УДК 004.056.55 DOI: 10.33075/2220-5861-2020-1-125-134

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ХЕШИРОВАНИЯ В СИСТЕМАХ КОНТРОЛЯ ОКРУЖАЮЩЕЙ СРЕДЫ

Н.Л. Корепанова, Д.А. Ситников

Севастопольский государственный университет
РФ, г. Севастополь, ул. Университетская, 33
E-mail: nat270702@gmail.com

Представлены математические модели хеширования для защиты данных от несанкционированного доступа и исследование основных характеристик их эффективности. В процессе выполнения тестирования алгоритмов проводились замеры времени реализации на операции гомоморфных свойств и общего времени работы алгоритмов хеширования. Сделан вывод, что алгоритм SHA-2 имеет значительное преимущество по быстродействию по сравнению с алгоритмами MD6 и ГОСТ Р 34.11-2012.

Ключевые слова: информационная безопасность; защита данных; алгоритмы хеширования; хеш-функция.

Поступила в редакцию: 23.12.2019. После доработки: 19.02.2020.

Введение. Для наблюдения за параметрами среды, подверженной неблагоприятным техногенным и антропогенным воздействиям, их оценки и планирования природоохранных мероприятий используют системы экологического мониторинга. Различают глобальный (биосферный), региональный (геосистемный) и локальный (биоэкологический) мониторинг. Основу экологического мониторинга составляют автоматизированные информационные системы (АИС), которые предназначены для получения, хранения, обработки и оценки данных о состоянии атмосферы, недр, водных и наземных объектов. АИС представляет собой совокупность информационно-поисковой системы, автоматизированной системы обработки данных, прогнозно-диагностической системы и системы управления. Данные АИС получает по специальным каналам связи от датчиков, выполняющих экспедиционные, наземные, морские наблюдения, наблюдения из космоса. Одним из принципов построения таких систем является принцип безопасности, который предполагает защищенный обмен данными между всеми элементами системы [1]. Для обеспечения безопасности передаваемой и сохраняемой информации в состав АИС экологического мониторинга целесообразно включать систему защиты информации, в основе которой лежат алгоритмы хеширования.

Постановка задачи. Хеширование есть разбиение множества ключей (однозначно характеризующих элементы хранения и представленных, как правило, в виде текстовых строк или чисел) на непересекающиеся подмножества (наборы элементов), обладающие определенным свойством. Это свойство описывается функцией хеширования, или хеш-функцией, и называется хеш-адресом. Решение обратной задачи возложено на хеш-структуры (хеш-таблицы): по хеш-адресу они обеспечивают быстрый доступ к нужному элементу. В идеале для задач поиска хеш-адрес должен быть уникальным, чтобы за одно обращение получить доступ к элементу, характеризуемому заданным ключом (идеальная хеш-функция). Однако, на практике идеал приходится заменять компромиссом и исходить из того, что получающиеся наборы с одинаковым хеш-адресом содержат более одного элемента.

В связи с этим возникает проблема анализа быстродействия функций хеширования для различных алгоритмов. В данной статье рассматриваются три алгоритма – MD6, SHA-2 и ГОСТ Р 34.11-2012. В качестве критериев, по которым оценивается быстродействие, выступает время выполнения операций сложения,

время выполнения операций умножения и общее время выполнения алгоритма. Выбранные алгоритмы должны производить вычисление хеш-суммы входного сообщения. Таким образом, исходными данными исследования являются сообщения, для которых необходимо вычислить хеш-сумму, и алгоритмы, по которым будет происходить вычисление, а выходными данными являются хеш-сумма и описанные ранее характеристики быстродействия.

Математическая модель хеширования АИС. Хэш-функция может выступать примером шифрования в целях аутентификации (установлении подлинности автора и документа), так же ее называют односторонней функции или дайджест-функцией.

Функция хеширования (хэш-функция) представляет собой преобразование, на вход которого подается сообщение переменной длины x, а выходом является строка фиксированной длины h(x). Иначе говоря, хэш-функция h(x) принимает в качестве аргумента сообщение (документ) x произвольной длины и возвращает хеш-значение (хеш) H=h(x) фиксированной длины (рис. 1).

