number of parts.</param>
        /// <param name="iterationByteSize"> Number of bytes to read per iteration.</param>
        /// <returns> Parameters for splitting </returns>
        Key CreateSymmetricKey(string fileName, int contParts, int iterationByteSize);
    }
}
```

Рис. 10. Интерфейс IFileSystemServices

1) fileName – Имя файла для разбиения;
2) contParts – Количество частей;
3) iterationByteSize – Количество байт для считывания за итерацию (целочисленная переменная).

2. CreateAsymmetricKey – Метод создания ключа для асимметричного разбиения. Который возвращает экземпляр класса KeyDto и принимает 3 параметра:
   1) fileName – Имя файла для разбиения:
   2) contParts – Количество частей:
   3) iterationByteSize – Количество байт для считывания за итерацию(массив).

На рисунках 13 и 14 представлен класс FileSystemServices который реализует интерес IFileSystemServices Key [19, 20]. Методы CreateAsymmetricKey и CreateSymmetricKey вызывают статический метод CreateKey. В методе CreateSymmetricKey создается массив, имеющий размерность contParts и содержащий значение iterationByteSize в каждой ячейке.

Описание работы метода CreateKey

1. Вызывается приватный метод CheckOrCreateFolder (рис. 18) в котором проверяется существование папки, наименование которой совпадает

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using FileBreaker.Common.Dto;
using FileBreaker.Common.Interfaces;
using Newtonsoft.Json;

namespace FileBreaker.Services
{
    ///<inheritdoc/>
    public class FileSystemServices : IFileSystemServices
    {
        #region IFileSystemServices
        ///<inheritdoc/>
        public Key CreateAsymmetricKey(string fileName, int contParts, int[] iterationByteSize)
        {
            return CreateKey(fileName, contParts, iterationByteSize);
        }

        ///<inheritdoc/>
        public Key CreateSymmetricKey(string fileName, int contParts, int iterationByteSize)
        {
            return CreateKey(fileName, contParts, Enumerable.Repeat(iterationByteSize, contParts).ToArray()
        }
        #endregion

        private static Key CreateKey(string fileName, int contParts, int[] iterationByteSize)
        {
            string folderName = Path.GetFileName(fileName);

            CheckOrCreateFolder(folderName);

            Key key = new Key
            {
                FileName = fileName,
                StreamForRead = new FileStream(fileName, FileMode.Open, FileAccess.Read, FileShare.Read),
                WriteCollection = CreateWriteCollection(folderName, contParts, iterationByteSize)
            };

            SaveKey(folderName, key);
            return key;
        }
```

Рис. 11. Класс FileSystemServices реализующий интерфейс IFileSystemServices

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using FileBreaker.Common.Dto;
using FileBreaker.Common.Interfaces;
using Newtonsoft.Json;

namespace FileBreaker.Services
{
    ///<inheritdoc/>
    public class FileSystemServices : IFileSystemServices
    {
        #region IFileSystemServices

        private static Key CreateKey(string fileName, int contParts, int[] iterationByteSize)
        {

        #region private methods
        private static IEnumerable<StreamToWrite> CreateWriteCollection(string fileName, int contParts,
            int[] iterationByteSize)
        {
            List<StreamToWrite> createWriteCollection = new List<StreamToWrite>();

            for (int i = 0; i < contParts; i++)
                createWriteCollection.Add(new StreamToWrite
                {
                    FileName = $"{fileName}/part{i}.part",
                    IterationByteSize = iterationByteSize[i],
                    Bytes = new byte[iterationByteSize[i]],
                    Stream = new FileStream($"{fileName}/part{i}.part", FileMode.Create, FileAccess.ReadWrite,
                        FileShare.None)
                });

            return createWriteCollection;
        }

        private static void SaveKey(string folderName, Key key)
        {
            using (StreamWriter stream = File.CreateText($"{folderName}/.Key.json"))
            {
                JsonSerializer serializer = new JsonSerializer();
                serializer.Serialize(stream, key);
            }
        }

