УДК 004.056.55

А. В. Кизино, О. А. Новосельская
Белорусский государственный технологический университет

АЛГОРИТМЫ И ПРОГРАММНОЕ СРЕДСТВО ГЕНЕРАЦИИ КЛЮЧА ДЛЯ ПОСЛЕДУЮЩЕГО КОДИРОВАНИЯ ИНФОРМАЦИИ В ЗАЩИТНЫХ ВЕКТОРНЫХ ИЗОБРАЖЕНИЯХ

Рассмотрены теоретические и практические аспекты построения алгоритмов и программного средства для генерации ключей, предназначенных для последующего кодирования информации в защитных векторных изображениях. К подобного рода изображениям относят в первую очередь гильоши, которые являются векторными и воспроизводятся специальными способами печати. Однако для внедрения информации в электронные документы необходимо кодирование информации. В статье описаны основные функции для реализации кодирования информации в защитных векторных изображениях. Уделено внимание формированию контейнера, обеспечивающего целостность и конфиденциальность информации, с использованием комбинации симметричных и асимметричных алгоритмов шифрования, а также хэш-функций. Рассмотрены недостатки централизованных систем хранения ключей и показаны риски, связанные с их компрометацией. В качестве альтернативы предложен метод, аналогичный электронной подписи, что соответствует требованиям законодательства Республики Беларусь. Описан алгоритм генерации ключа на основе анализа траектории движения мыши пользователя. Предложенный метод обеспечивает высокую степень уникальности и криптостойкости формируемых ключей за счет использования биометрических и поведенческих параметров, что существенно затрудняет подбор или повторное воспроизведение ключа. Экспериментальная часть включает разработку программного средства с визуальным интерфейсом, позволяющего фиксировать параметры движения мыши и формировать массив значений, преобразуемых в криптографический ключ. Анализ стабильности и повторяемости результатов подтвердил надежность метода. Полученные ключи интегрируются в структуру защитного векторного изображения, что позволяет не только повысить уровень защиты, но и реализовать возможность внедрения скрытой информации в элементы фирменного стиля или электронную документацию.

Ключевые слова: гильош, генерация ключа, векторная графика, защита информации, электронная подпись, RSA, случайные последовательности, криптография.

Для цитирования: Кизино А. В., Новосельская О. А. Алгоритмы и программное средство генерации ключа для последующего кодирования информации в защитных векторных изображениях // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2025. № 2 (296) С. 91–98.
DOI: 10.52025/2520-6141-2025-296-12.

A. V. Kizino, O. A. Novoselskaya
Belarusian State Technological University

ALGORITHMS AND SOFTWARE FOR KEY GENERATION FOR SUBSEQUENT ENCODING OF INFORMATION IN PROTECTIVE VECTOR IMAGES

Theoretical and practical aspects of algorithms and software for the generation of cryptographic keys for the subsequent encoding of information in protective vector images are considered. Such images include first of all guilloches, which are vector images and are reproduced by special printing methods. However, for implementation the information into electronic documents it is necessary to encode information. The basic functions for realization the information coding in protective vector images are described in the article. Attention is paid to the formation of a cryptographic container ensuring the integrity and confidentiality of information using a combination of symmetric and asymmetric encryption algorithms, as well as hash functions. The disadvantages of centralized key storage systems are considered and the risks associated with their compromise are shown. As an alternative, an approach similar to electronic signature is proposed, which meets the requirements of the legislation of the Republic of Belarus. The algorithm of key generation based on the analysis of user's mouse movement trajectory is described. The proposed method provides a high degree of uniqueness and cryptostability of the generated keys due to the use of biometric and behavioural parameters, which significantly complicates the selection or repeated reproduction of the key. The experimental part includes the development of a software tool with a visual interface, which allows to fix the mouse movement parameters and form an array of values converted

into a cryptographic key. Analysis of stability and repeatability of the results confirmed the reliability of the method. The obtained keys are integrated into the structure of the protective vector image, which allows not only to increase the level of protection, but also to realize the possibility of introducing hidden information into elements of corporate identity or electronic documentation.

Keywords: guilloche, key generation, vector graphics, information security, digital signature, RSA, random sequences, cryptography.

For citation: Kizino A. V., Novoselskaya O. A. Algorithms and software for key generation for subsequent encoding of information in protective vector images. *Proceedings of BSTU, issue 3, Physics and Mathematics. Informatics*, 2025, no. 2 (296), pp. 91–98 (In Russian).
DOI: 10.52025/2520-6141-2025-296-12.

