Инженерный вестник Дона, №6 (2023)
ivdon.ru/ru/magazine/archive/n6y2023/8462

Разработка приложения для ассоциативной защиты файлов

Р.Ф. Гибадуллин[1], И.С. Вершинин[1], Е.Е. Глебов[2]

[1]Казанский национальный исследовательский технический университет им. А.Н. Туполева–КАИ, Казань, Россия
[2]Акционерное общество «Радиоприбор», Казань, Россия

Аннотация: В современной информационной среде, характеризующейся все возрастающей цифровизацией различных аспектов повседневной жизни, информационная безопасность приобретает первостепенное значение. Множество типов личной информации, включая идентификационные данные, финансовые сведения и медицинские записи, подвергаются хранению в цифровом виде. Организациям необходимо обеспечивать защиту своих интеллектуальных активов, конфиденциальных данных и коммерческой информации от воздействия конкурентов и внутренних угроз. Синергетический подход, заключающийся в объединении криптографии и стеганографии, обеспечивает повышенную сложность анализа передаваемых данных и уменьшает их уязвимость к атакам, основанным на статистическом анализе и других методах выявления закономерностей. Ассоциативная стеганография представляет собой методологию, интегрирующую основные принципы стеганографии и криптографии для обеспечения надежной защиты данных. Разработка программного приложения, предназначенного для ассоциативной защиты файлов, может применяться в широком спектре областей и обладать значительным потенциалом в контексте информационной безопасности. В научной статье рассматриваются предпосылки к созданию данного приложения, приводится описание программного проекта приложения с использованием языка UML (Unified Modeling Language) и анализируются аспекты его реализации. Кроме того, исследуются результаты апробации приложения и предлагается дальнейшая перспектива развития ассоциативной стеганографии.
Ключевые слова: ассоциативная стеганография, стегосообщение, стегостойкость, криптография, информационная безопасность, Unified Modeling Language, исполняющая среда .NET Framework, Windows Presentation Foundation, DeflateStream, BrotliStream, MemoryStream, параллельное программирование.

Введение

Стеганография важна для информационной безопасности, так как стремиться обеспечить неприметный обмен данными. Основная идея стеганографии заключается в том, чтобы маскировать информацию внутри других видов данных, таких, как изображения, аудио, видео, текстовые и бинарные файлы, что позволяет передавать конфиденциальную информацию так, чтобы третьи стороны не заметили этого.

Ассоциативная стеганография – это симбиоз стеганографии и криптографии, изначально разработанный для защиты данных при анализе сцен [1]. Большая часть ранних исследований [2, 3] была нацелена на управление защищенными картографическими базами данных, где каждый k-битовый код (объекта или координат) в результате матричной бинаризации десятичных чисел (γ=10) преобразуется в k-секционный стегоконтейнер. Вначале создается пустой контейнер длиной L=k(9n-12), где n – количество столбцов бинарной матрицы-эталона [2]. Формирование контейнера происходит с использованием псевдослучайной последовательности (ПСП) [4]. После этого биты эталонов (где каждый эталон ставится в соответствие определенной цифре скрываемого кода) встраиваются в позиции контейнера, определяемые набором масок (секретным ключом). Так как любой байт можно представить трехразрядным десятичным числом (k=3), а цифровой контент – последовательностью байтов, ассоциативная стеганография применима в разных областях [5].

В условиях современных вызовов информационной безопасности и растущего объема передаваемых и хранимых данных, актуальность применения стеганографического подхода к защите в существующих программных продуктах усиливается. Ассоциативная стеганография отличается от классических методов стеганографического преобразования, так как обеспечивает практически абсолютную стеганографическую устойчивость, а также повышенную помехозащищенность при хранении и передаче информации через незащищенные коммуникационные каналы в сравнении с общепризнанными криптографическими методами [6].

Актуальность разработки программного модуля для ассоциативной защиты файлов обусловлена несколькими факторами:

– Увеличение объема данных: с ростом объема передаваемых и хранимых данных возрастает вероятность утечки конфиденциальной информации.

– Киберугрозы и хакерские атаки: в современном мире киберугрозы и хакерские атаки становятся все более распространенными [7, 8].
– Высокая стеганографическая стойкость: в отличие от традиционных методов стеганографии [9, 10], ассоциативная защита обеспечивает практически абсолютную стеганографическую стойкость, что делает ее эффективным инструментом противодействия разведывательным атакам.
– Универсальность: ассоциативная стеганография применима для различных типов файлов, таких, как изображения, аудиофайлы, видеофайлы, текстовые документы и бинарные файлы.
– Повышенная помехозащищенность: ассоциативная защита файлов обеспечивает лучшую помехозащищенность при хранении и передаче информации по незащищенным каналам связи, что делает ее привлекательной альтернативой существующим криптографическим методам [11, 12].
– Интеграция с существующими программными продуктами: разработанный модуль может быть интегрирован с существующими программными продуктами, что обеспечивает дополнительный уровень безопасности и удобство использования для пользователей.

Таким образом, актуальность разработки программного модуля для ассоциативной защиты файлов обусловлена растущими вызовами в области информационной безопасности [13, 14], потребностью в эффективных и универсальных методах защиты данных [15, 16], а также стремлением предоставить пользователям высококачественные инструменты для обеспечения конфиденциальности и надежности хранения и передачи информации.

Программный проект

На рисунке 1 представлена диаграмма класса MainWindow, которая представляет собой проект приложения с интерфейсом для генерации секретного ключа, сокрытия и раскрытия файлов на основе ассоциативного механизма защиты.