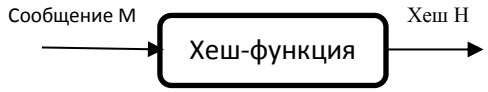

Рис. 1. Схема формирования хеш H = h(x)
Fig. 1. Scheme of the formation of hesh H = h(x)

Хеш-значение h(x) − это дайджест сообщения x, то есть сжатое двоичное представление основного сообщения x произвольной длины. Хеш-значение h(x) формируется функцией хэширования.

Функция хеширования позволяет сжать подписываемый документ x до 128 бит и более (в частности, 128 или 256 бит), тогда как x может быть размером в мегабайт или более. Следует отметить, что значение хэш-функции h(x) зависит сложным образом от документа x и не позволяет восстановить сам документ x.

Хеш-функция может применяться для проверки целостности данных. Хэш-функция, примененная к шифруемым данным, дает в результате значение (дайджест), состоящее из фиксированного небольшого числа байт. Дайджест передается вместе с исходным сообщением. Получатель сообщения, зная, какая хэш-функция была применена для получения дайджеста, заново вычисляет дайджест, используя незашифрованную часть сообщения. Если значение полученного и вычисленного дайджестов совпадают, значит, содержимое сообщения не было подвергнуто никаким изменениям. Значение дайджеста не дает возможности восстановить исходное сообщение, но зато позволяет проверить целостность данных [2].

Функция хеширования должна обладать следующими свойствами:

1. Хеш-функция может быть применена к аргументу любого размера.

2. Выходное значение хеш-функции имеет фиксированный размер.

3. Хеш-функцию h(x) достаточно просто вычислить для любого x. Скорость вычисления хеш-функции должна быть такой, чтобы скорость, выработки и проверки электронной цифровой подписи (ЭЦП) при использовании хеш-функции была значительно больше, чем при использовании самого сообщения.

4. Хеш-функция должна быть чувствительна к всевозможным изменениям в тексте x, таким как вставки, выбросы, перестановки и т.п.

5. Хеш-функция должна быть однонаправленной, то есть обладать свойством необратимости. Иными словами, задача подбора документа x, который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима.

6. Вероятность того, что значения хеш-функций двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала; то есть для любого фиксированного x с вычислительной точки зрения невозможно найти $x' \neq x$, такое что $h(x') = h(x)$.

Два различных сообщения могут быть сжаты в одну и ту же свертку (так называемая коллизия, или столкновение). Поэтому для обеспечения стойкости функции хеширования необходимо предусмотреть способ избегать столкно-

вений. Полностью столкновений избежать нельзя, поскольку в общем случае количество возможных сообщений превышает количество возможных выходных значений функции хеширования. Однако вероятность столкновения должна быть низкой.

Например, хеш-функция CRC-32, представляющая собой контрольную сумму, является линейным отображением и поэтому не удовлетворяет ни одному из этих трех свойств. Использование в качестве бесключевой хеш-функции, построенной на основе алгоритма блочного шифрования в режиме выработки имитовставки, также нецелесообразно, так как обратимость блочного шифрования позволяет подбирать входное сообщение для любого значения свертки при фиксированном и общеизвестном ключе. Для построения примера хеш-функции, удовлетворяющей свойству 1), рассмотрим функцию, заданную формулой $G_k(x) = E_k(x) \oplus x$, где E_k – алгоритм блочного шифрования; x – блок сообщения. Такая функция является однонаправленной, поэтому на ее основе можно построить хэш-функцию, определив одношаговую сжимающую функцию одной из следующих формул:

$$f(x, h) = E_k(x) \oplus x$$

или

$$f(x, h) = E_k(h) \oplus h.$$

Первая из этих функций лежит в основе российского стандарта хешфункции, а вторая – в основе американского стандарта SHA.

Рассмотрим конкретные примеры хеш-функций, построенных на основе некоторых алгоритмов преобразования блоков. Пусть E_k – алгоритм блочного шифрования, n – размер блока, l – размер ключа и G – некоторое отображение, ставящее в соответствие вектору длины n вектор длины l. Рассмотрим следующие одношаговые сжимающие функции, построенные на основе алгоритма E_k:

а) $f(x, h) = E_k(h) \oplus h$ (Дэвис – Мейер);

б) $f(x, h) = E_k(x) \oplus x$ (Матиас – Мейер – Оseac);

в) $f(x, h) = E_k(h) \oplus h \oplus E_k(x) \oplus x$ (Миагучи – Принель).