Введение. Основу защиты документов, используемых в качестве ценных бумаг, составляют специальные штриховые изображения, которые воспроизводятся на бумаге специальными способами печати [1–4]. Это является средством защиты полиграфической продукции от фальсификации и злоупотреблений.

Предварительно проводился анализ состава и структуры подобных изображений. В качестве базовых элементов обычно используются штриховые изображения, представленные в виде простейших фигур или полиномов [5–8]. Как правило, их называют гильошами. Самым простым способом формирования гильошей является использование стандартных фигур или полиномов n-степени с заданными аффинными преобразованиями (масштабирования, вращения, переноса). В дальнейшем их воспроизводят специальными красками и специальными видами печати.

Анализ структуры гильошей показал, что они хорошо воспроизводятся средствами векторной графики. Дальнейшее изучение параметров гильошей [3, 9, 10] позволило сформировать идею векторного кодирования информации за счет управления параметрами векторного изображения, такими как вид и толщина линий, их частота и расстояние между ними. Наиболее распространенными технологиями векторного кодирования на текущий момент являются системы European Article Number (EAN) и Quick Response code (QR) [11, 12]. В этих системах используются параметры штрихов как элементов внедрения закодированной информации. Существенным отличием предлагаемого метода является внедрение уникального ключа, который должен генерироваться случайным или псевдослучайным образом. Причем генерируемый ключ должен быть персонифицированным для конкретного пользователя. Данная идея и послужила основой для поиска вариантов генерации ключа при создании защитных векторных изображений.

Защитные векторные изображения будут строиться на основе сгенерированной случайной последовательности чисел и их преобразовании в заданные параметры штрихов (вид и толщина линий, параметры штрихов и пробелов в штрихпунктирных линиях, при этом расстояние между линиями является фиксированным). Для задания вышеуказанных параметров необходимо сформировать персонализированный ключ кодирования, который в дальнейшем будет являться криптографическим контейнером, обеспечивающим как целостность данных, так и их конфиденциальность.

Современные подходы к защите таких контейнеров подразумевают комбинацию симметричных и асимметричных алгоритмов шифрования с использованием хэш-функций для верификации. В работе [13] рассмотрены теоретические аспекты генерации ключей по алгоритмам AES (Advanced Encryption Standard), Data Encryption Standard (DES), RSA, алгоритмов электронных подписей El Gamal Signature Algorithm (EGSA), Digital Signature Algorithm (DSA), Elliptic Curve Digital Signature Algorithm (ECDSA), алгоритмы хэширования Secure Hash Algorithm (SHA) и др. Симметричные алгоритмы (AES, DES) используют единый ключ для шифрования (дешифрования), обеспечивая высокую скорость обработки (до 1 Гбит/с для AES-256). Однако их главный недостаток – необходимость безопасной передачи ключа, что создает уязвимости в распределенных системах. Классический RSA, несмотря на существенные вычислительные затраты и требование к длине ключа (рекомендуется более 2048 бит), остается наиболее распространенным решением благодаря простоте интеграции и поддержке в стандартах (PKCS#1, X.509) [14]. Его безопасность основана на сложности факторизации больших чисел: для модуля $N = pq$, где p и q – 1024-битные простые числа, время взлома методом решета числового поля оценивается в $>10^{100}$ операций.

Хэш-функции (SHA-256, SHA3-512) [13, 15] играют критическую роль в обеспечении целостности. Для защиты информации и сохранения целостности данных принято использовать хэш-функции (SHA-256, SHA3-512), которые преобразуют данные в уникальную последовательность фиксированной длины (256 бит для SHA-256). Их коллизионная стойкость (невозможность найти два разных сообщения с одинаковым хэшем) делает их незаменимыми в схемах электронной подписи.

Анализ архитектур ключевых систем показал, что централизованная генерация ключей создает высокую опасность изменения и передачи данных третьим лицам. Компрометация центра [16] приводит к потере и краже данных. Вместо этого предложена децентрализованная двухключевая система, аналогичная механизмам электронной подписи, согласно Закону Республики Беларусь № 113-З [17]. В такой системе: закрытый ключ генерируется и хранится локально на устройстве пользователя; открытый ключ распространяется через доверенный реестр; сеансовый ключ для симметричного шифрования контейнера создается при помощи алгоритма Диффи – Хеллмана. Этот подход, устраняющий риски, связанные с транспортировкой и хранением ключей, соответствует требованиям белорусского законодательства к усиленной квалифицированной электронной цифровой подписи.

