**ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ / INFORMATICS AND INFORMATION PROCESSES**

**DOI: https://doi.org/10.60797/IRJ.2025.152.4**

# КРИПТОГРАФИЧЕСКИЙ МОДУЛЬ ЗАЩИТЫ ФАЙЛОВ ГЕОИНФОРМАЦИОННОЙ СИСТЕМЫ MAPINFO С ПРИМЕНЕНИЕМ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ И ПАРАЛЛЕЛЬНОЙ БИБЛИОТЕКИ ПЛАТФОРМЫ .NET

Научная статья

**Яковлев И.В.[1], Пукита М.Г.[2,] ***

[1, 2] Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация

* Корреспондирующий автор (pukita02[at]mail.ru)

**Аннотация**

В статье представлено исследование, направленное на разработку программного модуля для криптографической защиты файлов геоинформационной системы MapInfo. Основное внимание уделено применению регулярных выражений для идентификации и шифрования конфиденциальной информации, а также параллельным вычислениям на базе библиотеки платформы .NET, что обеспечивает высокую производительность. Предложенное решение позволяет осуществлять избирательное шифрование ценных данных, таких как координаты, минимизируя затраты ресурсов и времени. Проведенный анализ существующих решений выявил их недостаточную гибкость и неспособность обрабатывать данные частично, что подчеркивает новизну подхода. Тестирование показало, что параллельная реализация модуля значительно превосходит последовательную, особенно при обработке больших объемов данных, а использование алгоритмов AES и ГОСТ «Кузнечик» обеспечивает высокий уровень криптографической стойкости. Разработанный модуль обладает практической значимостью для организаций, работающих с конфиденциальной геопространственной информацией, и демонстрирует перспективы для дальнейшего масштабирования и интеграции в существующие системы.

**Ключевые слова:** криптографическая защита, геоинформационные системы, MapInfo, регулярные выражения, параллельные вычисления, .NET, шифрование, AES, ГОСТ «Кузнечик», производительность, конфиденциальность данных.

# CRYPTOGRAPHIC MODULE OF MAPINFO GEOGRAPHIC INFORMATION SYSTEM FILE PROTECTION USING REGULAR EXPRESSIONS AND .NET PLATFORM PARALLEL LIBRARY

Research article

**Iakovlev I.V.[1], Pukita M.G.[2,] ***

[1, 2] Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation

* Corresponding author (pukita02[at]mail.ru)

**Abstract**

The article presents research aimed at developing a software module for cryptographic protection of MapInfo geographic information system files. The main attention is paid to the application of regular expressions for identification and encryption of confidential information, as well as parallel computing based on the .NET platform library, which provides high performance. The proposed solution allows selective encryption of valuable data, such as coordinates, minimising resource and time costs. Analyses of existing solutions revealed their lack of flexibility and inability to process data partially, highlighting the novelty of the approach. Testing has shown that the parallel implementation of the module significantly outperforms the serial one, especially when processing large amounts of data, and the use of AES and GOST "Grasshopper" algorithms provides a high level of cryptographic strength. The developed module has practical significance for organizations working with sensitive geospatial information and demonstrates prospects for further scaling and integration into existing systems.

**Keywords:** cryptographic protection, geographic information systems, MapInfo, regular expressions, parallel computing, .NET, encryption, AES, "Grasshopper", performance, data privacy.

## Введение

В последние годы наблюдается растущий интерес к разработке технологий защиты данных, применимых в специфических областях, таких как геоинформационные системы (ГИС). Современные исследования в данной области подтверждают необходимость создания узкоспециализированных инструментов, способных обеспечивать избирательную защиту данных в условиях высокой производительности. Например, в работе [1], [2] рассмотрены подходы к шифрованию данных в облачных средах, однако они не адаптированы для работы с форматом данных ГИС, что затрудняет их практическое применение.

Кроме того, в публикациях [3], [4] обсуждаются криптографические алгоритмы, такие как AES и Serpent, используемые в современных системах. Однако эти работы фокусируются на защите данных в целом и не учитывают необходимость избирательного шифрования, что критически важно для приложений, работающих с большими объёмами геопространственной информации. Такой подход приводит к значительным временным и ресурсным затратам.

Исследования, связанные с использованием регулярных выражений для обработки текстовых данных, также находят широкое применение в различных областях, например, в валидации данных [5] и обработке текстовых файлов

[6]. Тем не менее, их потенциал для задач шифрования данных в ГИС, особенно в сочетании с параллельными вычислениями [7], [8], остаётся недостаточно исследованным.

Предложенный в настоящей работе подход ориентирован на устранение этих недостатков. Он сочетает в себе высокую производительность параллельных вычислений на платформе .NET с возможностью точечного шифрования данных, определённых с помощью регулярных выражений. Такой подход позволяет минимизировать объём зашифрованных данных, обеспечивая при этом конфиденциальность и целостность наиболее ценной информации, например, координатных данных. По сравнению с существующими решениями, данный подход предлагает уникальную комбинацию технологий, что подчёркивает его актуальность.

Целью работы является разработка программного модуля криптографической защиты файлов ГИС MapInfo с использованием регулярных выражений и параллельной библиотеки платформы .NET. Для достижения данной цели решаются задачи: исследование существующих подобных решений; изучение ГИС MapInfo и ее файлового формата; изучение криптографических методов защиты данных; разработка программного модуля; тестирование разработанного модуля; оценка производительности.

Новизна данной разработки заключается в объединении современных методов шифровая, регулярных выражений и параллельной библиотеки платформы .NET, что позволяет обеспечить надежную защиту файлов и повысить производительность модуля.

Созданный в процессе исследования программный модуль практически значимо для организаций, использующих ГИС MapInfo для хранения конфиденциальной информации, поскольку он поможет эффективно защищать данные от несанкционированного доступа, обеспечивая конфиденциальность и целостность геопространственных данных.

**Существующие решения**

В рамках данной работы разрабатывается не отдельное приложение с графическим пользовательским интерфейсом, а программная библиотека, которая предназначена для последующей интеграции в другие программные продукты. Несмотря на это, рассмотреть существующие аналоги все же необходимо для лучшего понимания предметной области и обоснования актуальности разработки.

VeraCrypt [9] – бесплатное приложение с открытым исходным кодом для шифрования данных на основе TrueCrypt. Оно позволяет создавать зашифрованные контейнеры, а также шифровать целые диски и разделы. VeraCrypt использует надежные алгоритмы шифрования, такие как AES, Serpent, Twofish, позволяя применять несколько слоев шифрования. Приложение поддерживает Windows, macOS и Linux, может работать в портативном режиме. Однако оно довольно сложно в использовании для начинающих, не имеет официальной технической поддержки, а шифрование/расшифровка может занимать много времени и ресурсов.

NordLocker [10] – коммерческое приложение от компании NordVPN для безопасного хранения и обмена файлами с использованием алгоритма AES-256. Оно предоставляет простой интуитивно понятный интерфейс, интегрируется с облачными сервисами (Dropbox, Google Drive, OneDrive), позволяя шифровать файлы в облаке. Кроме шифрования, NordLocker предлагает функции блокировки, скрытия файлов и создания защищенных папок. Недостатками являются необходимость покупки лицензии, возможные опасения по конфиденциальности при использовании облака и ограниченная гибкость настроек.

AxCrypt [11] – приложение с открытым исходным кодом для шифрования файлов, совместимое с Windows, macOS, Linux и мобильными устройствами. Оно имеет простой интерфейс, поддерживает AES, Blowfish, интегрируется с облачными хранилищами. Основными недостатками являются ограниченный функционал бесплатной версии и тот факт, что, как и другие рассмотренные приложения, AxCrypt шифрует файлы целиком, не позволяя зашифровать только определенные данные внутри файла.

Анализ существующих решений показал, что они не обеспечивают требуемую функциональность целевого шифрования конфиденциальных данных (например, координат) внутри файлов ГИС MapInfo, оставляя остальную информацию открытой. На данный момент, хоть и имеется множество приложений, позволяющих шифровать файлы и даже дисковые пространства, необходимость в данной разработке все же присутствует. Эта необходимость обеспечивается из самой реализации модуля, ведь на выходе будет иметься файл, который не будет зашифрован полностью, как в данных приложениях. В данном файле будут шифроваться только определенные данные, например, координаты, которые будут найдены с помощью регулярных выражений.

Таким образом, обзор аналогов помог определить практическую значимость и новизну разрабатываемой библиотеки, которая будет предоставлять уникальную функциональность целевого шифрования конфиденциальных данных в файлах ГИС с использованием регулярных выражений и параллельных вычислений.

**Использование регулярных выражений для криптографической защиты файлов ГИС MapInfo**

Регулярные выражения (regular expressions) – инструмент, который широко используется для обработки текстовых данных. Они обеспечивают гибкий и эффективный подход манипулирования подстроками в тексте [12].

Регулярное выражение представляет последовательность символов, определяющих шаблон для дальнейшего поиска. Эти шаблоны могут быть простыми, могут быть сложными, но, несмотря на это, они обеспечивают точное соответствие необходимым текстовым данным. К примеру, простое регулярное выражение может состоять лишь из буквенных символов, которые должны присутствовать в искомой строке. Более сложные же выражения включают в себя метасимволы: подстановочные знаки и квантификаторы. Метасимволы служат прототипом для определенных типов символов или последовательностей в строке. Например, метасимвол точка «.» представляет любой символ, в то время как квантификатор звездочка «*» говорит о том, что предыдущий символ (группа символов) может присутствовать в искомой подстроке нуль или более раз. Комбинируя различные метасимволы с обычными символами, можно создавать сложные шаблоны для поиска специфичной информации.

Бывает такое, что одна и та же информация в различных источниках может приобретать различный вид. Взять даже номера телефонов, существует несколько различных вариантов их написания. В регулярных выражениях есть возможность указания нескольких вариантов сопоставления с определенным шаблоном с помощью символа «|». Данным символом разделяются все возможные варианты для поиска. Также в регулярных выражениях имеется возможность группирования, которая позволяет пользователям создавать шаблоны внутри выражения, заключая их в круглые скобки (). Эта функциональность не только упорядочивает сложные шаблоны, но и облегчает целенаправленное извлечение информации из совпадающих подстрок.