Значением любой из хеш-функций, из приведенных одношаговых сжимающих функций, является вектор длины n. В случае если эта величина оказывается недостаточной, ее можно увеличить, заменив одношаговую функцию f’ на функцию f с удвоенной размерностью значений. Это можно сделать, например, путем двукратного применения функции f с последующим перемешиванием полублоков согласно формуле:

$$f'(x,h1 ,h2) = \pi(f(x,h1),f(x, h2)),$$

в которой π переставляет произвольные полублоки a, b, c, d по правилу π ((a, b), (c, d)) = (a, d, c ,b) . Такой подход реализован в конструкции одношаговой функции MDC-2. Другие примеры бесключевых хеш-функций дают известные алгоритмы MD-4, MD-5 и SHA. Они оперируют с блоками длины n, совпадающей с длиной результирующего значения свертки, причем n = 128 для алгоритма MD-4 и n = 160 для MD-5 и SHA. Указанные алгоритмы спроектированы специально с учетом эффективной реализации на 32-разрядных ЭВМ. При их использовании исходное сообщение M разбивается на блоки длиной m = 512 бит. Последний блок формируется путем дописывания к концу сообщения комбинации 10...0 до получения блока размера 448 бит, к которому затем добавляется комбинация из 64 бит, представляющая битовую длину сообщения. Затем вычисляется значение свертки с использованием одношаговой сжимающей функции

В стандарте хеш-функции ГОСТ Р 34.11-94 приняты значения n = m = 512. Одношаговая сжимающая функция f (x, h), используемая для вычисления последовательности значений $h_i=f(x_i,h_{i-1})$, построена на базе четырех параллельно работающих схем блочного шифрования (ГОСТ 28147-89), каждая из которых имеет 256-битовый ключ и оперирует с блоками размера 64 бита. Каждый из ключей вычисляется в соответствии с некоторой линейной функцией от блока исходного сообщения x_i и значения h_{i-1}. Значение h_i является линейной функцией от результата шифрования, блока исходного сообщения x_i и значения h_{i-1}. После

вычисления значения h_N для последовательности блоков M_1, M_2, ..., M_N применяют еще два шага вычисления согласно формуле $h(M) = f(Z \oplus M_N, f(L, h_N))$, где Z – сумма по модулю два всех блоков сообщения, а L – длина сообщения.

Таким образом, функция хеширования может использоваться для обнаружения изменений сообщения, то есть она может служить для формирования криптографической контрольной суммы (также называемой кодом обнаружения изменений или кодом аутентификации сообщения). В этом качестве хеш-функция используется для контроля целостности сообщения, при формировании и проверке электронной цифровой подписи.

Хеш-функции широко используются также в целях аутентификации пользователей. В ряде технологий информационной безопасности применяется своеобразный прием шифрования – шифрование с помощью односторонней хеш-функции. Своеобразие этого шифрования заключается в том, что оно, по существу, является односторонним, то есть не сопровождается обратной процедурой – расшифрованием на приемной стороне. Обе стороны (отправитель и получатель) используют одну и ту же процедуру одностороннего шифрования на основе хеш-функции.

Известные функции хеширования:

– отечественный стандарт ГОСТ Р34.11-94. Вычисляет хеш размером 32 байт;

– MD (Message Digest) – ряд алгоритмов хеширования, наиболее распространенных в мире. Каждый из них вырабатывает 128-битовый хэш-код. Алгоритм MD2 – самый медленный из них, MD4 – самый быстрый;

– алгоритм MD5 является модификацией MD4, при которой пожертвовали скоростью ради увеличения безопасности. Алгоритм MD5 применяется в последних версиях Microsoft Windows для преобразования пароля пользователя в 16-байтовое число;

– SHA (Secure Hash Algorithm) – это алгоритм вычисления дайджеста сообщений, вырабатывающий 160-битовый хэш-код входных данных. Широко распространен в мире, используется во многих сетевых протоколах защиты информации.

Хеш-функции широко используются также для аутентификации пользователей. Существует множество криптографических протоколов, основанных на применении хеш-функций [3].