Целью настоящего исследования является разработка метода генерации ключа на основе движения мыши, а также оценка эффективности предложенного метода.

Методическая часть. Классический алгоритм RSA основывается на вычислительной сложности факторизации больших чисел [18]. На данный момент он является наиболее часто используемым для генерации числовой последовательности. Максимальное распространение он получил в области передачи защищенных изображений. Однако RSA уязвим к атакам при некорректной генерации простых чисел, а централизованное хранение публичных ключей создает риски компрометации. Также с улучшением технологий и наращиванием вычислительных мощностей возрастает опасность взлома такого ключа путем подбора начальных значений, поэтому в настоящее время требуемая длина ключа возросла с 1024 до 2048 бит. В свою очередь, увеличение длины ключа повышает требования к вычислительной мощности компьютерных систем, что приводит к снижению производительности системы и ограничивает применение данного метода для всех пользователей. Поэтому в работе ставится задача разработки метода генерации ключа, который позволит воспроизводить последовательность, регулируемую по длине и уникальную для каждой генерации.

Поскольку в электронных подписях документов используется поведенческие особенности человека по траектории движения мыши, в работе решено также основываться на данном подходе.

Существует несколько подходов к расчету траектории движения мыши (на основе эллиптических кривых, угловой меры и др.). Использование фиксированной угловой меры и скорости перемещения курсора с заданной частотой считывания позволит зафиксировать координаты курсора и преобразовать их в числовые объекты. Для расчета угловой меры используются классические формулы расчета угла между векторами. Угол между векторами θ рассчитывается по тригонометрической формуле [19]

$$\theta = \arctan(\Delta y / \Delta x). \quad (1)$$

Определение скорости перемещения v осуществляется по формуле [20]

$$v = \sqrt{\frac{\Delta x^2 + \Delta y^2}{\Delta t}}. \quad (2)$$

Считывание координат положения курсора в единицу времени требует подбора оптимальных параметров интервала и продолжительности считывания, а также размера поля ввода. Для получения массива уникальных объектов необходима фильтрация полученных значений.

Основная часть. Генерация случайной последовательности чисел на основе траектории движения мыши пользователем позволяет создать ключ, уникальный для каждого человека. Его сложно подобрать и невозможно повторить.

Расчет по формулам (1), (2) включает следующие пространственно-геометрические параметры, описывающие перемещение указателя мыши по заданному полю экрана:

– пространственные координаты указателя мыши в определенные моменты времени;

– угол (направление) движения указателя мыши между фиксируемыми координатами;

– скорость движения указателя мыши между фиксируемыми координатами.

Для алгоритмической реализации указанных параметров в методе генерации ключа необходимо выполнить следующие этапы.

1. На начальном этапе необходимо определить параметр T – время генерации ключа, а также случайным образом выбрать число n – временной интервал, с которым будет фиксироваться положение курсора мыши.

2. Далее каждые n секунд записывать положение курсора в границах экранных координат x и y, пока не истечет T секунд.

3. После получения массива точек массив необходимо подвергнуть фильтрации, в результате которого удаляются все повторяющиеся значения.

4. На втором этапе необходимо рассчитать дополнительные параметры, такие как угол (Arc) и скорость (Speed) перемещения курсора. В отличие от координат x и y, которые являются случайными числами, параметры Arc, Speed генерируются на основе предыдущих значений.

5. Полученный массив объектов необходимо снова отсортировать и отфильтровать для вычленения и удаления повторяющихся элементов.

В результате имеем уникальный набор объектов. Количество генерируемых объектов зависит

от ряда факторов, таких как выбранный интервал для фиксации, скорость передвижения мыши, процент охвата рабочей области, время генерации, технические характеристики устройства ввода и время отклика системы. А также размер поля для считывания позиций курсора. Генерация для определения ее оптимальных размера поля и времени проводилась в разных размерах границ областей, паттернах поведения пользователя и времени генерации. В результате исследования проведено 120 генераций для различных комбинаций соотношений времени и размера области. Результаты представлены в таблице.