        private static void CheckOrCreateFolder(string fileName)
        {
            if (!Directory.Exists(fileName)) Directory.CreateDirectory(fileName);
        }
        #endregion
    }
}
```

Рис. 12. Класс FileSystemServices реализующий интерфейс IFileSystemServices

Рис. 13. Интерфейс ISplitServicess

Рис. 14. Класс SplitServicess реализующий интерфейс ISplitServices

с файлом для разбиения. Если такой папки нет она будет создана.
2. Выделяется память для экземпляра класса Key.
3. Заполняются все его значения.
4. Экземпляр класса Key серилизуется в JSON файл.
5. Возвращается экземпляр класса Key.

Описание интерфейса ISplitServices и его реализации

На рисунке 13 представлен интерес ISplitServicess который обязует класс, который его реализует содержать в себе метод SplitFile принимающий на вход экземпляр класса Key [19].

На рисунке 14 представлен класс BreakerServices реализующий интерес IBreakerServices. В методе SplitFile в блоке try помещена логика разбиения файла, а в блоке finally помещен вызов метода Dispose() сделано это для того, чтобы закрыть потоки связанные с файлами в любом случае. В методе SplitFile выполняется цикл while который работает пока из файла можно считывать байты и пока текущая позиция в потоке меньше длины потока. Внутри цикла while есть вложенный цикл foreach. В котором на каждой итерации считается массив байт Bytes размерностью IterationByteSize для каждого потока на запись Key [19, 20].

Выводы

В результате проделанной работы предложен метод, который может быть использован для обеспечения безопасности передачи информации, а также для повышения эффективности за счет избыточности современной сетевой инфраструктуры. Приведен пример построения подобной системы передачи данных путем создания защищенных инсталляционных пакетов, что может быть использовано при защите объектов интеллектуальной собственности от несанкционированного использования с использованием при разбиении модификатора входа хэш-функции («соли»). Для маркирования блоков разделенных данных предлагается использовать алгоритм генерации одноразовых паролей на основе генератора псевдослучаиных чисел (ГПСЧ) и линейного конгруэнтного метода.

Подобная технология позволит повысить надежность и безопасность передачи конфиденциальных данных посредством использования общественных почтовых сервисов, не прибегая к криптографическим методам. Также структура позволяет провести исследование свойств сети при мультиплексной передаче данных. Программная реализация вышеописанного инсталлятора позволит создать надежную защиту дистрибутивов программных продуктов от несанкционированной установки. Представленная программа является универсальным средством, позволяющим защищать от несанкционированного использования как программные продукты, так и другие объекты интеллектуальной собственности.

Литература

1. *Schneider M., Shulman H., Sidis A., Sidis R., Waidner M. Diving into Email Bomb Attack. 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 2020. pp. 286-293. DOI: 10.1109/DSN48063.2020.00045.*
2. *Дементьев В. Е., Чулков А. А. Кибервоздействия на протоколы сетей передачи данных // Изв. ТулГУ. Технические науки. 2020. № 10. С. 245–254.*
3. *Maximov R. V., Sokolovsky S. P., Telenga A. P. Model of client-server information system functioning in the conditions of network reconnaissance. CEUR Workshop Proceeding. 2019. pp. 44–51.*
4. *Mvah, F., Kengne Tchendji, V., Tayou Djamegni, C. et al. GaTeBaSep: game theory-based security protocol against ARP spoofing attacks in software-defined networks. Int. J. Inf. Secur. (2023). https://doi.org/10.1007/s10207-023-00749-0*
5. *Stepanov P. P. Attack on the Address Resolution Protocol / Stepanov P. P., Nikonova G. V., Pavlychenko T. S., Gil A. S. // 2020 International Conference Engineering and Telecommunication (En&T), 2020, pp. 1-3.*
6. *Zhang Z., Liu Z., Bai J. Network attack detection model based on Linux memory forensics // Proceedings - 2022 14th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2022. – 14. 2022. – С. 931-935.*
7. *Xia, J.; Cai, Z.; Hu, G.; Xu, M. An Active Defense Solution for ARP Spoofing in OpenFlow Network. Chin. J. Electron. 2019, 28, 172–178.*
8. *Stepanov P. P. The problem of security address resolution protocol / P. P. Stepanov, G. V. Nikonova, T. S. Pavlychenko, A. S. Gil // Journal of Physics: Conference Series. – 2021, Vol. 1791, p.p. 1–8.*
9. *Galal, A. A., Ghalwash, A. Z., Nasr, M. A New Approach for Detecting and Mitigating Address Resolution Protocol (ARP) Poisoning // (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 13, No. 6, 2022. P.337–382.*
10. *Степанов, П. П. Особенности работы протокола разрешения адресов в компьютерных сетях / П. П. Степанов, Г. В. Никонова, Т. С. Павлюченко, В. В. Соловьев // Программная инженерия. – 2022. Том 13, № 5. – С. 211–218.*
11. *Shah Z, Cosgrove S. Mitigating ARP Cache Poisoning Attack in Software-Defined Networking (SDN): A Survey. Electronics. 2019; 8(10):1095. https://doi.org/10.3390/electronics8101095.*
12. *Биджиева С. Х., Шебзухова К. В. Сетевые протоколы передачи данных: преимущества и недостатки // Тенденции развития науки и образования. 2022. Т. 86. № 1. С. 43–45. doi: 10.18411/trnio-06-2022-14.*
13. *Hijazi, S.; Obaidat, M. Address resolution protocol spoofing attacks and security approaches: A survey. Secur. Priv. 2019, 2, e49*
14. *Дементьев В. Е., Чулков А. А. Кибервоздействия на протоколы сетей передачи данных // Изв. ТулГУ. Технические науки. 2020. № 10. С. 245–254.*
15. *Барабошкин Д. А., Бакаева О. А. Анализ алгоритмов шифрования данных // За нами будущее: взгляд молодых ученых на инновационное развитие общества: сб. ст. науч. конф. 2022. Т. 2. С. 449–452.*
16. *Снейдер, И. Эффективное программирование TCP/IP. Библиотека программиста: пер. с анг. – М.: ДМК Пресс. – 2019. – 322 с.*
17. *Барабошкин Д. А., Бакаева О. А. Разработка комбинированного алгоритма шифрования мультимедийных данных в процессе их передачи // Математическое моделирование, численные методы и комплексы программ: сб. тр. X Междунар. науч. молодежн. школы-семинара им. Е. В. Воскресенского. 2022. С. 27–31. URL: https://conf. svmo.ru/files/2022/papers/paper05.pdf (дата обращения: 28.02.2023).*
18. *Mujahid Shah, Sheeraz Ahmed, Khalid Saeed, Muhammad Junaid, Hamayun Khan, Ata-ur Rehman. Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool. // 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019. Sukkur, Pakistan. DOI: 10.1109/ICOMET.2019.8673520.*
19. *Кабанов А. А., Никонова Г. В., Павлюченко Т. С., Степанов П. П. Комплект программ на основе методологии объектно-ориентированного программирования. Свидетельство о регистрации программы для ЭВМ 2020663836, 03.11.2020. Заявка № 2020663293 от 03.11.2020.*
20. *Степанов П. П., Никонова Г. В., Соловьев В. В. Комплект программ для тестирования компьютерных сетей на проникновение. Свидетельство о регистрации программы для ЭВМ 2021661694, 14.07.2021. Заявка № 2021660970 от 14.07.2021.*