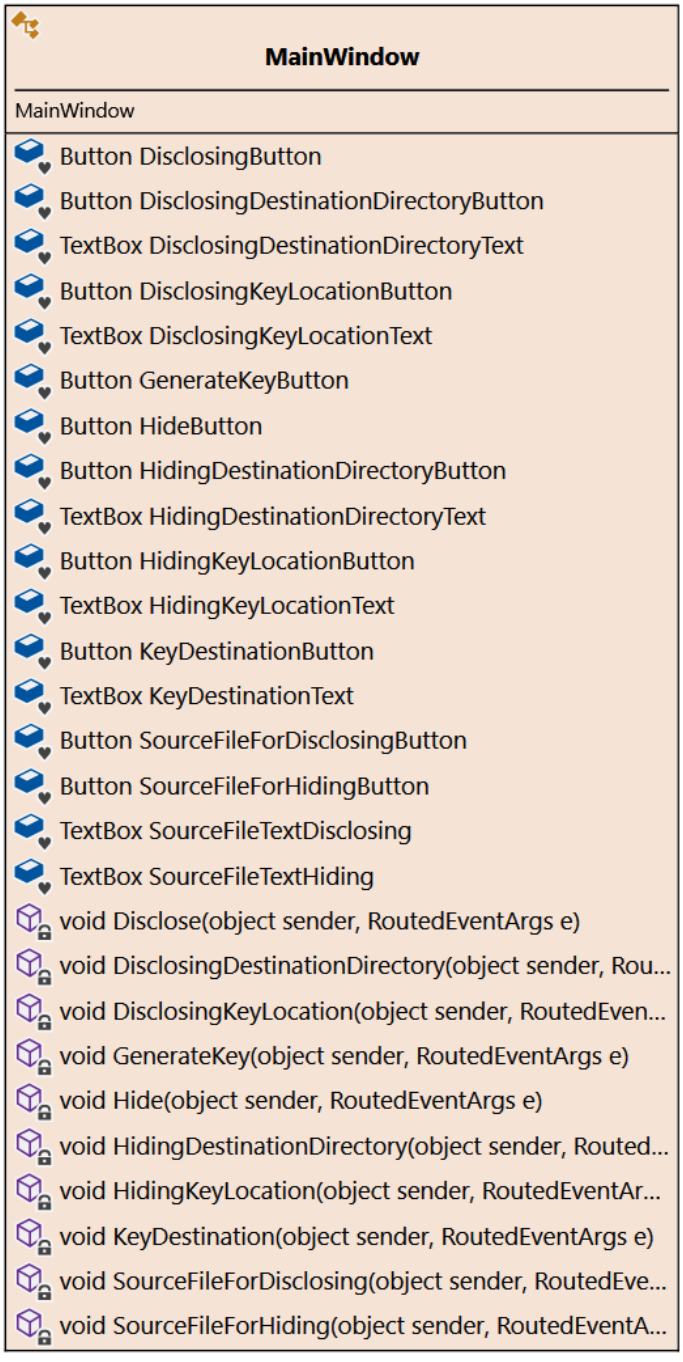

Рис. 1. – Диаграмма класса MainWindow

Проект приложения состоит из трех функциональных частей:

1. Генерация ключа. Кнопка GenerateKeyButton обеспечивает вызов метода GenerateKey для генерации ключа в директории, которая задается в текстовом поле KeyDestinationText посредством кнопки KeyDestinationButton.
2. Сокрытие файла. Кнопка HideButton обеспечивает вызов метода Hide для сокрытия файла, место локализации которого указано в текстовом поле HidingDestinationDirectoryText. Секретный ключ, используемый для сокрытия, указывается в текстовом поле HidingKeyLocationText. Задание пути расположения исходного файла и секретного ключа обеспечивается посредством нажатия на кнопки SourceFileForHidingButton и HidingKeyLocationButton соответственно. Сокрытый файл формируется в директории, которая задается в текстовом поле HidingDestinationDirectoryText посредством кнопки HidingDestinationDirectoryButton.
3. Раскрытие файла. Кнопка DisclosingButton обеспечивает вызов метода Disclose для раскрытия файла, место локализации которого указано в текстовом поле DisclosingDestinationDirectoryText. Секретный ключ, используемый для раскрытия, указывается в текстовом поле DisclosingKeyLocationText. Задание пути расположения сокрытого файла и секретного ключа обеспечивается посредством нажатия на кнопки SourceFileForDisclosingButton и DisclosingKeyLocationButton соответственно. Раскрытый файл формируется в директории, которая задается в текстовом поле DisclosingDestinationDirectoryText посредством кнопки DisclosingDestinationDirectoryButton.

В основу работы методов GenerateKey, Hide, Disclose заложена сборка "stego.exe", представляющая собой консольное приложение без графического

интерфейса. Данная сборка обладает следующей командной функциональностью:

– Генерация ключа:

stego -keygen key.bin

stego -keygen c:\documents\key.bin

– Сокрытие:

stego -hide key.bin file.txt c:\Stego\

stego -hide c:\documents\key.bin c:\documents\file.txt c:\Stego\

stego -hide c:\documents\key.bin c:\documents*.doc c:\Stego\

– Раскрытие:

stego -disclose key.bin file.txt.stego c:\Stego\

stego -disclose c:\documents\key.bin c:\documents\file.txt.stego c:\Stego\

stego -disclose c:\documents\key.bin c:\documents*.stego c:\Stego\

Процедура генерации ключа подробно представлена в работе [17]. Проекты модулей сокрытия и раскрытия файлов представлены в виде диаграмм деятельности на языке UML [18, 19] (см. рисунки 2 и 3).

Области расширения в данных диаграммах представляют собой структурированные узлы деятельности, которые позволяют представлять повторяющиеся или параллельные действия для коллекций элементов. На диаграммах они выглядят как прямоугольники с закругленными углами, внутри которых расположены действия и другие элементы. На двух противоположных сторонах области расширения располагаются маленькие прямоугольники, называемые входными и выходными пинами. Входные пины обозначают коллекцию элементов, которые передаются в область расширения, а выходные пины указывают на результаты, полученные после выполнения действий внутри области расширения для каждого элемента коллекции. Область расширения имеет специальное свойство "mode",

которое указывает, выполняются ли действия параллельно (concurrent) или последовательно (iterative) для каждого элемента коллекции.