В регулярных выражениях также имеется возможность отбора подстрок на основе информации, предшествующей этой подстроке или следующей после нее. Такая функциональность называется положительным/отрицательным просмотром вперед или назад. Это позволяет пользователям фильтровать совпадения и фокусироваться лишь на конкретном содержимом в текстовых данных.

Регулярные выражения предоставляют мощный и гибкий инструмент для работы со строками, позволяя выполнять такие операции, как: поиск подстрок, валидация данных (с помощью регулярных выражений можно проверять, соответствует ли определенная строка заданному шаблону, например, проверка корректности электронного адреса или номера телефона), извлечение данных, замена текста, разбиение текста.

Преимущества использования регулярных выражений:

- точная фильтрация данных на основе различных шаблонов поиска;
- можно создавать сложные шаблоны для поиска, шифрования, маскирования и проверки целостности данных;
- алгоритмы поиска на основе регулярных выражений оптимизированы для работы с большими объемами данных;
- существует множество библиотек и инструментов для работы с регулярными выражениями в различных языках программирования;
- при внесении изменений легко исправить только регулярное выражение, не меняя логику функционирования всего приложения.

В контексте обработки текстовых данных в разрабатываемом программном модуле, регулярные выражения будут использоваться для поиска координат или другой конфиденциальной информации для последующего шифрования криптографическими алгоритмами, рассмотренными в предыдущей главе. Так как в файлах содержатся не только координатные данные, можно значительно ускорить процесс обработки файлов. В данной реализации шифруются данные, которые соответствуют регулярным выражениям, остальная информация при этом остается неизменной, что может существенно сократить время обработки.

Использование регулярных выражений в сочетании с криптографической защитой в ГИС MapInfo может обеспечить гибкую и производительную обработку текстовых данных, одновременно повышая безопасность и конфиденциальность этих данных.

**Параллельное программирование в рамках разрабатываемого модуля**

В условиях современного роста объемов обрабатываемых данных производительность программного обеспечения становится одним из решающих факторов успеха. Один из наиболее эффективных методов увеличения производительности заключается в использовании параллельного программирования, которое позволяет распределить вычислительные задачи между несколькими потоками или процессорами. Параллельное программирование представляет собой область разработки программного обеспечения, направленную на создание программ, которые могут выполнять вычисления одновременно на нескольких ядрах процессора или различных устройствах. Важнейшими понятиями в этой области являются поток, процесс, синхронизация, распределение задач и пул потоков. Поток является базовой единицей параллельного выполнения, и он может независимо выполнять часть программы. Процесс же представляет собой отдельную программу, которая выполняется в операционной системе и может содержать несколько потоков. Синхронизация позволяет координировать выполнение потоков и предотвращать возможные конфликты при совместном доступе к общим ресурсам [13], [14]. Распределение задач подразумевает разделение программы на независимые задачи, которые могут выполняться параллельно, тем самым ускоряя выполнение программы. Пул потоков – это механизм управления множеством потоков, позволяющий повторно использовать потоки, что помогает избежать излишней нагрузки на систему.

Основная цель параллельного программирования заключается в эффективном использовании вычислительных ресурсов, что особенно важно для задач, требующих обработки больших объемов данных или выполнения сложных вычислений. В то же время параллельное программирование сталкивается с рядом вызовов, таких как необходимость разработки специализированных подходов для синхронизации потоков, управления гонками данных и балансировки нагрузки. Также стоит отметить сложности отладки и тестирования, поскольку поведение параллельных программ может быть непредсказуемым из-за взаимодействия потоков. Еще одной проблемой является потребность в дополнительных ресурсах для синхронизации и обмена данными между потоками, что может негативно сказаться на производительности.

В рамках данного исследования была выбрана многопоточная модель параллельного программирования, что позволяет выполнять операции одновременно на нескольких ядрах процессора. Для реализации параллельного программирования на платформе .NET используется библиотека Parallel LINQ (PLINQ), которая автоматически распределяет вычислительные задачи между доступными потоками или процессорами. PLINQ предоставляет возможности для автоматического распараллеливания запросов, таких как Where, Select, OrderBy, и может масштабироваться на любое количество доступных процессоров. Это упрощает решение проблем синхронизации потоков и гонок данных, позволяя более эффективно использовать ресурсы системы [15].

Также для безопасного доступа к данным в многопоточной среде используются конкурентные коллекции, предоставляемые пространством имен System.Collections.Concurrent. Эти коллекции, такие как ConcurrentBag<T>, ConcurrentDictionary<TKey, TValue>, ConcurrentQueue<T> и ConcurrentStack<T>, обеспечивают безопасную работу с

данными в многопоточном окружении, минимизируя блокировки и оптимизируя производительность при работе с данными.

В рамках разрабатываемого модуля криптографической защиты файлов используется коллекция ConcurrentBag<T> [16] для хранения набора данных, содержащих зашифрованные фрагменты текста и их соответствие регулярным выражениям. Применение данной коллекции позволяет обеспечить безопасность доступа к данным из нескольких потоков, минимизировать блокировки за счет эффективных механизмов синхронизации и поддерживать высокую скорость обработки данных в многопоточной среде.

Таким образом, использование параллельного программирования и конкурентных коллекций в данном программном модуле позволяет существенно повысить производительность, улучшить корректность обработки данных и эффективно использовать ресурсы системы. Библиотека PLINQ и коллекция ConcurrentBag демонстрируют свою эффективность при решении задач, связанных с шифрованием больших объемов данных, и позволяют масштабировать систему для дальнейшего увеличения производительности.

**Методология разработки модуля**

При проектировании программного модуля криптографической защиты файлов ГИС MapInfo были определены основные технологические решения, обеспечивающие баланс между производительностью, гибкостью и безопасностью. Основное внимание уделено выбору алгоритмов и подходов, которые наилучшим образом соответствуют задачам обработки данных в геоинформационных системах.

Регулярные выражения (Regular Expressions) выбраны в качестве основного инструмента для идентификации конфиденциальной информации в файлах ГИС MapInfo. Этот выбор обусловлен следующими факторами:

- Регулярные выражения предоставляют мощный механизм для описания сложных шаблонов данных, таких как координаты, даты или специфические текстовые фрагменты. Например, для идентификации координат в формате `X Y` использовалось выражение `-?\d+(\.\d+)?\s-?\d+(\.\d+)?`, которое охватывает как целые числа, так и дробные значения с возможным отрицательным знаком.

- Современные реализации регулярных выражений, такие как в .NET, оптимизированы для работы с большими текстовыми массивами, что делает их подходящими для обработки файлов большого размера.

Регулярные выражения легко адаптировать под изменения в формате данных, что упрощает обслуживание и расширение функционала модуля. Пример использования: регулярные выражения применялись для выборки координатных данных из файлов формата MIF, где каждый текстовый блок содержал множество элементов, не представляющих интереса для шифрования (см. рисунок 1).

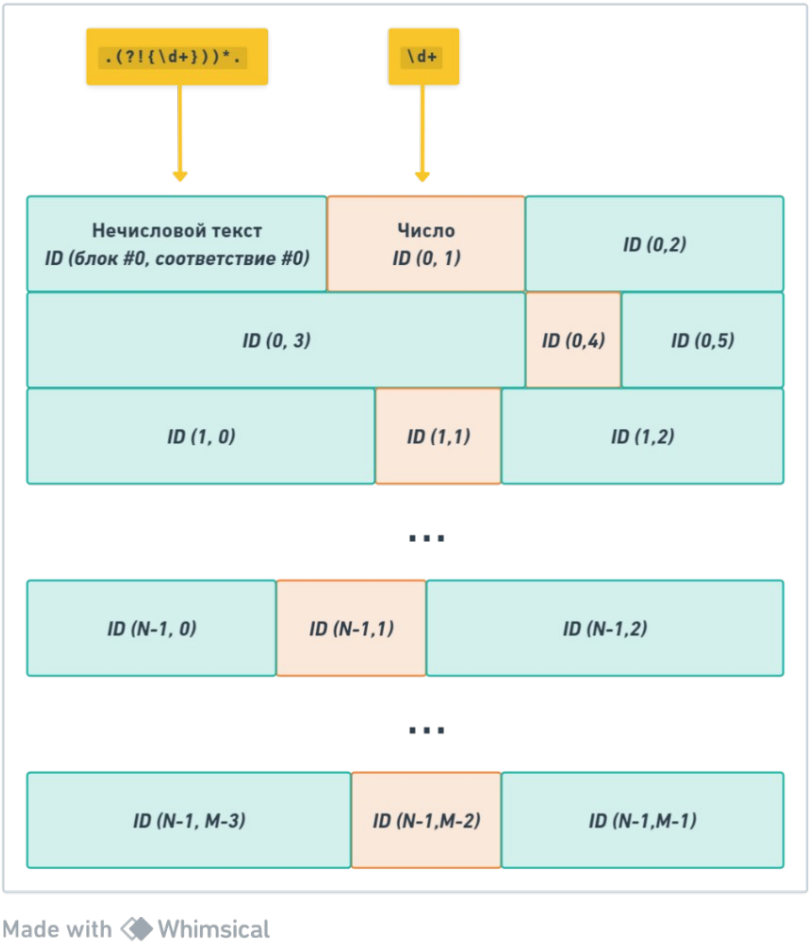



Рисунок 1 - Поиск соответствий для последующего шифрования
DOI: https://doi.org/10.60797/IRJ.2025.152.4.1

Многопоточность реализована с использованием библиотеки Parallel LINQ (PLINQ) в платформе .NET, что позволяет параллельно обрабатывать фрагменты данных и оптимизировать использование вычислительных ресурсов. PLINQ автоматически распределяет задачи между доступными ядрами процессора, что особенно важно при обработке больших файлов с большим количеством координатных данных. Для работы в многопоточном окружении использовались конкурентные коллекции, такие как ConcurrentBag<T>. Это обеспечило потокобезопасное хранение обработанных данных без необходимости использования сложных механизмов синхронизации. Пример использования: текстовый файл разбивался на блоки с помощью функции SplitText, а затем каждый блок обрабатывался параллельно. Для каждого блока осуществлялся поиск координат с помощью регулярных выражений, зашифровка совпадений и их сохранение в структуре CipherDataSet.