Применение хеш-функций в криптографии:

1) Проверка целостности данных при их передаче или хранении:

– с помощью ключевой хеш-функции;

– с помощью бесключевой хеш-функции (аутентификация источника–данных).

2) Проверка парольной фразы. В большинстве случаев парольные фразы не хранятся на целевых объектах, хранятся лишь их хеш-значения.

3) Ускорение поиска данных. Для осуществления быстрого поиска нужного сообщения в большом списке сообщений различной длины удобнее сравнивать друг с другом не сами сообщения, а короткие значения их сверток, играющих одновременно роль контрольных сумм. Основным требованием к таким хеш-функциям является равномерность распределения их значений при случайном выборе значений аргументов.

Хеш-функции также используются в некоторых структурах данных – хеш-таблица и декартовых деревьях.

Требования к хеш-функции в этом случае другие:

– хорошая перемешиваемость данных;

– быстрый алгоритм вычисления.

Хеш-функции имеют также разнообразные применения при проведении статистических экспериментов, при тестировании логических устройств, при построении алгоритмов быстрого поиска и проверки целостности записей в базах данных.

Алгоритм хеширования MD6. Алгоритм MD6 хеширования переменной разрядности предназначен для создания дайджестов сообщений переменной длины. С учетом требований устойчивости к дифференциальному криптоанализу осуществляется проверка подлинности

опубликованных сообщений, путем сравнения дайджеста сообщения с опубликованным. Таким образом, хеш-функция отображает входную двоичную строку произвольной длины в строку длиной d, при устойчивости к коллизиям, взлому, нахождению прообраза и псевдослучайности.

При применении хеш-функции размер обрабатываемого за один цикл блока данных составляет 512 байт, в то время как во многих других функциях этот параметр не превышает 512 бит. Такое решение позволяет существенно затруднить проведение некоторых видов атак. Особенностью алгоритма также является применение сжатия на основе структур деревьев. Функция сжатия находится в узле дерева, что является аналогом структуры дерева Меркла. Кроме того, для повышения производительности на низкоскоростных процессорах используются последовательные структуры вместо иерархических.

Такие конструктивные особенности, как использование нумерации узлов дерева, корень и z-биты на входе в подфункцию позволяют предотвращать атаки вставок и расширения. Всего при помощи трех простейших операций – сложение, сдвиг с константами и сложение по модулю 2 удается достигнуть нелинейности функции.

Алгоритм хеширования SHA-2. В алгоритме SHA-2 хеш-функции предназначены для создания отпечатков или дайджестов сообщений произвольной битовой длины на основе структуры Меркла-Дамгора. Структура Меркла-Дамгора – метод построения криптографических хеш-функций, предусматривающий разбиение входных сообщений произвольной длины на блоки фиксированной длины и работающий с ними по очереди с помощью функции сжатия, каждый раз принимая входной блок с выходным от предыдущего прохода. При использовании алгоритма SHA-2 считается, что если функция сжатия устойчива к коллизиям, то и хеш-функция будет также устойчива – чтобы доказать устойчивость структуры сообщение дополняется блоком, который кодирует длину первоначального сообщения (упрочнение Меркла – Дамгора).

Односторонняя функция сжатия преобразует два входных блока фиксированной длины в выходной блок того же размера, что и входные; алгоритм начинается с вектора инициализации IV, функция выполняется последовательно над результатом каждого предыдущего прохода.

Структура предусматривает вектор инициализации – фиксированное значение, которое зависит от реализации алгоритма, и которое применяется к первому проходу – применению функции сжатия к нему и первому блоку сообщения. Результат каждого прохода передается на следующий вход и очередному блоку сообщения. Последний блок дополняется нулями, если необходимо, а также, добавляется блок с информацией о длине целого сообщения. Для упрочнения хеша последний результат иногда пропускают через функцию финализации, которая может использоваться также для уменьшения размера выходного хеша сжатием результата последнего применения в хеш более маленького размера, или чтобы гарантировать лучшее смешивание битов и усилить влияние небольшого изменения входного сообщения на хеш (обеспечить лавинный эффект). Функция финализации часто строится с использованием функции сжатия.