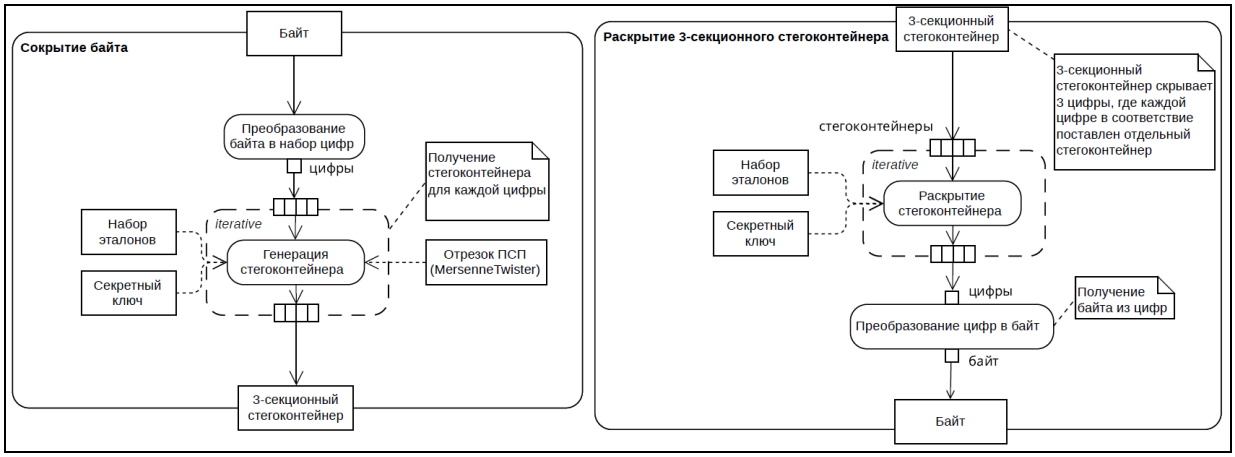

Рис. 2. – Диаграммы деятельности процедур сокрытия байта и раскрытия 3-секционного стегоконтейнера

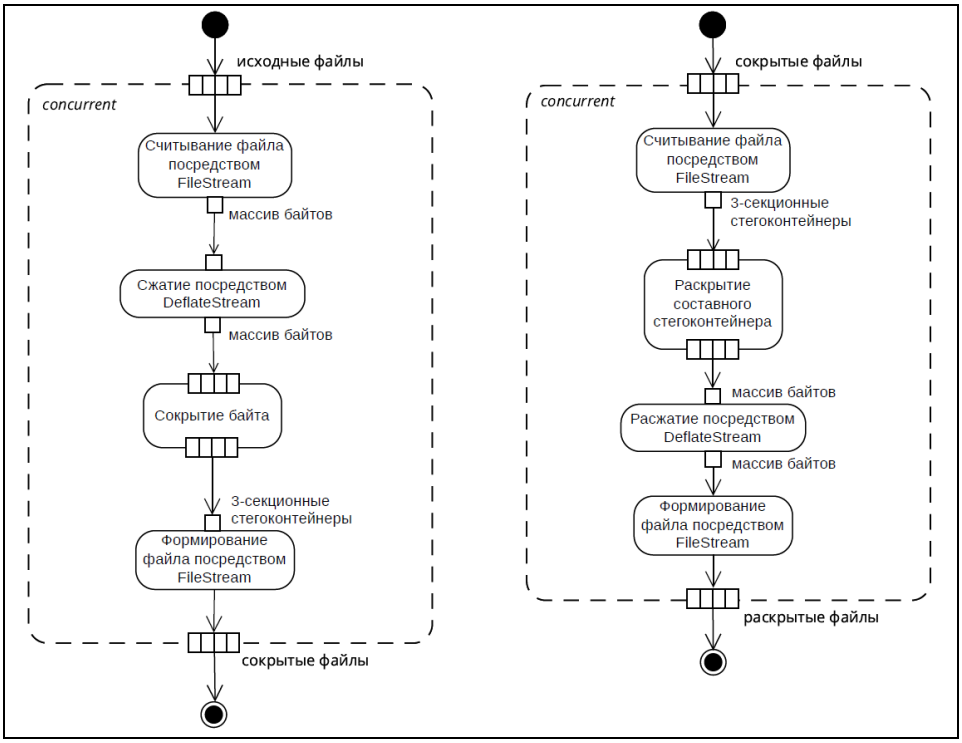

Рис. 3. – Диаграммы деятельности процедур сокрытия и раскрытия файлов

Реализация проекта

Раскроем аспекты программной реализации представленного проекта на базе исполняющей среды .NET Framework с прикладным слоем WPF (Windows Presentation Foundation).

WPF – это фреймворк для разработки настольных приложений на платформе Windows. WPF является частью .NET Framework и позволяет разработчикам создавать графические интерфейсы с высокой степенью кастомизации и разделением логики приложения от представления. WPF использует XAML (eXtensible Application Markup Language) для описания пользовательских интерфейсов [20]. XAML – это язык разметки, позволяющий создавать иерархии объектов и их свойств, что облегчает разработку сложных интерфейсов и обеспечивает чистую разделение логики и представления. Основные возможности WPF включают:

– Разделение логики и представления с использованием паттерна MVVM (Model-View-ViewModel) [21].
– Векторная графика и аппаратное ускорение, что обеспечивает высокое качество отображения и производительность [22].
– Стили, шаблоны и триггеры, которые позволяют создавать настраиваемые и гибкие пользовательские интерфейсы.
– Поддержка анимации и мультимедиа, что делает возможным создание интерактивных и динамических приложений [23, 24].
– Возможность создания пользовательских элементов управления для повторного использования.