Геоинформационные системы, такие как MapInfo, требуют гибкости в обработке данных, поскольку они содержат различные типы информации, например координаты, описания объектов и метаданные. Применение регулярных выражений и многопоточности позволило достичь следующих целей:

- Избирательное шифрование данных, что минимизировало нагрузку на систему.
- Ускорение обработки больших объемов данных благодаря распараллеливанию задач.
- Гибкость настройки поиска и обработки данных, что делает модуль универсальным для различных форматов файлов ГИС.

Такая методология обеспечила высокую производительность модуля при сохранении простоты его интеграции в систему MapInfo.

**Разработка программного модуля**

Программный модуль состоит из нескольких классов, реализующих алгоритм шифрования и дешифрования текстовых данных с использованием российского криптографического алгоритма «Кузнечик» (ГОСТ Р 34.12-2015) [17] и стандартного алгоритма AES.

Основной класс Program содержит метод Main, который является точкой входа в приложение.

Класс KuznyechikStream реализует поток для шифрования/дешифрования данных с использованием алгоритма «Кузнечик». Он наследуется от базового класса Stream и реализует методы чтения, записи и преобразования данных с применением шифрования/дешифрования.

Класс Kuznyechik реализует сам алгоритм «Кузнечик». Он содержит методы для генерации подключей, шифрования и дешифрования блоков данных, а также вспомогательные методы для выполнения операций над битовыми векторами.

Программа также использует стандартные классы AES и CryptoStream для работы с алгоритмом шифрования AES.

Архитектура модуля построена таким образом, что позволяет работать с различными алгоритмами шифрования (в данном случае Кузнечик и AES), а также обрабатывать большие объемы текстовых данных за счет использования параллельных вычислений и потоковой обработки данных.

Функциональная часть программного модуля обеспечивается выполнением следующих шагов:

1. Импортируются необходимые пространства и инициализируются объекты, которые будут использоваться для реализации механизма криптографической защиты данных.

2. Создается экземпляр ConcurrentBag<CipherDataSet> для хранения фрагментов текста и метаданных и экземпляр класса ThreadLocal<Kuznyechik> или ThreadLocal<ICryptoTransform> для хранения криптографических объектов в потоках.

3. Считывает исходный текст из файла.

```
string text;
using (Stream fs = File.OpenRead(input_path))
{
  using (StreamReader reader = new StreamReader(fs))
  {
    text = reader.ReadToEnd();
  }
}
```

1. Исходный текст, с помощью заданного разделителя и минимального размера блока, разбивается на блоки с помощью метода SplitText.

```
char[] separators = new[] { '\n' };
const int blockSize = 7;
var textBlocks = SplitText(text, separators, blockSize);
static List<string> SplitText(string text, char[] delimiters, int minBlockSize)
{
  var blocks = new List<string>();
  var currentBlock = new StringBuilder();
  foreach (var character in text)
  {
    currentBlock.Append(character);
    // Проверяем, является ли текущий символ одним из разделителей и достигнут ли минимальный размер блока
    if (delimiters.Contains(character) && currentBlock.Length >= minBlockSize)
    {
      blocks.Add(currentBlock.ToString());
      currentBlock.Clear();
    }
  }
  if (currentBlock.Length > 0)
  {
    blocks.Add(currentBlock.ToString());
  }
  return blocks;
}
```

2. Определяется базовое регулярное выражение baseRegex для поиска координатных точек. Так как точка состоит из двух координат, целесообразно шифровать их вместе, как единое целое. Более сложное выражение regex используется для разделения блоков на координатные фрагменты и фрагменты, соответствующие любой последовательности символов, в которой не присутствуют координаты.

```
const string baseRegex = @"-?\d+(\.\d+)?\ -?\d+(\.\d+)?";
var regex = new Regex($@"({baseRegex})|((.(?!{baseRegex}))*.)", RegexOptions.Singleline);
```

3. Методом makeCollectionForStorage запускается параллельная обработка блоков текста. Проверяется условие соответствия регулярному выражению (baseRegex) и обработанные объекты сохраняется в параллельную коллекцию fragments (ConcurrentBag<ChipherDataSet>). Если условие истинно, то создается новый объект CipherDataSet с флагом приватности true, зашифрованным id с помощью метода EncryptID и зашифрованным текстом с помощью метода Encrypt. Если условие ложно, то создается новый объект CipherDataSet с флагом приватности false, зашифрованным id и текстом без шифрования. В качестве id выступает структура IndexData, которая хранит в себе индекс блока и индекс соответствия в данном блоке.

```
static ConcurrentBag<CipherDataSet> makeCollectionForStorage(List<string> textBlocks, Regex regex, string baseRegex, ThreadLocal<ICryptoTransform> encryptor)
{
  var fragments = new ConcurrentBag<CipherDataSet>();
  textBlocks.AsParallel().Select((block, i) => new { block, i }).ForAll(tuple =>
```

```
    {
      var matches = regex.Matches(tuple.block).OfType<Match>().ToList();
      matches.AsParallel().Select((match, i) => new { match, i }).ForAll(innerTuple =>
        {
          if (Regex.IsMatch(innerTuple.match.Value, baseRegex))
          {
            fragments.Add(new CipherDataSet
            {
              privateText = true,
              id = new IndexData { block = tuple.i, match = innerTuple.i }.EncryptID(encryptor),
              text = Encrypt(innerTuple.match.Value, encryptor)
            });
          }
          else
          {
            fragments.Add(new CipherDataSet
            {
              privateText = false,
              id = new IndexData { block = tuple.i, match = innerTuple.i }.EncryptID(encryptor),
              text = innerTuple.match.Value
            });
          }
        });
    });
    return fragments;
  }
  [DataContract]
  public struct CipherDataSet
  {
    [DataMember]
    public bool privateText;
    [DataMember]
    public string id;
    [DataMember]
    public string text;
  }
  [DataContract]
  public struct IndexData
  {
    [DataMember(Order = 1)]
    public long match;
    [DataMember(Order = 2)]
    public long block;

    public string EncryptID(ThreadLocal<Kuznyechik> kuznyechik)
    {
      DataContractJsonSerializer serializer = new DataContractJsonSerializer(typeof(IndexData));
      MemoryStream ms = new MemoryStream();
      serializer.WriteObject(ms, this);
      string data = Encoding.UTF8.GetString(ms.ToArray());
      return Encrypt(data, kuznyechik);
    }

    public void DecryptID(string encryptedData, ThreadLocal<Kuznyechik> kuznyechik)
    {
      string json = Decrypt(encryptedData, kuznyechik);
      DataContractJsonSerializer serializer = new DataContractJsonSerializer(typeof(IndexData));
      IndexData data = (IndexData)serializer.ReadObject(new MemoryStream(Encoding.UTF8.GetBytes(json)));
      this.block = data.block; this.match = data.match;
    }
  }
```

При работе с алгоритмом AES, вместо объекта kuznyechik типа ThreadLocal<Kuznyechik> на вход подаются объекты encryptor и decryptor типа ThreadLocal<ICryptoTransform>.

4. После параллельной обработки полученные данные с помощью метода SerializeToJson преобразуются в JSON-файл [18]:

```
static void SerializeToJson(ConcurrentBag<CipherDataSet> fragments, string filePath)
{
  try
  {
    var serializer = new DataContractJsonSerializer(typeof(ConcurrentBag<CipherDataSet>));
    using (var fileStream = File.Create(filePath))
    {
      serializer.WriteObject(fileStream, fragments);
    }
  }
  catch (Exception ex) { Console.WriteLine(ex.Message); }
}
```

5. Создание выходного файла, содержащего исходный текст, где зашифрованные фрагменты заменены на соответствующие маркеры.

```
static void CreateOutputFile(ConcurrentBag<CipherDataSet> fragments, ThreadLocal<Kuznyechik> kuznyechik, string filePath)
{
  try
  {
    var orderedFragmentsEnc = fragments
      .AsParallel()
      .OrderBy(ds =>
      {
        var id = new IndexData();
        id.DecryptID(ds.id, kuznyechik);
        return (id.block, id.match);
      }).Select(ds => ds.privateText ? "[begin]" + ds.text + "[end]" : ds.text).ToList();

    string[] orderedFragmentsEncArray = orderedFragmentsEnc.ToArray();
    string orderedFragmentsEncString = string.Join(" ", orderedFragmentsEncArray);

    using (StreamWriter writer = new StreamWriter(filePath))
    {
      writer.WriteLine(orderedFragmentsEncString);
    }
  }
  catch (Exception ex) { Console.WriteLine(ex.Message); }
}
```

При работе с алгоритмом AES работа ведется с аналогичными функциями, только вместо объекта kuznyechik типа ThreadLocal<Kuznyechik> используется объект encryptor/decryptor типа ThreadLocal<ICryptoTransform> [19].

Таким образом, обеспечивается потокобезопасное и параллельное кодирование координатных сведений с введением маркеров для удобства. На рисунке 2 представлена диаграмма работы программного модуля.