Таблица

Подбор оптимальных параметров для генерации ключа

Время, с / размер, px	500×500	750×750	1000×1000
10	*76*	*191*	*195*
20	*102*	*275*	*399*
30	*151*	*389*	*756*
60	*274*	*892*	*1094*

Удовлетворительной считается генерация, в результате которой после всех операций и очисток получается массив значений объемом от 250 до 400 значений. Меньший объем точек может быть недостаточным для последующего использования, а больший 400 – избыточным.

В результате установлено, что соотношение, в котором размер области 750×750 px и время генерации 30 с, а также соотношение, в котором размер области 1000×1000 px и время генерации 20 с, являются наиболее предпочтительными, так как в среднем выдают значения, удовлетворяющие выдвинутым выше условиям. Однако вторая пара значений (размер области 1000×1000 px и время генерации 20 с) требует меньших временных затрат.

Для определения стабильности генерации был проведен дополнительный ряд генераций в ранее определенных условиях. Результаты генераций случайных последовательностей объектов представлены на рис. 1.

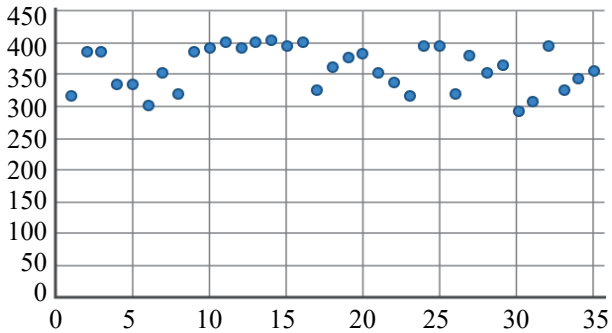

Рис. 1. Распределение числа отфильтрованных значений по номеру генерации

На рис. 1 показано, что после фильтраций все полученные последовательности лежат в диапазоне 300–400 объектов, а это является достаточным для формирования ключа. Однако для удовлетворительной генерации необходимо соблюдение ряда требований к пользователю:

– передвижение мыши должно происходить непрерывно на протяжении всего времени генерации;

– не допускается остановка мыши на одной точке длительное время, так как это влечет за собой генерацию одинаковых точек, которые будут удалены, тем самым длин сгенерированной последовательности будет недостаточно;

– необходимо использовать всю область генерации в равном объеме, постоянно перемещая курсор в различных направлениях, захватывая углы и края области;

– по возможности стараться не выходить за пределы рабочей области;

– использовать устройства ввода с высокой или средней чувствительностью, отдавать предпочтение проводным устройствам или беспроводным устройствам новых моделей.

При соблюдении данных требований рассматриваемый метод показывает наилучшие результаты и представляет набор с удовлетворительным количеством уникальных объектов.

Программная реализация приложения для генерации ключа потребовала внедрения ряда функций. Запуск процесса генерации осуществляется функцией `start()`, которая определяет интервалы получения точек и запускает процесс считывания движений пользователя (листинг 1).

```
   function start(id) {
if (flag == false) {

  document.getElementById(id)
    .style.display = "none";

  flag = true;
  time_i = getRandomNumber(25, 50);

  let timerId = setInterval(() =>
    add_arr(),
    time_i);

  setTimeout(() => {
    clearInterval(timerId);

    removeusingSet(arr);
    reapeat();
    create_char();
  },
   20000); }}
```

Листинг 1. Фрагмент кода функции `start()`

В листинге 1 интервал для считывания `time_i` выбирается псевдослучайным образом в промежутке (25; 50) мс. Непостоянный интервал позволяет извлекать точки из траектории движения

на различных участках, даже если движения были повторяющимися. Время генерации в 20 000 мс или 20 с определено опытным путем исходя из размера поля 1000×1000 px.

Функция `coordinate(event)` принимает в себя параметр `event`, который отвечает за значение положения мыши в текущий момент. При наличии объекта с таким же набором координат он не записывается. Функция `new_coord(obj)` принимает сгенерированный объект `obj`, в который входят значения *x* и *y*. Эта функция переводит данные значения в декартову систему координат для последующих вычислений. Функция `find_ang(obj1, obj2)` принимает два объекта после преобразования и находит угол между двумя векторами по формуле (1). Результат вычисления записывается в параметр `Arc` первого объекта. Функция `speed(obj1, obj2)`, подобно функции `find_ang(obj1, obj2)`, принимает на вход два объекта и возвращает скорость перемещения курсора между двумя зафиксированными точками при использовании формулы (2).