Посредством WPF разработана интерфейсная часть приложения (см. рисунок 4), которая обеспечивает взаимодействие пользователя в ключевыми сборками .NET Framework.

В представленной на рисунке 4 вкладке Hiding обеспечивается обращение к сборке "stego.exe" по нажатию на кнопку Hide. Ниже представлен код метода, который будет выполняться в данном случае:

```
var startInfo = new System.Diagnostics.ProcessStartInfo
{
```

```
  FileName = @"data\stego.exe",
  UseShellExecute = false,
  CreateNoWindow = true,
  Arguments = "-hide \"" + @HidingKeyLocationText.Text + "\" \""
+ @SourceFileTextHiding.Text + "\" \"" +
@HidingDestinationDirectoryText.Text + "\""
};
System.Diagnostics.Process proc =
System.Diagnostics.Process.Start(startInfo);
proc.WaitForExit();
System.Windows.MessageBox.Show("Hiding is completed!", "Stego");
```

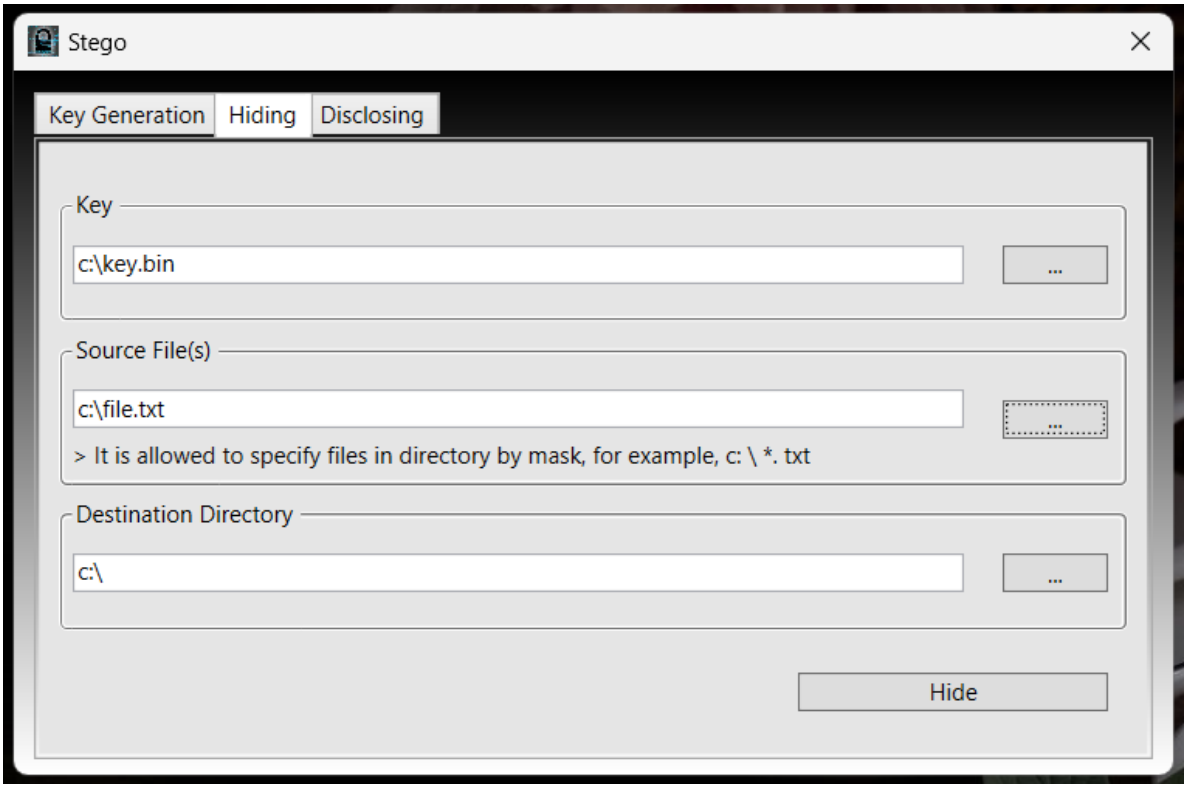

Рис. 4. – Интерфейсная часть приложения

Программный код выполняет следующие действия:

1. Создает объект ProcessStartInfo с именем "startInfo". Этот объект содержит информацию, необходимую для запуска процесса:
   a. FileName: Устанавливает имя исполняемого файла ("stego.exe") в папке "data".

b. UseShellExecute: Устанавливает значение false, указывая, что процесс должен быть запущен непосредственно из исполняемого файла, а не через оболочку операционной системы.

c. CreateNoWindow: Устанавливает значение true, указывая, что при запуске процесса не должно быть видимого окна.

d. Arguments: Устанавливает аргументы командной строки, которые будут переданы исполняемому файлу "stego.exe". В данном случае передаются путь к секретному ключу и директория исходных файлов, которые требуется сокрыть.

2. Запускает процесс, используя информацию из объекта startInfo, и сохраняет ссылку на созданный процесс в переменной proc.
3. Вызывает метод WaitForExit, который блокирует текущий поток до завершения процесса proc.
4. После завершения процесса отображает окно сообщения с текстом "Hiding is completed!" и заголовком "Stego".

Исполняемый файл "stego.exe" получен на основе программного кода с методами:

- `static void keygen(string keyfile)`
- `static string hide(byte c)`
- `static void hidefiles(string keyfile, string sourcefile, string destination)`
- `static byte disclose(string stego)`
- `static void disclosefiles(string keyfile, string sourcefile, string destination)`

Детально с реализацией каждого метода можно ознакомиться в источниках [25, 26]. В рамках данной статьи ограничимся описанием методов hide и hidefiles.