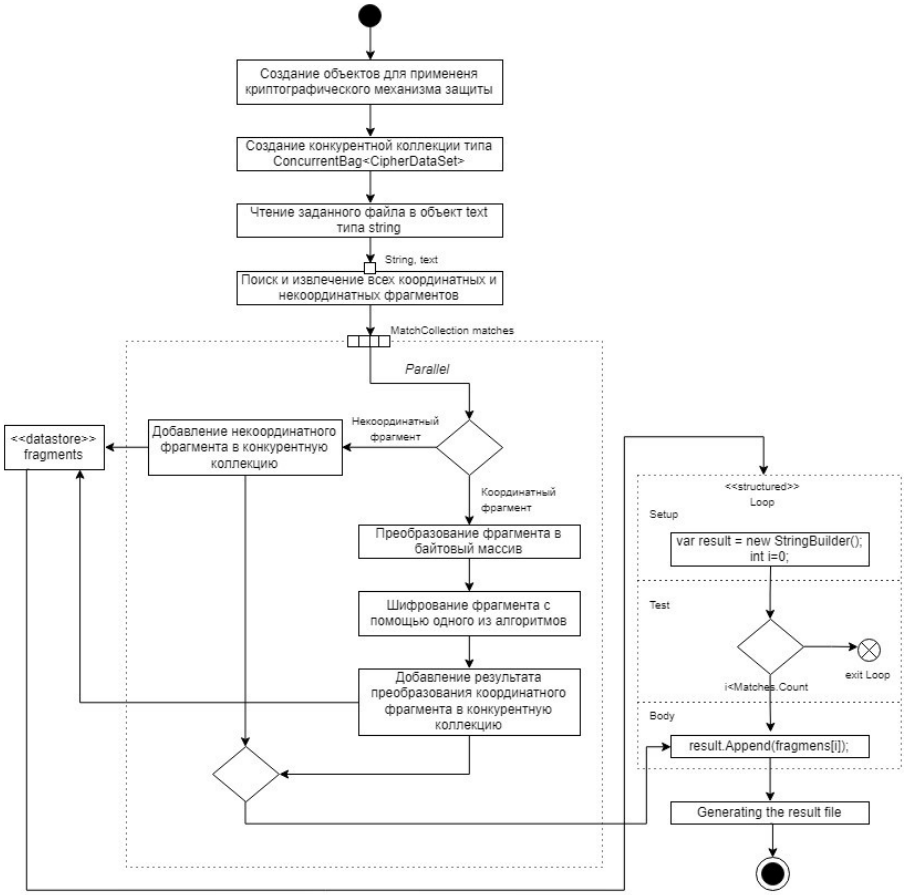



Рисунок 2 - Диаграмма работы программного модуля
DOI: https://doi.org/10.60797/IRJ.2025.152.4.2

Процесс дешифрования происходит следующим образом:

1. Считываются зашифрованные данные из JSON-файла PrivateText.json с помощью метода DeserializeFromJson. Он возвращает коллекцию IEnumerable<CipherDataSet>, где каждый элемент CipherDataSet содержит значения privateText, id, text.

2. Затем вызывается метод Decoding или Decoding_AES в зависимости от выбранного алгоритма шифрования (ГОСТ Р 34.12-2015 или AES соответственно), где элементы коллекции IEnumerable<CipherDataSet> сортируются по их индексам блоков и индексам соответствия с помощью OrderBy. Для расшифровки индексов используются методы DecryptID или DecryptID_AES [20].

3. Затем отсортированные элементы преобразуются в список строк, где зашифрованные фрагменты (privateText = true) расшифровываются с помощью Decrypt или Decrypt_AES, а незашифрованные фрагменты (privateText = false) остаются неизменными.

Полученный список строк объединяется в одну строку с помощью string.Join, которая затем записывается в файл с расшифрованным текстом (decyptedFile) с помощью StreamWriter.

**Тестирование и оценка программного модуля**

Демонстрация работы модуля производится на примере файла «admin.mif», содержимое которого представлено на рисунке 3. Результатом работы модуля является файл «output.mif». Его содержимое представлено на рисунке 4. Также создается файл «PrivateText.json» (рис. 5).

Для того чтобы убедиться в эффективности использования параллельной библиотеки в рамках данного модуля, было произведено тестирование двух версий программ: параллельной и последовательной.

admin.mif – Блокнот

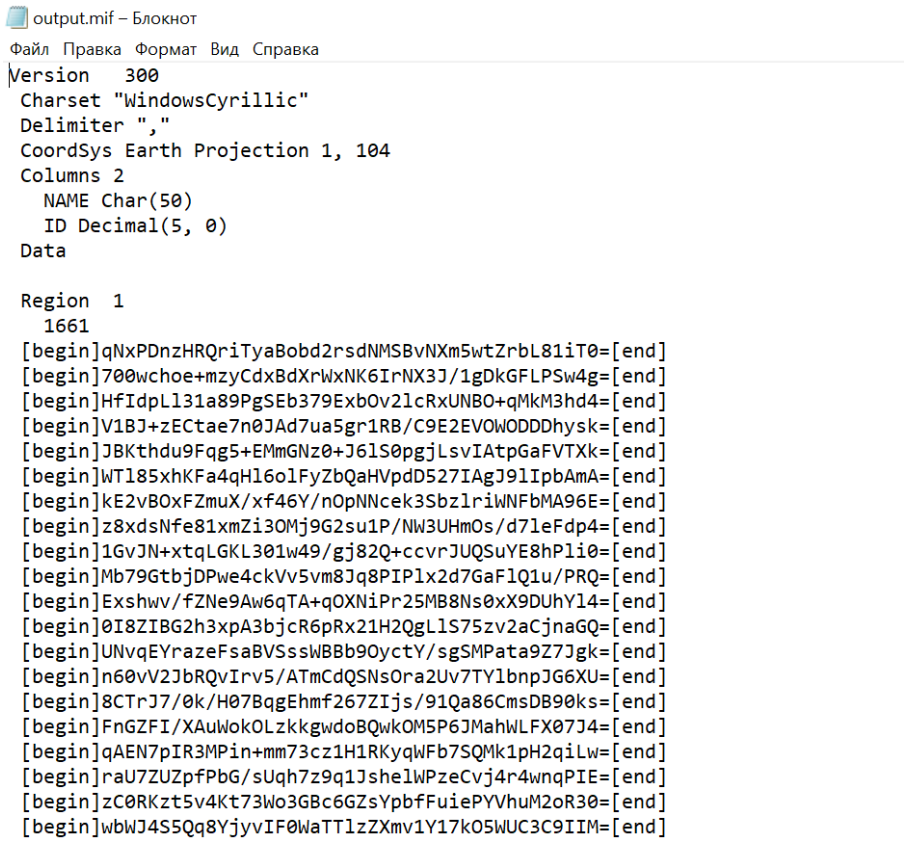
Файл Правка Формат Вид Справка

```
Version   300
Charset "WindowsCyrillic"
Delimiter ","
CoordSys Earth Projection 1, 104
Columns 2
  NAME Char(50)
  ID Decimal(5, 0)
Data

Region  1
  1661
88.624146 56.833153
88.644989 56.831108
88.646378 56.826385
88.64444 56.820549
88.643875 56.810272
88.641663 56.804436
88.640823 56.788887
88.6436 56.778877
88.643326 56.773605
88.644714 56.768883
88.643875 56.753052
88.644989 56.748047
88.644714 56.742767
88.649429 56.738602
88.679977 56.735268
88.709427 56.733879
```

Рисунок 3 - Пример исходного файла

DOI: https://doi.org/10.60797/IRJ.2025.152.4.3



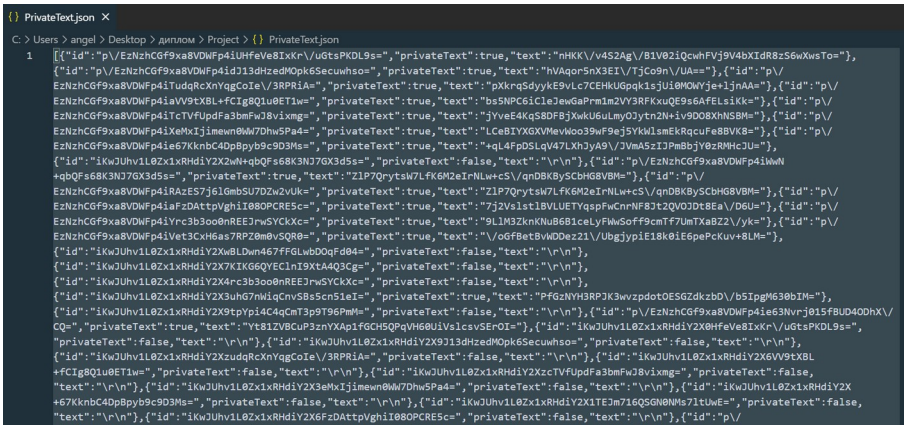
output.mif – Блокнот

Файл Правка Формат Вид Справка

```
Version   300
 Charset "WindowsCyrillic"
 Delimiter ","
 CoordSys Earth Projection 1, 104
 Columns 2
   NAME Char(50)
   ID Decimal(5, 0)
 Data

 Region  1
   1661
[begin]qNxPDnzHRQriTyaBobd2rsdNMSBvNXm5wtZrbL81iT0=[end]
[begin]700wchoe+mzyCdxBdXrWxNK6IrNX3J/1gDkGFLPSw4g=[end]
[begin]HfIdpLl31a89PgSEb379ExbOv2lcRxUNBO+qMkM3hd4=[end]
[begin]V1BJ+zECtae7n0JAd7ua5gr1RB/C9E2EVOWODDDhysk=[end]
[begin]JBKthdu9Fqg5+EMmGNz0+J6lS0pgjLsvIAtpGaFVTXk=[end]
[begin]WTl85xhKFa4qHl6olFyZbQaHVpdD527IAgJ9lIpbAmA=[end]
[begin]kE2vBOxFZmuX/xf46Y/nOpNNcek3SbzlriWNFbMA96E=[end]
[begin]z8xdsNfe81xmZi3OMj9G2su1P/NW3UHmOs/d7leFdp4=[end]
[begin]1GvJN+xtqLGKL301w49/gj82Q+ccvrJUQSuYE8hPli0=[end]
[begin]Mb79GtbjDPwe4ckVv5vm8Jq8PIPlx2d7GaFlQ1u/PRQ=[end]
[begin]Exshwv/fZNe9Aw6qTA+qOXNiPr25MB8Ns0xX9DUhYl4=[end]
[begin]0I8ZIBG2h3xpA3bjcR6pRx21H2QgLlS75zv2aCjnaGQ=[end]
[begin]UNvqEYrazeFsaBVSssWBBb9OyctY/sgSMPata9Z7Jgk=[end]
[begin]n60vV2JbRQvIrv5/ATmCdQSNsOra2Uv7TYlbnpJG6XU=[end]
[begin]8CTrJ7/0k/H07BqgEhmf267ZIjs/91Qa86CmsDB90ks=[end]
[begin]FnGZFI/XAuWokOLzkkgwdoBQwkOM5P6JMahWLFX07J4=[end]
[begin]qAEN7pIR3MPin+mm73cz1H1RKyqWFb7SQMk1pH2qiLw=[end]
[begin]raU7ZUZpfPbG/sUqh7z9q1JshelWPzeCvj4r4wnqPIE=[end]
[begin]zC0RKzt5v4Kt73Wo3GBc6GZsYpbfFuiePYVhuM2oR30=[end]
[begin]wbWJ4S5Qq8YjyvIF0WaTTlzZXmv1Y17kO5WUC3C9IIM=[end]
```