В результате генерации создается массив объектов с полями, представленными в численном виде. При необходимости генерации случайной последовательности используется функция `create_str(arr)`. Функция объединяет объекты в шестнадцатеричную строку и сохраняет ключ в JSON формат. На рис. 2 показан результат вывода полученной строки.

{"key":"416091ed2572e68a97385372f5e5878544decc066383ecd36d4c785f1b196f9ee49874
67da4e1c3b155d9f5ed9c1d08f8a44d1763fecefd8ff97b0600ff26172754a6a64405590dcb419
90f3a5e128a7fec868d13ec2fa001e9ee59dbb39994ce637fb4ff45c558593376decc6801fd8d3
8eff008e1f28f2a1b021bb3670b188d8843764b74f545986f3197b657b1437a7865a05113fc07f
cf793c06e34d84a13323095731f1a6a448101d90468bcf38bccc7a1ccb23dd484b8addbe7f7ebd
e893ecdb73969c1faead519618b27486d11aa75a72bcd3867e4971d55771bb43111c1ed962e740
995c49ed6e7ed7246b1f08344f3aa07ff46d0fa6fcdb5467687a5a4a9b7a09ca77c11f60d19c57
2431024ec160ed97b30392c7f79b0f5f1d76212a26f7ac77f94bf25396c57b39c347156c92b8e3
98b6719248888fa685bc13ac5743245f0f042fcefce6802f1abd60692f1a9fe1a22edf03b8132d
3f2d3ebc178c50a60b41ef5371a65f174e52b5965a4b87a0b85a4a9937de3548f7fa77c4479269
b5b0e46f457a184138457a2558c745f1a72860462d6fc8bc462d802ff446e060ce1846e06f6477
acd9726cf12e856e7e2a1e2c4d30d902e032d6ad3d5e4efeabc2a46d8aac9f9d418d361218127c
5d392a90315bd7a7bcad5550f0a89c686326224a8153a525a370610547a76a18a79c906d561c04

Рис. 2. Сгенерированная последовательность в шестнадцатеричной кодировке

Созданный массив может изменяться в соответствии с требованиями конкретного приложения к ключу. Реализация генерации ключа в виде массива объектов позволяет преобразовать его в словарь для шифрования защитных векторных изображений, где требуется набор уникальных комбинаций параметров для характеризации линий. Для преобразования полученной последовательности в ключ используются дополнительные функции. Функция `find_num_dots(obj)` определяет тип линии, которую будет определять принятый функцией объект, и количество точек, которыми линия будет задаваться, в случае, если линия задана как пунктирная.

Для удаления повторяющихся элементов массива используется функция `check_arr(arr)`, которая принимает на вход полученный массив объектов после преобразований и возвращает новый массив (листинг 2).

```
     function check_arr(res) {
/*. . .*/
  start_arr = Array.from(
     new Set(
      start_arr.map(JSON.stringify)))
    .map(JSON.parse);
  for (let i = 0; i < start_arr
    .length; i++) {
    res[i].x = start_arr[i].x;
    res[i].type = start_arr[i].type;
    res[i].dots = start_arr[i].dots;
    res[i].dot_arr = start_arr[i]
      .dot_arr;
  }
  res = res.slice(0, start_arr.length);
  return res;}
```

Листинг 2. Фрагмент кода функции `check_arr()`

Данная функция проверяет и сохраняет во временный массив упрощенное представление объектов и помещает новый массив в коллекцию, в которой происходит очистка от дублирующих значений.

После всех преобразований данные формируются в ключ в формате словаря, где каждый объект соответствует букве, а каждое поле – параметр линии, которой данная буква будет кодироваться. За преобразование в ключ отвечает функция `save_key(res)`, принимающая на вход массив объектов и возвращающая файл в формате JSON (листинг 3).

```
function save_key(res) {
  const key = {
    characters: [],
    startLine: {
      "stroke": {
        "width": res[0].x,
        "type": res[0].type,
        "dashArray": res[0].dot_arr}
    },
    endLine: {
      "stroke": {
        "width": res[1].x,
        "type": res[1].type,
        "dashArray": res[1].dot_arr }
    }
  };
    key.characters.push(temp);
  }
  jsonString = JSON.stringify(key);
}
```