Код программы метода hide:

```
static string Hide(byte c)
```

```
{
  string str = Convert.ToUInt32(c).ToString().PadLeft(3, '0');
  char[] stego = new char[3 * 384];
  MersenneTwister r = new MersenneTwister();
  for (int i = 0; i < 3; i++)
  {
    int h = str[i] - 48;
    for (int k = 0; k < 384; k++)
      if (key[h, k] == '0')
        stego[i * 384 + k] = r.genrand_Int32() % 2 == 0 ? '0' :
'1';
      else
        stego[i * 384 + k] = etalons[h, k];
  }
  return new string(stego);
}
```

Из кода можно заметить, что размер секции стегоконтейнера равен 384, указывающее на то, что n выбран равным 44. Такой выбор n обусловлен сформулированным в диссертации [27] нестрого индуктивным утверждением: формирование стегоконтейнера при k=3 и γ=10 с удовлетворением энтропийного критерия полноты покрытия занимает минимальное время при $40 \le n \le 60$ и $10^5 \le K \le 3\times10^5$. Данное утверждение продиктовано необходимостью снижения времени формирования базы данных сцен с ассоциативной защитой и криптоанализа получаемых сообщений при условии обеспечения безусловной стойкости к перебору случайных ключей. В работе [6] показана допустимость снижения n до 30, но в таком случае необходимый для удовлетворения критерия полноты покрытия объем перебора ключей для подбора подходящего контейнера нарастает. В целях устранения данного недостатка была разработана многопоточная программа генерации стегосообщения, в которой использованы модифицированный безызбыточный алгоритм маскирования

[5] и случайный выбор гаммы. Так, используя кластер с соответствующим числом многоядерных узлов, при n=30 всегда можно сгенерировать стегосообщение, удовлетворяющее требованию безусловной стегостойкости [6].

Код программы метода hidefiles:

```
static void Hidefiles(string @keyfile, string @sourcefile,
string @destination)
{
  Stegomask s = new Stegomask(N);
  s.GetEtalons(out etalons);
  FileStream kf = new FileStream(@keyfile, FileMode.Open);
  if (kf.Length > 400) throw new Exception("Invalid key
detected.");
  BinaryReader br = new BinaryReader(kf);
  int ki = br.ReadInt32();
  int kic = 0;
  for (int i = 0; i < 10; i++)
    for (int j = 0; j < M; j++)
      if (kic < kf.Length / 4 && ki == (i * M + j))
      {
        key[i, j] = '1';
        if (++kic < kf.Length / 4)
          ki = br.ReadInt32();
      }
      else
        key[i, j] = '0';
  kf.Close();
  br.Close();
  if (!sourcefile.Contains("\\"))
    sourcefile = Directory.GetCurrentDirectory() + "\\" +
sourcefile;
  if (@destination[@destination.Length - 1] != '\\')
```

```
    @destination = @destination.Trim('"') + "\\";
  string @location =
System.Reflection.Assembly.GetEntryAssembly().Location;
  string locerr = @location.Remove(@location.LastIndexOf('\\') +
1) + "log.txt";
  StreamWriter sw = new StreamWriter(new FileStream(locerr,
FileMode.Append));

Parallel.ForEach(Directory.GetFiles(@sourcefile.Remove(@sourcefi
le.LastIndexOf('\\') + 1),
  @sourcefile.Substring(@sourcefile.LastIndexOf('\\') + 1)),
(file) =>
  {
    Stopwatch t = new Stopwatch();
    MemoryStream ms = new MemoryStream();
    FileStream fs = new FileStream(@file, FileMode.Open);
    byte[] buf = new byte[fs.Length];
    fs.Read(buf, 0, buf.Length);
    t.Start();
    using (DeflateStream ds = new DeflateStream(ms,
CompressionMode.Compress))
    {
      ds.Write(buf, 0, buf.Length);
    }
    byte[] f = ms.ToArray();
    t.Stop();
    lock ("log")
    {
      sw.WriteLine(@"{4}: File '{0}' size of {1:0.00} KB is
compressed {2:0.000} times. Compression time - {3:0.000} sec.",
                file.Substring(file.LastIndexOf('\\') + 1),
(double)fs.Length / 1024, (double)fs.Length / (double)f.Length,
t.Elapsed.TotalSeconds, DateTime.Now);
```

```
    }
    t.Reset();
    t.Start();
    FileStream hw = new FileStream(@destination +
@file.Substring(@sourcefile.LastIndexOf('\\') + 1) + ".stego",
FileMode.Create);
    int state_cur = 0;
    int state_old = 0;
    Console.Write("Completed 0%");
    for (int i = 0; i < f.Length; i++)
    {
      string stego = Hide(f[i]);
      for (int j = 0; j < stego.Length; j += 8)
        hw.WriteByte(Convert.ToByte(stego.Substring(j, 8), 2));
      state_cur = (int)(i * 100 / f.Length);
      if (state_cur > state_old)
      {
        Console.CursorLeft = 10;
        Console.Write($"{state_cur}%");
        state_old = state_cur;
      }
    }
    hw.Close();
    Console.CursorLeft = 10;
    Console.Write("100%");
    t.Stop();
    lock ("log")
    {
      sw.WriteLine(@"{3}: File '{0}' is successfully hidden. The
size of hidden file is {1:0.00} times larger than the original
one. Concealment time - {2:0.000} sec.",
                file.Substring(file.LastIndexOf('\\') + 1),
(double)new FileInfo(@destination +
```

```
@file.Substring(@sourcefile.LastIndexOf('\\') + 1) +
".stego").Length / (double)fs.Length, t.Elapsed.TotalSeconds,
DateTime.Now);
    }
  });
  sw.Close();
}
```

В данном коде BinaryReader обеспечивает простой и эффективный способ чтения данных типов Int32 (при чтении секретного ключа) и byte (при чтении исходных файлов, подлежащих сокрытию) из двоичных потоков. BinaryWriter обеспечивает запись данных типа byte в двоичный поток (в ходе формирования сокрытого файла).