Рисунок 4 - Пример выходного файла
DOI: https://doi.org/10.60797/IRJ.2025.152.4.4



PrivateText.json

C: > Users > angel > Desktop > диплом > Project > {} PrivateText.json


```
[{"id":"p\/EzNzhCGf9xa8VDWFp4iUHfeVe8IxKr\/uGtsPKDL9s=","privateText":true,"text":"nHKK\/v4S2Ag\/B1V02iQcwhFVj9V4bXIdR8zS6wXwsTo="},
{"id":"p\/EzNzhCGf9xa8VDWFp4id313dHzedMOpk6Secuwhso=","privateText":true,"text":"hVAqorSnX3EI\/TjCo9n\/UA=="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iTudqRcXnYqgCoIe\/3RPRiA=","privateText":true,"text":"pXkrqSdyykE9vLc7CEHkUGpqk1sjUi0MOWYje+ljnAA="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iaVV9tXBL+fCIg8Q1u0ET1w=","privateText":true,"text":"bs5NPC6iCleJewGaPrm1m2VY3RFKxuQE9s6AfELsikk="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iTcTVfUpdFa3bmFwJ8vixmg=","privateText":true,"text":"jYveE4KqS8DFBjXwkU6uLmyOJytn2N+iv9DO8XhNSBM="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iXeMxIjimewn0WW7DhwSPa4=","privateText":true,"text":"LCeBIYXGXVMevWoo39wF9ej5YkWlsmEkRqcuFe8BVK8="},{"id":"p\/
EzNzhCGf9xa8VDWFp4ie67KknbC4DpBpyb9c9D3Ms=","privateText":true,"text":"+qL4FpDSLqV47LXhJyA9\/JVmA5zIJPmBbjV0zRMHcJU="},
{"id":"iKwJUhv1L0Zx1xRHdiY2X2wN+qbQFs68K3NJ7GX3d5s=","privateText":false,"text":"\r\n"},{"id":"p\/EzNzhCGf9xa8VDWFp4iWwN
+qbQFs68K3NJ7GX3d5s=","privateText":true,"text":"Z1P7QrytsW7LfK6M2eIrNLw+cS\/qnDBKBySCbHG8VBM="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iRAzES7j6lGmbSU7DZw2vUk=","privateText":true,"text":"Z1P7QrytsW7LfK6M2eIrNLw+cS\/qnDBKBySCbHG8VBM="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iaFzDAttpVghiI08OPCRE5c=","privateText":true,"text":"7j2VslstlBVLUETYqspFwCnrNF8Jt2QVOJDt8Ea\/D6U="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iYrc3b3oo0nREEJrwSYCkXc=","privateText":true,"text":"9LlM3ZknKNuB6B1ceLyFWwSoff9cmTf7UmTXaBZ2\/yk="},{"id":"p\/
EzNzhCGf9xa8VDWFp4iVet3CxH6as7RPZ0m0vSQR0=","privateText":true,"text":"\/oGfBet8vWDDez21\/UbgjypiE18k0iE6pePcKuv+8LM="},
{"id":"iKwJUhv1L0Zx1xRHdiY2XwBLDwn467fFGLwbDOqFd04=","privateText":false,"text":"\r\n"},
{"id":"iKwJUhv1L0Zx1xRHdiY2X7KIKG6QYEClnI9XtA4Q3Cg=","privateText":false,"text":"\r\n"},
{"id":"iKwJUhv1L0Zx1xRHdiY2X4rc3b3oo0nREEJrwSYCkXc=","privateText":false,"text":"\r\n"},
{"id":"iKwJUhv1L0Zx1xRHdiY2X3uhG7nWiqCnvSBs5cn51eI=","privateText":true,"text":"PfGzNYH3RPJK3wvzpdotOESGZdkzbD\/b5IpgM630bIM="},
{"id":"iKwJUhv1L0Zx1xRHdiY2X9tpYpi4C4qCmT3p9T96PmM=","privateText":false,"text":"\r\n"},{"id":"p\/EzNzhCGf9xa8VDWFp4ie63Nvrj015fBUD4ODhX\/
CQ=","privateText":true,"text":"Yt81ZVBCuP3znYXAp1fGCH5QPqVH60UiVslcsvSErOI="},{"id":"iKwJUhv1L0Zx1xRHdiY2X0HfeVe8IxKr\/uGtsPKDL9s=",
"privateText":false,"text":"\r\n"},{"id":"iKwJUhv1L0Zx1xRHdiY2X9J13dHzedMOpk6Secuwhso=","privateText":false,"text":"\r\n"},
{"id":"iKwJUhv1L0Zx1xRHdiY2XzudqRcXnYqgCoIe\/3RPRiA=","privateText":false,"text":"\r\n"},{"id":"iKwJUhv1L0Zx1xRHdiY2X6VV9tXBL
+fCIg8Q1u0ET1w=","privateText":false,"text":"\r\n"},{"id":"iKwJUhv1L0Zx1xRHdiY2XzcTVfUpdFa3bmFwJ8vixmg=","privateText":false,
"text":"\r\n"},{"id":"iKwJUhv1L0Zx1xRHdiY2X3eMxIjimewn0WW7DhwSPa4=","privateText":false,"text":"\r\n"},{"id":"iKwJUhv1L0Zx1xRHdiY2X
+67KknbC4DpBpyb9c9D3Ms=","privateText":false,"text":"\r\n"},{"id":"iKwJUhv1L0Zx1xRHdiY2X1TEJm716QSGN0NMs7ltUwE=","privateText":false,
"text":"\r\n"},{"id":"iKwJUhv1L0Zx1xRHdiY2X6FzDAttpVghiI08OPCRE5c=","privateText":false,"text":"\r\n"},{"id":"p\/
```


Рисунок 5 - Пример JSON-файла
DOI: https://doi.org/10.60797/IRJ.2025.152.4.5

Все эксперименты были произведены на персональном компьютере с процессором Intel Core i5-10210U с базовой частотой 1,6 ГГц под управлением ОС Windows 10. С помощью ГИС MapInfo были созданы тестовые файлы разного размера с различным количеством координатных данных. В таблицах 1-4 представлены времена обработки сгенерированных файлов для криптографических алгоритмов AES и Кузнечик соответственно. При этом под $t_s$ (sequential processing time) понимается время последовательной обработки файла, то время, за которое происходит проверка соответствия регулярному выражению для формирования коллекции с координатными и некоординатными фрагментами текста. Под $t_b$ (build time) – время сборки всех обработанных фрагментов и последующая их запись в новый файл. Под $t_d$ (decryption time) – время, затраченное на дешифрование координатных фрагментов, сборка всех данных в изначальном порядке и запись в новый файл.

Для каждого входного набора проводится пять экспериментов. Время, которое заносится в таблицу, берется как среднее арифметическое от времени, полученного в результате этих экспериментов (при 100 опытах). На рисунке 6 приведен пример одного такого эксперимента с файлом размером 630 Кбайт, в качестве алгоритма шифрования выбран алгоритм AES.

```
AES
Шифрование фрагментов, удовлетворяющих регулярное выражение, и создание выходных файлов:
Время шифрования 1265,1039 мс
Время создания файла 974,7613 мс
Дешифрование полученного файла с зашифрованным текстом:
Файл дешифрован за 2272,1458 мс
Количество зашифрованных фрагментов: 23430
```

Рисунок 6 - Пример эксперимента последовательной обработки файла алгоритмом AES
DOI: https://doi.org/10.60797/IRJ.2025.152.4.6

Таблица 1 - Время последовательной обработки файлов алгоритмом AES

DOI: https://doi.org/10.60797/IRJ.2025.152.4.7

| Размер файла, КБ | Количество координат в файле | Среднее время обработки, мс | | |
|---|---|---|---|---|
| | | $t_s$ | $t_b$ | $t_d$ |
| 630 | 23430 | 1865 | 1011 | 1297 |
| 1220 | 46817 | 3300 | 1488 | 2460 |
| 2515 | 93643 | 14772 | 6085 | 9287 |
| 5000 | 187219 | 18568 | 10909 | 13875 |
| 10000 | 374408 | 43581 | 18712 | 39667 |

Таблица 2 - Время параллельной обработки файлов алгоритмом AES

DOI: https://doi.org/10.60797/IRJ.2025.152.4.8

| Размер файла, КБ | Количество координат в файле | Среднее время обработки, мс | | |
|---|---|---|---|---|
| | | $t_p$ | $t_b$ | $t_d$ |
| 630 | 23430 | 1260 | 905 | 2054 |
| 1220 | 46817 | 2538 | 2932 | 3997 |
| 2515 | 93643 | 4752 | 4364 | 7689 |
| 5000 | 187219 | 7327 | 3165 | 6901 |
| 10000 | 374408 | 18103 | 22608 | 18956 |

Таблица 3 - Время последовательной обработки алгоритмом Кузнечик

DOI: https://doi.org/10.60797/IRJ.2025.152.4.9

| Размер файла, КБ | Количество координат в файле | Среднее время обработки, мс | | |
|---|---|---|---|---|
| | | $t_s$ | $t_b$ | $t_d$ |
| 630 | 23430 | 47551 | 33904 | 48686 |
| 1220 | 46817 | 94021 | 67310 | 96170 |
| 2515 | 93643 | 191318 | 178603 | 285241 |
| 5000 | 187219 | 411617 | 282653 | 408866 |
| 10000 | 374408 | 848554 | 563671 | 816030 |

Таблица 4 - Время параллельной обработки файлов алгоритмом Кузнечик

DOI: https://doi.org/10.60797/IRJ.2025.152.4.10

| Размер файла, КБ | Количество координат в файле | Среднее время обработки, мс | | |
|---|---|---|---|---|
| | | $t_p$ | $t_b$ | $t_d$ |
| 630 | 23430 | 11868 | 8142 | 11801 |

| Размер файла, КБ | Количество координат в | Среднее время обработки, мс | | |
|---|---|---|---|---|
| | | $t_p$ | $t_b$ | $t_d$ |
| 1220 | 46817 | 25378 | 15486 | 24041 |
| 2515 | 93643 | 46753 | 32633 | 49455 |
| 5000 | 187219 | 96395 | 66568 | 113984 |
| 10000 | 374408 | 201095 | 146188 | 210434 |

В результате всех проведенных экспериментов можно с уверенностью заявить, что при увеличении размера входного файла параллельная версия программы также увеличивает свое преимущество перед последовательной версией. Это происходит благодаря тому, что при параллельной обработке используются все мощности процессора (рис. 7), в то время как при последовательной обработке используется только 30-40% возможностей процессора (рис. 8).