При этом структура имеет несколько нежелательных свойств:

– атака нахождения второго прообраза для длинных сообщений всегда намного более эффективна, чем полный перебор. Атака для сообщения из 2k блоков может быть выполнена за время $2 \cdot kn/2 + 1 + 2n-k + 1$;

– множественные коллизии (много сообщений имеют одинаковый хеш) могут быть найдены лишь незначительно большими усилиями, чем коллизии;

– атаки дополнением сообщения: при данном хеше h(x) неизвестного входного сообщения легко найти значение h(pad(x)), где pad – функция дополнения; это значит, что возможно найти хеши входных сообщений.

Исходное сообщение после дополнения разбивается на блоки, каждый блок – на 16 слов. Алгоритм пропускает каждый блок сообщения через цикл с 64 или 80 итерациями (раундами). На каждой итерации 2 слова преобразуются, функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются, сумма является значением хеш-функции. Тем не менее, инициализация внутреннего состояния производится результатом обработки предыдущего блока. Поэтому независимо обрабатывать блоки и складывать результаты нельзя.

Алгоритм хэширования ГОСТ Р 34.11-2012. ГОСТ Р 34.11-2012 определяет алгоритм и процедуру вычисления хеш-функции для любой последовательности двоичных символов, которые применяются в криптографических методах обработки и защиты информации, в том числе для реализации процедур обеспечения целостности, аутентичности, ЭЦП при передаче, обработке и хранении информации в автоматизированных системах.

Функция хеширования ГОСТ Р 34.11-2012 используется при реализации систем электронной цифровой подписи на базе ассиметричного криптографического алгоритма. Семейство хеш-функций Стрибог состоит из двух хеш-функций с длинами результирующего значения в 256 и 512 бит, которые отличаются начальным внутренним состоянием и его частью, принимаемой за результат вычислений. Входное сообщение разбивается на блоки фиксированного размера, для сообщений размером не кратным длине блока используется дополнение. Начальное внутреннее состояние хеш-функции обновляется последовательной обработкой блоков сообщения функцией сжатия. Параллельно с этим вычисляются число обработанных бит и контрольная сумма блоков. После всех блоков сообщения функция сжатия обрабатывает блок с общей длиной сообщения и блок с контрольной суммой для завершения вычисления значения хеш-функции. Размер блоков сообщения и внутреннего состояния хеш-функции составляет 512 бит. Основное отличие хеш-функции Стрибога от своего предшественника – функция сжатия.

Основная операция функции сжатия обозначается как LPS и состоит из трёх преобразований: подстановки на байтах, транспонирования матрицы байт и умножения 64-битных векторов на матрицу 64 × 64 в GF(2):

1. S – нелинейная биекция. 512 бит аргумента рассматриваются как массив из шестидесяти четырех байт, каждый из которых заменяется по заданной стандартом таблице подстановки;
2. P – переупорядочивание байт. Байты аргумента меняются местами по определенному в стандарте порядку;
3. L – линейное преобразование. Аргумент рассматривается как 8 64-битных векторов, каждый из которых заменяется результатом умножения на определённую стандартом матрицу 64 × 64 над GF(2).

Переупорядочивание байт P, определённое стандартом, является операцией транспонирования матрицы байт размером 8 × 8.

В функции сжатия используются только преобразование LPS и побитовое исключающее ИЛИ над 512-битными блоками. Вместе со сложением по модулю 2^512 они составляют полный набор операций, использующихся в функции хеширования ГОСТ Р 34.11-2012.

Результаты исследований алгоритмов хеширования. Для исследования алгоритмов хеширования была создана программная система на языке программирования Java. Оценка быстродействия алгоритмов проводилась по трем критериям: общее время работы алгоритма; время, потраченное на выполнение операций сложения; время, потраченное на выполнение операций умножения. Каждый алгоритм тестировался на одном и том же массиве данных, состоящем из 15 одинаковых входных наборов (сообщений) различной длины, для которых вычислялось хеш-значение. На вход системы подавалось входное сообщение, выбралась определенная хэш-функция, при этом включался флаг вычисления характеристик быстродействия, В процессе выполнения

тестирования алгоритмов проводились замеры времени реализации на операции гомоморфных свойств (сложение и умножение) и общего времени работы алгоритма хеширования.

Результаты экспериментальных исследований, в которых менялось количество символов в тестируемых сообщениях, сведены в табл. 1.