DeflateStream используется для сжатия данных файла перед их сокрытием. В представленном коде сначала создается экземпляр класса MemoryStream, который далее используется для хранения сжатых данных. Затем создается экземпляр класса DeflateStream, который использует ранее созданный MemoryStream для хранения сжатых данных и настроен на режим сжатия. Массив buf содержит данные файла, которые нужно сжать. После вызова метода Write, данные из buf сжимаются и записываются в ms. После этого, данные из ms используются для последующего сокрытия. Использование DeflateStream позволяет снизить объем передаваемых и хранящихся данных, уменьшая размер файла, который будет скрыт. Это может привести к более эффективному использованию пространства и повышению производительности при обработке больших файлов.

Метод ForEach класса Parallel используется для параллельной обработки (сокрытия) файлов указанной директории (sourcefile). ForEach является методом из пространства имен System.Threading.Tasks, который предоставляет возможность параллельного выполнения итераций цикла foreach для определенной коллекции элементов (списка файлов в нашем

случае). Он автоматически разбивает итерации на части и выполняет их параллельно на разных потоках, обеспечивая более эффективное использование ресурсов многоядерного процессора и повышая скорость сокрытия файлов. Оператор lock используется в данном коде для синхронизации доступа журналу регистрации событий (лог-файлу) между различными параллельными операциями.

Тестирование

Благодаря ведению журнала событий, которое учтено при реализации проекта приложения (см. выше метод hidefiles), в ходе работы программы можно получить следующую информацию:

– дату и время завершения операции,
– размер исходного файла,
– полученный коэффициент сжатия,
– затраченное время на сокрытие или раскрытие файла.

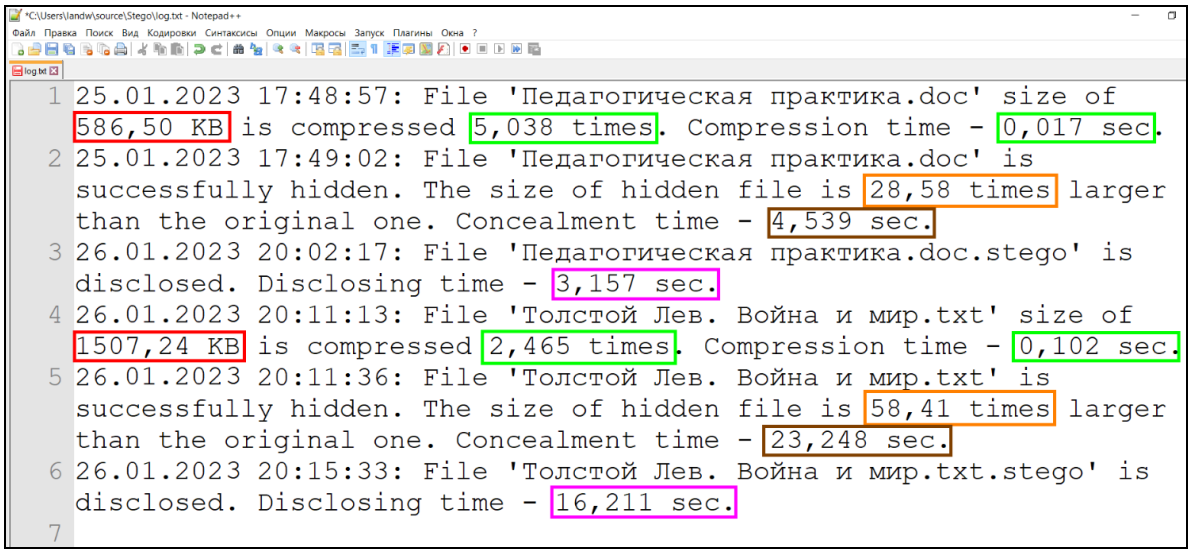

Рис. 5. – Содержимое лог-файла

Содержимое лог-файла (см. рисунок 5) после обработки двух файлов (рабочая программа учебной дисциплины «Педагогическая практика» и роман Льва Николаевича Толстого «Война и мир») показывает, что компрессия позволяет сократить размер файла в несколько раз. Время

сокрытия 600 КБ данных составляет в среднем 23 сек. Процедура раскрытия защищенного файла включает операцию декомпрессии, поэтому время раскрытия в лог-файле ниже времени сокрытия незначительно (время компрессии данных в ходе сокрытия файла регистрируется в журнале отдельно). Размер сокрытого файла превышает размер исходного файла в десятки раз.

Для достижения более высокого коэффициента сжатия можно использовать альтернативный алгоритм сжатия, реализованный в классе BrotliStream [28]. BrotliStream предоставляет более эффективное сжатие по сравнению с DeflateStream, однако, этот выигрыш в степени сжатия сопровождается существенным увеличением времени выполнения алгоритма на этапе сжатия данных, которое может быть в 10 раз и более медленнее. Важно отметить, что замедление производительности заметно только при сжатии данных, в то время как процесс распаковки выполняется с сопоставимой эффективностью по сравнению с DeflateStream.

Заключение

Разработанное приложение позволяет обеспечить сокрытие/раскрытие файлов любого типа посредством ассоциативного механизма защиты, и нашло успешное применение в АО «Радиоприбор» для защиты исходных кодов программ, а также некоторых архивных сведений, представляющих коммерческую ценность. Также приложение может служить обучающим инструментом для студентов и специалистов в области информационной безопасности, демонстрируя принципы ассоциативной стеганографии.

Существенными недостатками разработанного приложения для ассоциативной защиты файлов является низкая скорость сокрытия сравнительно с криптографическими алгоритмами [29] и существенное увеличение выходных данных после стеганографических преобразований. Тем не менее ассоциативный подход обеспечивает высокую скорость

раскрытия сравнительно с процедурой расшифрования в симметричных шифрах, при безызбыточном маскировании обеспечивает правильное санкционированное распознавание при случайном искажении до 3% данных, а при избыточности Q=5 до 6%, где Q – число наборов масок, совокупность которых используется в качестве секретного ключа при распознавании [30]. При большем числе искажений происходит их детектирование (отказ от распознавания). Это существенное преимущество. Не менее существенна повышенная стойкость ассоциативной защиты к действию атак дезинформации при такой избыточности.