К тому же, в результате сравнительного анализа двух приведенных алгоритмов, стоит сделать вывод, что AES работает в разы быстрее алгоритма «Кузнечик». Это связано с тем, что компания Intel с 2010 года начала выпускать процессоры с набором новых инструкций – Intel Advanced Encryption Standard (AES) New Instructions (AES-NI) [21], [22]. Благодаря этим инструкциям алгоритм выполняется с использованием аппаратных средств, что делает процесс шифрования и дешифрования гораздо быстрее. В случае последовательной реализации использование инструкций AES-NI позволяет добиться увеличения скорости в 2-3 раза по сравнению с чисто программной реализацией алгоритма AES. Однако в условиях параллельной обработки данных наблюдается рост производительности, достигающий 10-кратного ускорения.

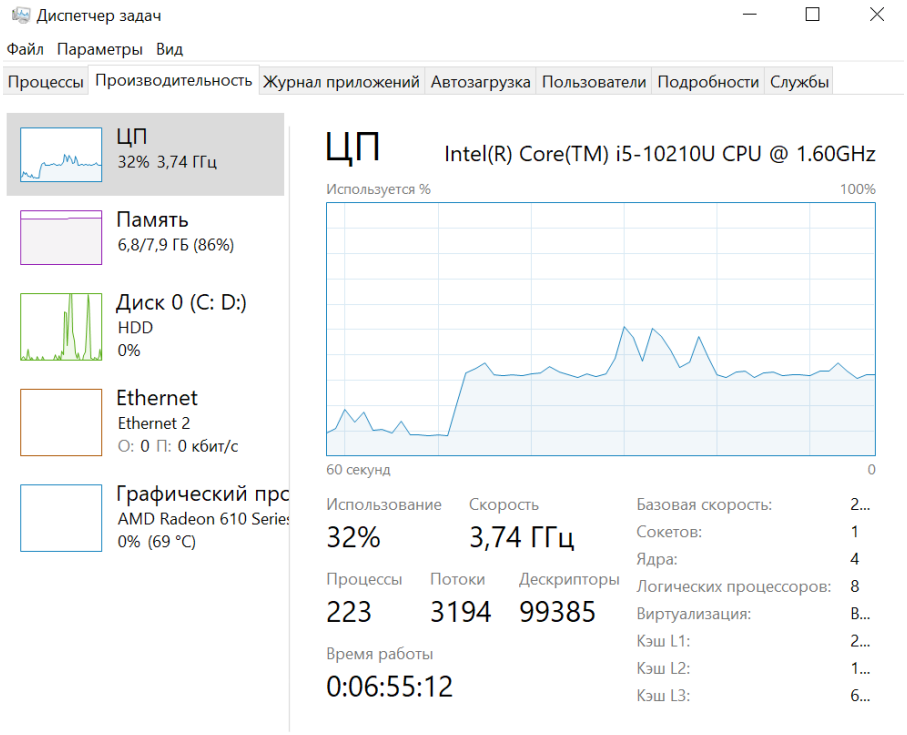



Рисунок 7 - Нагрузка на процессор при последовательной обработке
DOI: https://doi.org/10.60797/IRJ.2025.152.4.11

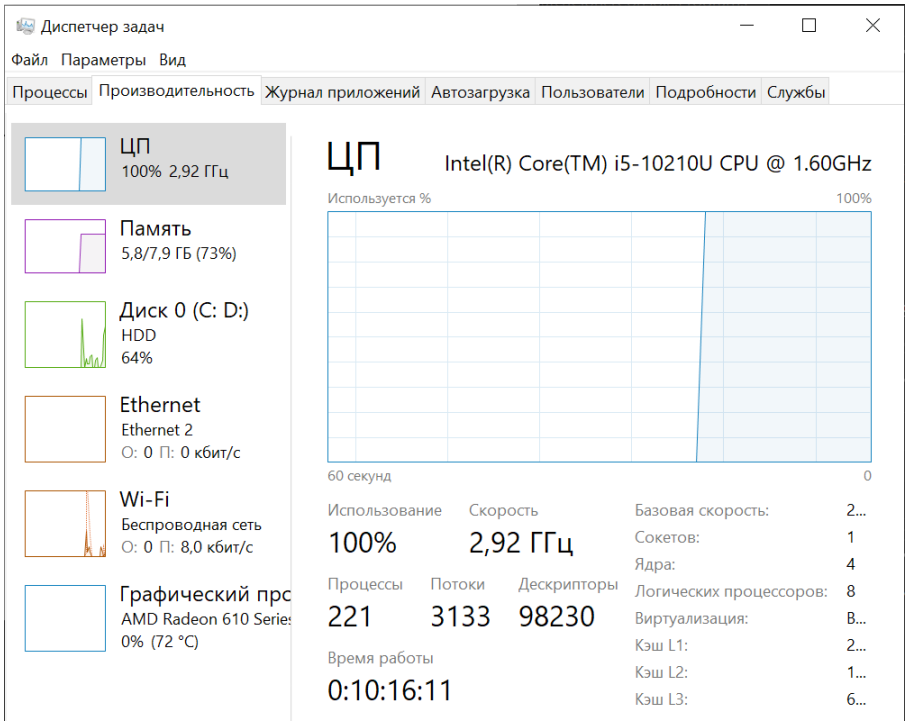



Рисунок 8 - Нагрузка на процессор при параллельной обработке
DOI: https://doi.org/10.60797/IRJ.2025.152.4.12

Стоит также оценить данный модуль и со стороны стойкости алгоритмов шифрования. Так как в модуле применяются алгоритм AES и Кузнечик с длиной ключа 256 бит, можно сказать, что модуль обладает высокой стойкостью, ведь на данный момент не обнаружено никаких эффективных криптоаналитических атак, способных поставить под угрозу используемые алгоритмы с такой длиной ключа. Это обеспечивает надежную защиту зашифрованных данных от несанкционированного доступа и восстановления исходных данных без соответствующего ключа.

В результате проведенного тестирования и последующего анализа для оценки производительности и стойкости данного модуля выявлено, что данный модуль функционирует с необходимыми показателями производительности и обеспечивает высокий уровень криптостойкости.

**Сравнительный анализ предложенного модуля с существующими решениями**

В процессе разработки модуля криптографической защиты файлов ГИС MapInfo был проведён анализ существующих решений для защиты данных. Несмотря на наличие ряда инструментов для шифрования, таких как VeraCrypt, NordLocker и AxCrypt, их функциональность не позволяет в полной мере удовлетворить специфические потребности геоинформационных систем (см. таблицу 5).

В отличие от рассмотренных решений, предложенный модуль шифрует только ценные данные, такие как координаты, оставляя остальную информацию открытой. Это снижает затраты на обработку данных и уменьшает объём результирующих файлов.

Сравнительные оценки показывают, что на персональном компьютере с процессором Intel Core i5-10210U для обработки файла размером 500 МБ с 250 тыс. координатных точек последовательный алгоритм занимает около 150 сек., в то время как параллельный подход сокращает это время до 40 сек.

Регулярные выражения обеспечивают эффективный поиск данных по заданным шаблонам. Например, поиск координатных данных в формате `X Y` в файле размером 500 МБ занимает около 100 мсек., что позволяет поддерживать высокую производительность.

Использование алгоритма AES с аппаратной поддержкой (AES-NI) обеспечивает прирост скорости до 3–5 раз в сравнении с программной реализацией, особенно на процессорах с несколькими ядрами.

Применение предложенного модуля наиболее эффективно в следующих случаях:

- Обработка больших файлов (от 100 МБ и выше) с высокой плотностью координатных данных.
- Организации, которым требуется минимизировать ресурсы, затрачиваемые на шифрование, за счёт избирательного подхода.

Сценарии, где важна интеграция с существующими системами обработки данных, такими как ГИС MapInfo.

Таблица 5 - Сравнение со стандартными решениями

DOI: https://doi.org/10.60797/IRJ.2025.152.4.13

| Характеристика | VeraCrypt | NordLocker | AxCrypt | Предложенный модуль |
|---|---|---|---|---|
| Избирательное шифрование | Нет | Нет | Нет | Да |
| Поддержка ГИС форматов | Нет | Нет | Нет | Да |
| Производительность (500 МБ) | 50–100 сек. | 300–500 сек. | 200–400 сек. | 40–60 сек. (параллельно) |

**Заключение**

В результате проделанной работы разработан программный модуль для избирательного шифрования файлов геоинформационной системы MapInfo, основанный на использовании регулярных выражений и параллельной библиотеки платформы .NET. В ходе работы проведён анализ производительности, показавший, что параллельная реализация обеспечивает значительное ускорение обработки данных до десяти раз по сравнению с последовательной при работе с большими объёмами. В модуле внедрены алгоритмы AES и ГОСТ «Кузнечик», которые обеспечивают высокий уровень криптографической стойкости. Новизна разработки заключается в сочетании регулярных выражений и параллельной обработки для точечного шифрования конфиденциальной информации, а также в применении подхода избирательного шифрования, что позволяет минимизировать ресурсные затраты за счёт обработки только ценных данных, таких как координаты. Оригинальность работы состоит в адаптации предложенного решения к специфическому формату файлов ГИС MapInfo с учётом их особенностей, а также в использовании конкурентных коллекций платформы .NET, что позволило достичь высокой эффективности при многопоточной обработке данных.