Листинг 3. Код функции `save_key(res)`

Массив `characters` является контейнером для хранения символов с их параметрами шифрования. Поля `startLine` и `endLine` – стартовая и конечная линии, которые шифруются как

равнозначные с остальными буквами алфавита. Поле `plainCharacter` содержит букву, к которой относятся объект с параметрами `cipherCharacter`. Подобным образом каждой букве присваиваются такие же параметры. Объем генерации позволяет создать объекты не только для основного массива букв, но и для дополнительных знаков и символов, которые впоследствии могут быть использованы как дополнительная мера защиты информации. После при помощи встроенной функции `JSON.stringify()` массив конвертируется в формат JSON.

После генерации и преобразований пользователь получает на свое устройство ключ в формате файла JSON. Данный ключ имеет специфическую структуру из-за особенностей строения приложения, в котором он впоследствии будет использован. Средняя длина ключа составляет 9680 бит, что более чем в 4,5 раза превосходит длину ключа в RSA.

Заключение. Для решения задачи генерации ключа, который обеспечит высокую уникальность, разработан метод генерации ключа на основе движения мыши. Установлено, что для получения массива значений объемом от 250 до 400 объектов необходимо задать область регистрации положения курсора 1000×1000 px и время генерации 20 с. Стабильность генерации обеспечивает повторяемость 80%. Для определения вероятности коллизий был проведен анализ сгенерированных последовательностей как до преобразования в ключ, так и после. Анализ показал, что из 120 сгенерированных ключей разной длины не было сгенерировано ни одного ключа, который бы совпал.

Предложенный метод показал высокую производительность по сравнению с методом RSA. Так, благодаря фиксированному времени генерации и асинхронной обработке значений, предложенный метод генерации ключа задействует меньшие вычислительные мощности по сравнению с методом RSA. На системах с 4-ядерными процессорами (тесты проводились на процессоре Intel i3-12100) рассмотренный метод задействовал максимально 10% CPU, в то время как RSA задействует максимально 38%, что соответствует полной загрузке как минимум одного ядра. Результаты сравнения времени генерации и загрузки CPU представлены на рис. 3. На графике показано, что предложенный метод намного менее ресурсозатратный, что делает его эффективным для использования в системах с небольшими вычислительными мощностями в сравнении с RSA.

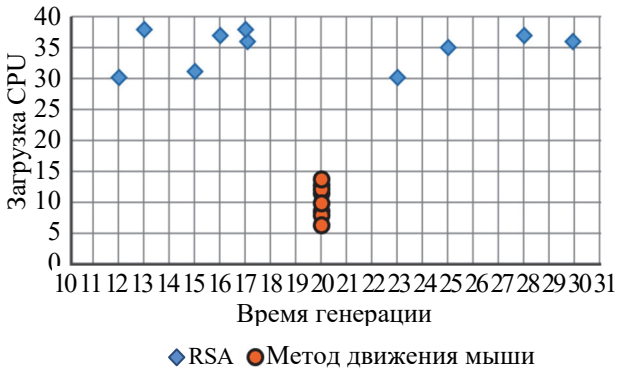

Рис. 3. Сравнение времени генерации и загрузки CPU для RSA и предложенного метода

Метод позволяет корректировать длину последовательности исходя из требований систем, для которых генерируется ключ, тем самым можно избежать потерь вычислительных мощностей и ресурсов, которые потребуются для генерации избыточного ключа другими методами.

Список литературы

1. Richard D. Warner, Richard M. Adams. Introduction to Security Printing. Berkeley, Portland: Graphic Arts Center Publishing Company, 2005. 115 p.

2. Thinger Ch., Paliwal K. Guilloche Pattern for Security in Confidential Documents // International Journal of Innovative Science, Engineering & Technology. 2016. Vol. 3, issue 10. P. 376–378.

3. Алгоритмы и программное средство для генерации защитных изображений печатных документов / О. А. Новосельская [и др.] // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2022. № 1 (254). С. 64–72. DOI: 10.52065/2520-6141-2022-254-1.

4. Liakhovych O., Riznyk V. Guilloche as a special kind of printed documents protection // XII International PhD Workshop OWD 2010, 23–26 October 2010. Wisla, 2010. P. 57–60.