Мыслимое дальнейшее развитие приложений ассоциативной стеганографии – защищенная идентификация (с целью выявления подделок) по фиксированному множеству фрагментов архивных документов и библиографических редкостей. Интересен также вопрос применения ассоциативного подхода в целях сокрытия конфиденциальных данных в видеоизображениях.

Литература

1. Duda R.O., Hart P.E., Stork D.G. Pattern classification and scene analysis. New York: Wiley, 1973. V. 3. pp. 731-739.

2. Raikhlin V.A., Vershinin I.S., Gibadullin R.F., Pystogov S.V. Reliable Recognition of Masked Binary Matrices. Connection to Information Security in Map Systems // Lobachevskii Journal of Mathematics, 2013. V. 34, №4. pp. 319-325.

3. Raikhlin V.A., Gibadullin R.F., Vershinin I.S., Pystogov S.V. Reliable recognition of masked cartographic scenes during transmission over the network // 2016 International Siberian Conference on Control and Communications (SIBCON). IEEE, 2016. pp. 1-5.

4. Tian X., Benkrid K. Mersenne Twister Random Number Generation on FPGA, CPU and GPU // 2009 NASA/ESA Conference on Adaptive Hardware and Systems. San Francisco, CA, USA, 2009. pp. 460-464.

5. Raikhlin V.A., Vershinin I.S., Gibadullin R.F. The Elements of Associative Steganography Theory // Moscow University Computational Mathematics and Cybernetics, 2019. V. 43, №1. pp. 40-46.

6. Raikhlin V.A., Gibadullin R.F., Vershinin I.S. Is It Possible to Reduce the Sizes of Stegomessages in Associative Steganography? // Lobachevskii Journal of Mathematics, 2022. V. 43, №2. pp. 455-462.

7. Афанасьева Н.С., Елизаров Д.А., Мызникова Т.А. Классификация фишинговых атак и меры противодействия им // Инженерный вестник Дона, 2022, №5. URL: ivdon.ru/ru/magazine/archive/n5y2022/7641/.

8. Анисимова Г.Б., Грачёв П.В. Проектирование и разработка информационной системы управления заявками для компании оператора мобильной связи // Инженерный вестник Дона, 2022, №5. URL: ivdon.ru/ru/magazine/archive/n5y2022/7660/.

9. Kotkar A., Khadapkar S., Gupta A., Jangale S. Multiple layered Security using combination of Cryptography with Rotational, Flipping Steganography and Message Authentication // 2022 IEEE International Conference on Data Science and Information System (ICDSIS). IEEE, 2022, pp. 1-5.

10. Manohar N., Kumar P.V. Data Encryption & Decryption Using Steganography // 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2020, pp. 697-702.

11. Sultan B., Wani M.A. Multi-data Image Steganography using Generative Adversarial Networks // 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2022. pp. 454-459.

12. Ivasenko M., Suprun O., Suprun O. Information Transmission Protection Using Linguistic Steganography With Arithmetic Encoding And Decoding Approach // 2021 IEEE 3rd International Conference on Advanced Trends in Information Theory (ATIT). IEEE, 2021, pp. 174-178.

13. Khashirova T.Y., Mamuchiev I.I., Mamuchieva M.I., Ozhiganova M.I., Kostyukov A.D., Shumeiko I. Assessment of Information Security in Integrated Systems // 2021 International Conference on Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS). IEEE, 2021. pp. 201-205.

14. Monakhov Y.M., Monakhov M.Y., Telny A.V., Kuznetsova A.P. Prediction of the Information Security State of the Protected Object Using Recurrent Correction // 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). IEEE, 2020. pp. 602-605.

15. Lorence D.P., Churchill R. Incremental adoption of information security in health-care organizations: implications for document management // IEEE Transactions on Information Technology in Biomedicine. V. 9, №2. pp. 169-173.

16. Wang Y., Li C., Cheng N. Internet Security Protection in Personal Sensitive Information // 2014 Tenth International Conference on Computational Intelligence and Security. IEEE, 2014. pp. 628-632.

17. Вершинин И.С. Моделирование двумерно-ассоциативных механизмов маскирования стилизованных бинарных изображений // Диссертация на соискание ученой степени кандидата технических наук: 05.13.18. Казань, 2004. 113 с.

18. Booch G., Jacobson I., Rumbaugh J. The unified modeling language. // Unix Review, 1996. V. 14, №13, P. 5.

19. Fowler M. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.

20. Esch M., Schloss H., Scholtes I. The SEMPA prototype - using XAML and web services for rich interactive Peer-to-Peer applications // 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007). IEEE, 2007. pp. 271-277.

21. Li X., Chang D., Pen H., Zhang X., Liu Y., Yao Y. Application of MVVM design pattern in MES // 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, 2015. Pp. 1374-1378.

22. Xue Y., Ge Y., Zhao J. A practical data structure and algorithm research on drawing and editing vector graphics // 2010 IEEE Youth Conference on Information, Computing and Telecommunications. IEEE, 2010. pp. 33-36.

23. Senthil K.B., Jayasimman L. A study on the user interface design for a Multimedia learning system with emphasis on 3D Animation // IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM - 2012). IEEE, 2012. pp. 110-115.

24. Thuong T.T., Roisin C. An abstract animation model for integrating SMIL basic animation elements with multimedia documents // IEEE International Conference on Multimedia and Expo. IEEE, 2002. V. 2. pp. 297-300.