Тем не менее, предложенное решение обладает рядом ограничений, которые могут сдерживать его применение в специфических сценариях.

Одной из ключевых проблем является ограниченность форматов файлов, которые поддерживаются модулем. В настоящий момент он работает только с файлами формата MIF, что делает невозможным его использование с другими популярными форматами, такими как ShapeFile [23], GeoJSON [24]. Для устранения этого ограничения предлагается разработать универсальный интерфейс, который обеспечит обработку различных типов данных, либо расширить функционал модуля, добавив поддержку новых форматов. Ещё одной проблемой является необходимость ручной настройки шаблонов регулярных выражений для каждого типа данных, что может быть затруднительно при работе с нестандартными или сложными структурами данных. Для решения этой проблемы возможно создание модуля автоматической генерации шаблонов на основе анализа данных или внедрение технологий машинного обучения для автоматического распознавания структур.

Эффективность многопоточной обработки, реализованной в модуле, в значительной степени зависит от аппаратного обеспечения, и на системах с ограниченным количеством процессорных ядер она может быть снижена. Для преодоления данного ограничения возможно внедрение адаптивного управления потоками, которое учитывало бы архитектурные особенности используемой аппаратной платформы, а также применение гибридных подходов, включая использование распределённых вычислений [25], [26], [27]. Несмотря на высокую криптографическую стойкость алгоритмов AES и «Кузнечик», их использование может быть недостаточным для долгосрочной защиты данных, особенно в условиях развития квантовых вычислений. В этой связи целесообразной мерой представляется интеграция с библиотеками, поддерживающими постквантовые алгоритмы, такими как CRYSTALS-Kyber [28] или NTRU [29], что позволит обеспечить защиту данных в долгосрочной перспективе.

Дальнейшее развитие модуля может быть направлено на интеграцию с облачными сервисами, что позволит использовать его в распределённых средах для обработки данных, хранящихся в облачных системах. Дополнение функционала инструментами для анализа геопространственных данных, такими как автоматическое выявление аномалий, может повысить его ценность для конечных пользователей. Реализация модульной архитектуры, при которой функции модуля (например, поиск данных, шифрование, обработка ошибок) разделяются на независимые компоненты, сделает решение более гибким и масштабируемым.

**Конфликт интересов**

Не указан.

**Рецензия**

Кацко С.Ю., Сибирский государственный университет геосистем и технологий, Новосибирск, Российская Федерация
DOI: https://doi.org/10.60797/IRJ.2025.152.4.14

**Conflict of Interest**

None declared.

**Review**

Katsko S.Y., Siberian State University of Geosystems and Technologies, Novosibirsk, Russian Federation
DOI: https://doi.org/10.60797/IRJ.2025.152.4.14

## Список литературы / References


1. Гибадуллин Р.Ф. Развитие единообразного формализма защиты точечных, линейных и площадных объектов картографии / Р.Ф. Гибадуллин // Вестник Казанского государственного технического университета им. А.Н. Туполева. — 2010. — № 2. — С. 101–105.
2. Spero E. Helping users secure their data by supporting mental models of VeraCrypt / E. Spero, M. Stojmenović, R. Biddle // HCI International 2019-Posters: 21st International Conference, HCII 2019. — Orlando: Springer International Publishing 2019. — Vol. 21. — Pt. I. — P. 211–218.
3. Mahdi O.M.E. EFTS: An encryption file transfer system applying advanced encryption standard (AES) algorithm / O.M.E. Mahdi, Ju. Juremi // AIP Conference Proceedings. — 2024. — Vol. 2802. — № 1.
4. Kabilan K. Implementation of SERPENT cryptographic algorithm for secured data transmission / K. Kabilan, M. Saketh, K.K. Nagarajan // 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). — Coimbatore, 2017. — P. 1–6. — DOI: 10.1109/ICIIECS.2017.8275863.
5. Мартинов Г.М. Перспективные технологии разработки математического обеспечения систем управления: использование регулярных выражений / Г.М. Мартинов, В.Л. Сосонкин // Мехатроника, автоматизация, управление. — 2006. — № 2. — С. 40–46.
6. Bourhis P. JSON: Data model and query languages / P. Bourhis, J.L. Reutter, D. Vrgoč // Information Systems. — 2020. — № 89. — P. 101478.
7. Fei X. A fast parallel cryptography algorithm based on AES-NI / X. Fei, K. Li, W. Yang // Journal of Intelligent & Fuzzy Systems. — 2016. — № 31 (2). — P. 1099–1107.
8. Reddy M.S.S. Design and Implementation of Parallel AES Encryption Engines for Multi-Core Processor Arrays / M.S.S. Reddy, P.J. Vijay, B.M. Krishna // International Journal of Engineering Development and Research. — 2014. — № 2 (4). — P. 3656–3661.
9. Spero E. Helping users secure their data by supporting mental models of VeraCrypt / E. Spero, M. Stojmenović, R. Biddle // HCI International 2019-Posters: 21st International Conference, HCII 2019. — Orlando: Springer International Publishing, 2019.
10. Djeki E. Data confidentiality and integrity in cloud storage environment / E. Djeki, C. Bondiombouy, Ju. Degila // Communication and Intelligent Systems: Proceedings of ICCIS 2020. — Singapore: Springer Singapore, 2021.
11. Mahdi O.M.E. EFTS: An encryption file transfer system applying advanced encryption standard (AES) algorithm / O.M.E. Mahdi, Ju. Juremi // AIP Conference Proceedings. — 2024. — Vol. 2802. — № 1.
12. Мартинов Г.М. Перспективные технологии разработки математического обеспечения систем управления: использование регулярных выражений / Г.М. Мартинов, В.Л. Сосонкин // Мехатроника, автоматизация, управление. — 2006. — № 2. — С. 40–46.
13. Гибадуллин Р.Ф. Параллельные модули импорт и экспорта защищенной картографической базы данных / Р.Ф. Гибадуллин, Р.М. Гарипов, М.М. Диаров // Поиск эффективных решений в процессе создания и реализации научных разработок в российской авиационной и ракетно-космической промышленности : Международная научно-практическая конференция, Казань, 05–08 августа 2014 года. — Казань: Издательство Казанского государственного технического университета, 2014. — Т. II. — С. 418–421.
14. Гибадуллин Р.Ф. Потокобезопасные вызовы элементов управления в обогащенных клиентских приложениях / Р.Ф. Гибадуллин // Программные системы и вычислительные методы. — 2022. — № 4. — С. 1–19. — DOI: 10.7256/2454-0714.2022.4.39029.
15. Гибадуллин Р.Ф. Неоднозначность результатов при использовании методов класса Parallel в рамках исполняющей среды .NET Framework / Р.Ф. Гибадуллин, И.В. Викторов // Программные системы и вычислительные методы. — 2023. — № 2. — С. 1–14. — DOI: 10.7256/2454-0714.2023.2.39801.
16. Sarcar V. Exploring Synchronization and Concurrent Collections / V. Sarcar // Parallel Programming with C# and. NET: Fundamentals of Concurrency and Asynchrony Behind Fast-Paced Applications. — Berkeley: Apress, 2024. — P. 143–218.
17. Бабенко Л.К. Дифференциальный анализ шифра Кузнечик / Л. К. Бабенко, Е.А. Ищукова, Е.А. Толоманенко // Известия Южного федерального университета. Технические науки. — 2017. — № 5 (190). — С. 25–37.
18. Bourhis P. JSON: Data model and query languages / P. Bourhis, J.L. Reutter, D. Vrgoč // Information Systems. — 2020. — № 89. — P. 101478.
19. Panjuta D. C# Cryptography / D. Panjuta, J. Jabbarzadeh // Learning C# Through Small Projects. — Cham: Springer Nature Switzerland, 2024. — P. 269–304.
20. Abdullah A.M. Advanced encryption standard (AES) algorithm to encrypt and decrypt data / A.M. Abdullah // Cryptography and Network Security. — 2017. — № 16 (1). — P. 1.
21. Fei X. A fast parallel cryptography algorithm based on AES-NI / X. Fei, K. Li, W. Yang // Journal of Intelligent & Fuzzy Systems. — 2016. — № 31 (2). — P. 1099–1107.
22. Hofemeier G. Introduction to intel aes-ni and intel secure key instructions / H. Gael, R. Chesebrough // Intel, White Paper. — 2012. — № 62. — P. 6.
23. Zhu J. An Assessment of Paths for Transforming IFC to Shapefile for Integration of BIM and GIS / J. Zhu, P. Wang, X. Wang // 2018 26th International Conference on Geoinformatics. — Kunming, 2018. — P. 1–5. — DOI: 10.1109/GEOINFORMATICS.2018.8557099.
24. Da Costa Rainho F. Web GIS: A new system to store spatial data using GeoJSON in MongoDB / F. da Costa Rainho, J. Bernardino // 2018 13th Iberian Conference on Information Systems and Technologies (CISTI). — Caceres, 2018. — P. 1–6. — DOI: 10.23919/CISTI.2018.8399279.

25. Sabirov N.A. Parallel Processing of SQL Queries Using MPI.NET / N.A. Sabirov, R.F. Gibadullin // 2024 International Russian Smart Industry Conference (SmartIndustryCon). — Sochi, 2024. — P. 344–349. — DOI: 10.1109/SmartIndustryCon61328.2024.10516133.

26. Uteyev G. Development of the Decentralized Biometric Identity Verification System Using Blockchain Technology and Computer Vision / G. Uteyev, R.F. Gibadullin // 2024 International Russian Smart Industry Conference (SmartIndustryCon). — Sochi, 2024. — P. 350–355. — DOI: 10.1109/SmartIndustryCon61328.2024.10516166.

27. Гибадуллин Р.Ф. Анализ параметров промышленных сетей с применением нейросетевой обработки / Р.Ф. Гибадуллин, Д.В. Лекомцев, М.Ю. Перухин // Искусственный интеллект и принятие решений. — 2020. — № 1. — С. 80–87. — DOI: 10.14357/20718594200108.