5. Stepien P.J., Gajda R., Marszałek, A. Guilloche in diffractive optically variable image devices. // Optical security and counterfeit deterrence techniques: II, SPIE conference series. 1998. Vol. 3314. SPIE, 4. P. 231–236.

6. Deineko Zh., Shakurova T., Lyashenko V. Guilloche rosette as an element of building complex geometric structures // Journal of Universal Science Research. 2023. Vol. 1 (10). P. 526–534. URL: https://universalpublishings.com/index.php/jusr/article/view/2269 (дата обращения: 10.04.2025).

7. Abu-Jassar A., Manakov V., Al-Abdallat A. Features of the Formation of Guilloche Rosettes // Journal of Universal Science Research. 2023. Vol. 1 (12). P. 129–138. URL: https://universalpublishings.com/index.php/jusr/article/view/3175 (дата обращения: 18.04.2025).

8. Liakhovych O., Riznyk V. Investigation of information technologies for creating guilloches // XIII International PhD Workshop OWD 2011, 22–25 October 2011. Wisla, 2010. P. 97–100.

9. Glissando // Web-site software product Glissando. URL: http://www.banknotes.ru/glissando.html (дата обращения: 25.04.2025).

10. Spirograph Guilloche Generator // Web-site Timestretch. URL: https://www.timestretch.com/2020/05/10/spirograph_guilloche_generator.html (дата обращения: 25.04.2025).

11. Roger C. Palmer. The bar code book. Peterborough, Helmers Publishing, Inc. Formerly North American Technology, Inc., 1995. 386 p.

12. GS1 General Specifications Standard. Release 25.0, Ratified, Jan 25. URL: https://ref.gs1.org/standards/genspecs/ (дата обращения: 05.05.2025).

13. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography. Boca Raton, CRC Press, 1996. 810 p.

14. Silverman R. D. A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths: RSA Laboratories Bulletin No. 13, April 2000. URL: http://www.rsasecurity.com.rsalabs/bulletins/ (дата обращения: 05.05.2025).

15. Belazzougui D., Botelho F. C., Dietzfelbinger M. Hash, displace, and compress – Springer Berlin. Heidelberg, 2009. 17 p.

16. Duursma I. M., Park S. K. ElGamal type signature schemes for n-dimensional vector spaces // Cryptology ePrint Archive. 2005. Report no. 2005/123. P. 312–325.

17. Об электронном документе и электронной цифровой подписи: Закон Респ. Беларусь, 28.12.2009, №113-3 // Национальный правовой портал Республики Беларусь. URL: https://pravo.by/ document/ ?guid=3871&p0=H10900113 (дата обращения: 05.05.2025).

18. Rivest R. L., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems // Communications of the ACM. 1978. Vol. 21, no. 2. P. 120–126.

19. Grgurich R., Blair H. T. An uncertainty principle for neural coding: Conjugate representations of position and velocity are mapped onto firing rates and co-firing rates of neural spike trains // Hippocampus. 2020 № 30 (4). С. 396–421. DOI: 10.1002/hipo.23197. Epub 2020 Feb 17. PMID: 32065487; PMCID: PMC7154697.

20. Волков В. М., Проконина Е. В. Итерационная реализация разностных схем в методе фиктивных областей для эллиптических задач со смешанными производными // Журнал Белорусского государственного университета. Математика. Информатика. 2019. № 1. С. 69–76. DOI: 10.33581/2520-6508-2019-1.

References

1. Richard D. Warner, Richard M. Adams. Introduction to Security Printing. Berkeley, Portland, Graphic Arts Center Publishing Company Publ., 2005. 115 p.

2. Thinger Ch., Paliwal K. Guilloche Pattern for Security in Confidential Documents. *International Journal of Innovative Science, Engineering & Technology*, 2016, vol. 3, issue 10, pp. 376–378.

3. Novoselskaya O. A., Savchuk N. A., Shcherbakova A. N., Romanenko D. M. Algorithms and Software for Generating Protective Images for Printed Documents. *Trudy BGTU* [Proceedings of BSTU], 2022, no. 1 (254): Physics and Mathematics. Informatics, issue 3, pp. 64–72 (In Russian). DOI: 10.52065/2520-6141-2022-254-1.

4. Liakhovych O., Riznyk V. Guilloche as a special kind of printed documents protection. *XII International PhD Workshop OWD 2010, 23–26 October*, 2010, pp. 57–60.