25. Stego. URL: bitbucket.org/landwatersun/stego/ (Дата обращения: 04.05.2023).

26. Вершинин И.С., Гибадуллин Р.Ф. Программа ассоциативной защиты файлов "Stego" // Свидетельство о государственной регистрации программы для ЭВМ № RU 2021613638. 2021.

27. Гибадуллин Р.Ф. Система баз данных картографии с ассоциативной защитой // Диссертация на соискание ученой степени кандидата технических наук: 05.13.19. Казань, 2011. 117 с.

28. Aher R.N., Pande M. Analysis of Lossless Data Compression Algorithm in Columnar Data Warehouse // 2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA). IEEE, 2022. pp. 1-4.

29. Вершинин И.С., Пыстогов С.В., Гибадуллин Р.Ф., Гашигуллин Д.А. Ассоциативная защита текстовых характеристик объектов // Вестник Казанского государственного технического университета им. А.Н. Туполева, 2020. Т. 76, №1. С. 117-125.

30. Вершинин И.С. Уточнение критерия избыточности помехоустойчивого сокрытия информации в рамках ассоциативной стеганографии // Информация и безопасность, 2016. Т. 19, № 4. С. 511-514.

References

1. Duda R.O., Hart P.E., Stork D.G. New York: Wiley, 1973. V. 3. pp. 731-739.

2. Raikhlin V.A., Vershinin I.S., Gibadullin R.F., Pystogov S.V. Lobachevskii Journal of Mathematics, 2013. V. 34, №4. pp. 319-325.

3. Raikhlin V.A., Gibadullin R.F., Vershinin I.S., Pystogov S.V. 2016 International Siberian Conference on Control and Communications (SIBCON). IEEE, 2016. pp. 1-5.

4. Tian X., Benkrid K. 2009 NASA/ESA Conference on Adaptive Hardware and Systems. San Francisco, CA, USA, 2009. pp. 460-464.

5. Raikhlin V.A., Vershinin I.S., Gibadullin R.F. Moscow University Computational Mathematics and Cybernetics, 2019. V. 43, №1. pp. 40-46.

6. Raikhlin V.A., Gibadullin R.F., Vershinin I.S. Lobachevskii Journal of Mathematics, 2022. V. 43, №2. pp. 455-462.

7. Afanasyeva N.S., Elizarov D.A., Myznikova T.A. Inzenernyj vestnik Dona, 2022, №5. URL: ivdon.ru/ru/magazine/archive/n5y2022/7641/.

8. Anisimova G.B., Grachev P.V. Inzhenernyj vestnik Dona, 2022, №5. URL: ivdon.ru/ru/magazine/archive/n5y2022/7660/.

9. Kotkar A., Khadapkar S., Gupta A., Jangale S. 2022 IEEE International Conference on Data Science and Information System (ICDSIS). IEEE, 2022, pp. 1-5.

10. Manohar N., Kumar P.V. 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2020, pp. 697-702.

11. Sultan B., Wani M.A. 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2022. pp. 454-459.

12. Ivasenko M., Suprun O., Suprun O. 2021 IEEE 3rd International Conference on Advanced Trends in Information Theory (ATIT). IEEE, 2021, pp. 174-178.

13. Khashirova T.Y., Mamuchiev I.I., Mamuchieva M.I., Ozhiganova M.I., Kostyukov A.D., Shumeiko I. 2021 International Conference on Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS). IEEE, 2021. pp. 201-205.

14. Monakhov Y.M., Monakhov M.Y., Telny A.V., Kuznetsova A.P. 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). IEEE, 2020. pp. 602-605.

15. Lorence D.P., Churchill R. IEEE Transactions on Information Technology in Biomedicine. V. 9, №2. pp. 169-173.

16. Wang Y., Li C., Cheng N. 2014 Tenth International Conference on Computational Intelligence and Security. IEEE, 2014. pp. 628-632.

17. Vershinin I.S. Dissertatciia na soiskanie uchenoi stepeni kandidata tekhnicheskikh nauk [Dissertation for the degree of Candidate of Technical Sciences]: 05.13.18. Kazan, 2004. 113 p.

18. Booch G., Jacobson I., Rumbaugh J. Unix Review, 1996. V. 14, №13, P. 5.

19. Fowler M. Addison-Wesley Professional, 2004.

20. Esch M., Schloss H., Scholtes I. 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007). IEEE, 2007. pp. 271-277.

21. Li X., Chang D., Pen H., Zhang X., Liu Y., Yao Y. 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, 2015. pp. 1374-1378.

22. Xue Y., Ge Y., Zhao J. 2010 IEEE Youth Conference on Information, Computing and Telecommunications. IEEE, 2010. pp. 33-36.

23. Senthil K.B., Jayasimman L. IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012). IEEE, 2012. pp. 110-115.

24. Thuong T.T., Roisin C. IEEE International Conference on Multimedia and Expo. IEEE, 2002. V. 2. pp. 297-300.

25. Stego. URL: bitbucket.org/landwatersun/stego/ (date accessed: 04.05.2023).

26. Vershinin I.S., Gibadullin R.F. Svidetelstvo o gosudarstvennoi registratcii programmy dlia EVM [Certificate of state registration of computer programs]. № RU 2021613638. 2021.

27. Gibadullin R.F. Dissertatciia na soiskanie uchenoi stepeni kandidata tekhnicheskikh nauk [Dissertation for the degree of Candidate of Technical Sciences]: 05.13.19. Kazan, 2011. 117 p.

28. Aher R.N., Pande M. 2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA). IEEE, 2022. pp. 1-4.

29. Vershinin I.S., Pystogov S.V., Gibadullin R.F., Gashigullin D.A. Vestneyk Kazanskogo gosudarstvennogo tekhnicheskogo universiteta im. A.N. Tupoleva. 2020. V. 76, №1. pp. 117-125.

30. Vershinin I.S. Informatciia i bezopasnost. 2016. V. 19, № 4. pp. 511-514.