28. Ji Y. A Side-Channel Attack on a Hardware Implementation of CRYSTALS-Kyber / Y. Ji, R. Wang, K. Ngo [et al.] // 2023 IEEE European Test Symposium (ETS). — Venezia, 2023. — P. 1–5. — DOI: 10.1109/ETS56758.2023.10174000.

29. Lee J. Cryptanalysis on "NTRU+: Compact Construction of NTRU Using Simple Encoding Method" / J. Lee, H. Ryu, M. Lee [et al.] // IEEE Transactions on Information Forensics and Security. — 2024. — Vol. 19. — P. 9508–9517. — DOI: 10.1109/TIFS.2024.3471074.

## Список литературы на английском языке / References in English

1. Gibadullin R.F. Razvitie edinoobraznogo formalizma zashhity tochechnyh, linejnyh i ploshhadnyh ob'ektov kartografii [Development of uniform formalism of protection of point, linear and area objects of cartography] / R.F. Gibadullin // Vestnik Kazanskogo gosudarstvennogo tehnicheskogo universiteta im. A.N. Tupoleva [Bulletin of Kazan State Technical University named after A.N. Tupolev]. — 2010. — № 2. — P. 101–105. [in Russian]

2. Spero E. Helping users secure their data by supporting mental models of VeraCrypt / E. Spero, M. Stojmenović, R. Biddle // HCI International 2019-Posters: 21st International Conference, HCII 2019. — Orlando: Springer International Publishing 2019. — Vol. 21. — Pt. I. — P. 211–218.

3. Mahdi O.M.E. EFTS: An encryption file transfer system applying advanced encryption standard (AES) algorithm / O.M.E. Mahdi, Ju. Juremi // AIP Conference Proceedings. — 2024. — Vol. 2802. — № 1.

4. Kabilan K. Implementation of SERPENT cryptographic algorithm for secured data transmission / K. Kabilan, M. Saketh, K.K. Nagarajan // 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). — Coimbatore, 2017. — P. 1–6. — DOI: 10.1109/ICIIECS.2017.8275863.

5. Martinov G.M. Perspektivnye tehnologii razrabotki matematicheskogo obespechenija sistem upravlenija: ispol'zovanie reguljarnyh vyrazhenij [Perspective technologies of development of mathematical support for control systems: use of regular expressions] / G.M. Martinov, V.L. Sosonkin // Mehatronika, avtomatizacija, upravlenie [Mechatronics, Automation, Management]. — 2006. — № 2. — P. 40–46. [in Russian]

6. Bourhis P. JSON: Data model and query languages / P. Bourhis, J.L. Reutter, D. Vrgoč // Information Systems. — 2020. — № 89. — P. 101478.

7. Fei X. A fast parallel cryptography algorithm based on AES-NI / X. Fei, K. Li, W. Yang // Journal of Intelligent & Fuzzy Systems. — 2016. — № 31 (2). — P. 1099–1107.

8. Reddy M.S.S. Design and Implementation of Parallel AES Encryption Engines for Multi-Core Processor Arrays / M.S.S. Reddy, P.J. Vijay, B.M. Krishna // International Journal of Engineering Development and Research. — 2014. — № 2 (4). — P. 3656–3661.

9. Spero E. Helping users secure their data by supporting mental models of VeraCrypt / E. Spero, M. Stojmenović, R. Biddle // HCI International 2019-Posters: 21st International Conference, HCII 2019. — Orlando: Springer International Publishing, 2019.

10. Djeki E. Data confidentiality and integrity in cloud storage environment / E. Djeki, C. Bondiombouy, Ju. Degila // Communication and Intelligent Systems: Proceedings of ICCIS 2020. — Singapore: Springer Singapore, 2021.

11. Mahdi O.M.E. EFTS: An encryption file transfer system applying advanced encryption standard (AES) algorithm / O.M.E. Mahdi, Ju. Juremi // AIP Conference Proceedings. — 2024. — Vol. 2802. — № 1.

12. Martinov G.M. Perspektivnye tehnologii razrabotki matematicheskogo obespechenija sistem upravlenija: ispol'zovanie reguljarnyh vyrazhenij [Perspective technologies of development of mathematical support for control systems: use of regular expressions] / G.M. Martinov, V.L. Sosonkin // Mehatronika, avtomatizacija, upravlenie [Mechatronics, Automation, Management]. — 2006. — № 2. — P. 40–46. [in Russian]

13. Gibadullin R.F. Parallel'nye moduli import i jeksporta zashhishhennoj kartograficheskoj bazy dannyh [Parallel modules of import and export of the protected cartographic database] / R.F. Gibadullin, R.M. Garipov, M.M. Diarov // Poisk jeffektivnyh reshenij v processe sozdanija i realizacii nauchnyh razrabotok v rossijskoj aviacionnoj i raketno-kosmicheskoj promyshlennosti : Mezhdunarodnaja nauchno-prakticheskaja konferencija, Kazan', 05–08 avgusta 2014 goda [Search for effective solutions in the process of creation and implementation of scientific developments in the Russian aviation and rocket-space industry : International Scientific and Practical Conference, Kazan, 05-08 August 2014]. — Kazan: Kazan State Technical University Publishing House, 2014. — Vol. II. — P. 418–421. [in Russian]

14. Gibadullin R.F. Potokobezopasnye vyzovy jelementov upravlenija v obogashhennyh klientskih prilozhenijah [Thread-safe calls of control elements in enriched client applications] / R.F. Gibadullin // Programmnye sistemy i vychislitel'nye metody [Software Systems and Computational Methods]. — 2022. — № 4. — P. 1–19. — DOI: 10.7256/2454-0714.2022.4.39029. [in Russian]

15. Gibadullin R.F. Neodnoznachnost' rezul'tatov pri ispol'zovanii metodov klassa Parallel v ramkah ispolnjajushhej sredy .NET Framework [Ambiguity of results when using methods of Parallel class within the framework of executable environment .NET Framework] / R.F. Gibadullin, I.V. Viktorov // Programmnye sistemy i vychislitel'nye metody [Programme Systems and Computational Methods]. — 2023. — № 2. — P. 1–14. — DOI: 10.7256/2454-0714.2023.2.39801. [in Russian]

16. Sarcar V. Exploring Synchronization and Concurrent Collections / V. Sarcar // Parallel Programming with C# and. NET: Fundamentals of Concurrency and Asynchrony Behind Fast-Paced Applications. — Berkeley: Apress, 2024. — P. 143–218.

17. Babenko L.K. Differencial'nyj analiz shifra Kuznechik [Differential analysis of the Grasshopper cipher] / L. K. Babenko, E.A. Ishhukova, E.A. Tolomanenko // Izvestija Juzhnogo federal'nogo universiteta. Tehnicheskie nauki [Proceedings of the Southern Federal University. Technical Sciences]. — 2017. — № 5 (190). — P. 25–37. [in Russian]

18. Bourhis P. JSON: Data model and query languages / P. Bourhis, J.L. Reutter, D. Vrgoč // Information Systems. — 2020. — № 89. — P. 101478.

19. Panjuta D. C# Cryptography / D. Panjuta, J. Jabbarzadeh // Learning C# Through Small Projects. — Cham: Springer Nature Switzerland, 2024. — P. 269–304.

20. Abdullah A.M. Advanced encryption standard (AES) algorithm to encrypt and decrypt data / A.M. Abdullah // Cryptography and Network Security. — 2017. — № 16 (1). — P. 1.

21. Fei X. A fast parallel cryptography algorithm based on AES-NI / X. Fei, K. Li, W. Yang // Journal of Intelligent & Fuzzy Systems. — 2016. — № 31 (2). — P. 1099–1107.

22. Hofemeier G. Introduction to intel aes-ni and intel secure key instructions / H. Gael, R. Chesebrough // Intel, White Paper. — 2012. — № 62. — P. 6.

23. Zhu J. An Assessment of Paths for Transforming IFC to Shapefile for Integration of BIM and GIS / J. Zhu, P. Wang, X. Wang // 2018 26th International Conference on Geoinformatics. — Kunming, 2018. — P. 1–5. — DOI: 10.1109/GEOINFORMATICS.2018.8557099.

24. Da Costa Rainho F. Web GIS: A new system to store spatial data using GeoJSON in MongoDB / F. da Costa Rainho, J. Bernardino // 2018 13th Iberian Conference on Information Systems and Technologies (CISTI). — Caceres, 2018. — P. 1–6. — DOI: 10.23919/CISTI.2018.8399279.

25. Sabirov N.A. Parallel Processing of SQL Queries Using MPI.NET / N.A. Sabirov, R.F. Gibadullin // 2024 International Russian Smart Industry Conference (SmartIndustryCon). — Sochi, 2024. — P. 344–349. — DOI: 10.1109/SmartIndustryCon61328.2024.10516133.

26. Uteyev G. Development of the Decentralized Biometric Identity Verification System Using Blockchain Technology and Computer Vision / G. Uteyev, R.F. Gibadullin // 2024 International Russian Smart Industry Conference (SmartIndustryCon). — Sochi, 2024. — P. 350–355. — DOI: 10.1109/SmartIndustryCon61328.2024.10516166.

27. Gibadullin R.F. Analiz parametrov promyshlennyh setej s primeneniem nejrosetevoj obrabotki [Analysis of industrial networks parameters using neural network processing] / R.F. Gibadullin, D.V. Lekomcev, M.Ju. Peruhin // Iskusstvennyj intellekt i prinjatie reshenij [Artificial Intelligence and Decision-Making]. — 2020. — № 1. — P. 80–87. — DOI: 10.14357/20718594200108. [in Russian]

28. Ji Y. A Side-Channel Attack on a Hardware Implementation of CRYSTALS-Kyber / Y. Ji, R. Wang, K. Ngo [et al.] // 2023 IEEE European Test Symposium (ETS). — Venezia, 2023. — P. 1–5. — DOI: 10.1109/ETS56758.2023.10174000.

29. Lee J. Cryptanalysis on "NTRU+: Compact Construction of NTRU Using Simple Encoding Method" / J. Lee, H. Ryu, M. Lee [et al.] // IEEE Transactions on Information Forensics and Security. — 2024. — Vol. 19. — P. 9508–9517. — DOI: 10.1109/TIFS.2024.3471074.