5. Stepien P. J., Gajda R., Marszałek A. Guilloche in diffractive optically variable image devices. *Optical security and counterfeit deterrence techniques*: *II, SPIE conference series*,1998, vol. 3314. SPIE, 4, pp. 231–236.

6. Deineko Zh., Shakurova T., Lyashenko V. Guilloche rosette as an element of building complex geometric structures. *Journal of Universal Science Research*, 2023, 1 (10), pp. 526–534. Available at: https://universalpublishings.com/index.php/jusr/article/view/2269 (accessed 10.04.2025).

7. Abu-Jassar A., Manakov V., Al-Abdallat A. Features of the Formation of Guilloche Rosettes. *Journal of Universal Science Research*, 2023, 1 (12), pp. 129–138. Available at: https://universalpublishings.com/index.php/jusr/article/view/3175 (accessed 18.04.2025).

8. Liakhovych O., Riznyk V. Investigation of information technologies for creating guilloches. *XIII International PhD Workshop OWD*, *2011, 22–25 October*, 2011, pp. 97–100.

9. Glissando. *Web-site software product Glissando*. Available at: http://www.banknotes.ru/glissando.html (accessed 25.04.2025).

10. Spirograph Guilloche Generator. *Web-site Timestretch*. Available at: https://www.timestretch.com/2020/05/10/spirograph_guilloche_generator.html (accessed 25.04.2025).

11. Roger C. Palmer. The bar code book. Helmers Publishing, Inc. Formerly North American Technology, Inc., 1995. 386 p.

12. GS1 General Specifications Standard. Release 25.0, Ratified, Jan 25. Available at: https://ref.gs1.org/standards/genspecs/ (accessed 10.04.2025).

13. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography. Boca Raton, CRC Press Publ., 1996. 810 p.

14. Silverman R. D. A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths: RSA Laboratories Bulletin No. 13, April 2000. Available at: http://www.rsasecurity.com.rsalabs/bulletins/ (accessed 05.05.2025).

15. Djamal Belazzougui, Fabiano C. Botelho, Martin Dietzfelbinger. Hash, displace, and compress – Springer Berlin. Heidelberg, 2009. 17 p.

16. Duursma I. M., Park S. K. ElGamal type signature schemes for n-dimensional vector spaces. *Cryptology ePrint Archive*, 2005, Report no. 2005/123, pp. 312–325.

17. About an electronic document and an electronic digital signature. The Law of the Republic of Belarus № 113-3 from 28.12.2009. Available at: https://pravo.by/document/?guid=3871&p0=H10900113 (accessed 05.05.2025).

18. Rivest R. L., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978, vol. 21, no. 2, pp. 120–126.

19. Grgurich R, Blair H. T. An uncertainty principle for neural coding: Conjugate representations of position and velocity are mapped onto firing rates and co-firing rates of neural spike trains. *Hippocampu*s, 2020, no. 30 (4), pp. 396–421. DOI: 10.1002/hipo.23197. Epub 2020 Feb 17. PMID: 32065487; PMCID: PMC7154697.

20. Volkov V. M., Prakonina A. U. Iterative realization of finite difference schemes in the ficttious domain method for elliptic problems with mixed derivatives. *Zhurnal Belorusskogo gosudarstvennogo universiteta. Matematika i informatika* [Journal of the Belarusian State University. Mathematics and Informatics], 2019, no. 1, pp. 69–76. (In Russian). DOI:10.33581/2520-6508-2019-1.

Информация об авторах

Кизино Александра Валентиновна – магистрант факультета информационных технологий. Белорусский государственный технологический университет (ул. Свердлова, 13а, 220006, г. Минск, Республика Беларусь). E-mail: kizino@belstu.by

Новосельская Ольга Александровна – кандидат технических наук, доцент, доцент кафедры информатики и веб-дизайна. Белорусский государственный технологический университет (ул. Свердлова, 13а, 220006, г. Минск, Республика Беларусь). E-mail: novoselskaya@belstu.by

Information about the authors

Kizino Alexandra Valentinovna – Master's student, Faculty of Information Technologies. Belarusian State Technological University (13a Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: kizino@belstu.by

Novoselskaya Olga Aleksandrovna – PhD (Engineering), Associate Assistant Professor, the Department of Informatics and Web Design. Belarusian State Technological University (13a Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: novoselskaya@belstu.by

Поступила после доработки 12.05.2025