На рис. 2–4 построены диаграммы зависимости каждого критерия от длины входного сообщения. Анализируя рис. 2 и рис. 3 можно сделать вывод о том, что суммарное время выполнения операций сложения и умножения в алгоритмах MD6 и ГОСТ Р 34.11-2012 составляет большую часть времени работы алгоритма, в то же время операции умножения выполняются дольше, чем операции сложения. Анализируя рис. 4 можно сделать вывод о том, что время выполнения операций сложения в алгоритме SHA-2 занимает малую долю от времени всего выполнения алгоритма. В данном алгоритме операция умножения отсутствует. Общее время выполнения алгоритма при этом достаточно мало за счет использования побитовых операций.

На рис. 5 отображена графическая зависимость времени, за которое выполняется работа всех алгоритмов хеширования от количества символов в текстовом файле.

Как видно из табличных данных и графиков, наименьшее время, затрачиваемое на выполнение всех алгоритмов, достигается у алгоритма SHA-2. Это объясняется тем, что данный алгоритм гораздо проще в реализации, и большинство вычислений выполняются путем побитовых операций, которые выполняются гораздо быстрее, чем такие операции как сложение и умножение. Алгоритм ГОСТ Р 34.11-2012 выполняется дольше, чем MD6, что объясняется более сложной структурой и количеством операций умножения.

При небольшом объеме текста алгоритм MD6 выигрывает по общему времени выполнения у алгоритма ГОСТ Р 34.11-2012 в сторону понижения от 19,3 до 14%. При большом объеме текста (от 4000 символов) эта разница составляет от 3,6 до 3,2% в сторону уменьшения. Что касается сравнения с алгоритмом SHA-2 то при небольшом объеме текста разница с алгоритмом MD6 составляет от 52% в сторону увеличения, а с алгоритмом ГОСТ Р 34.11-2012 от 61,3% в сторону увеличения. При большом объеме текста (от 4000 символов) эта разница составляет для обоих алгоритмов от 92 до 93% в сторону увеличения. Соответственно, можно сделать вывод, что алгоритм SHA-2 имеет значительное преимущество по быстродействию с алгоритмами MD6 и ГОСТ Р 34.11-2012, которое с увеличением объема текста возрастает.

Таблица 1. Результаты исследований

Количество символов	Алгоритм	Хеширование, мс	Гомоморфные свойства	
			Сложение, мс	Умножение, мс
100	MD6	873	362	331
	SHA-2	419	11	-
	ГОСТ Р 34.11-2012	1082	380	452
200	MD6	1080	390	492
	SHA-2	421	11	-
	ГОСТ Р 34.11-2012	1290	503	623
350	MD6	1278	507	581
	SHA-2	418	12	-
	ГОСТ Р 34.11-2012	1487	548	751
500	MD6	1685	696	799
	SHA-2	465	15	-
	ГОСТ Р 34.11-2012	1917	685	1032

650	MD6	1972	873	906
	SHA-2	445	17	-
	ГОСТ Р 34.11-2012	2194	787	1198
800	MD6	2016	941	994
	SHA-2	517	18	-
	ГОСТ Р 34.11-2012	2274	751	1308
1000	MD6	2599	1057	1168
	SHA-2	503	21	-
	ГОСТ Р 34.11-2012	2850	1104	1521
1500	MD6	3711	1461	1880
	SHA-2	522	30	-
	ГОСТ Р 34.11-2012	3972	1337	2368
2000	MD6	4309	1931	2242
	SHA-2	574	34	-
	ГОСТ Р 34.11-2012	4596	1410	2886
2500	MD6	5083	2395	2663
	SHA-2	561	36	-
	ГОСТ Р 34.11-2012	1573	3462	5363
3000	MD6	5627	2784	2810
	SHA-2	589	40	-
	ГОСТ Р 34.11-2012	5921	1832	3739
3500	MD6	6690	3048	3554
	SHA-2	615	43	-
	ГОСТ Р 34.11-2012	6997	2056	4571
4000	MD6	7848	3474	4003
	SHA-2	594	49	-
	ГОСТ Р 34.11-2012	8145	2597	5162
4500	MD6	8379	4024	4108
	SHA-2	605	54	-
	ГОСТ Р 34.11-2012	8681	2831	5450
5000	MD6	9085	4573	4868
	SHA-2	611	69	-
	ГОСТ Р 34.11-2012	9390	2584	6393

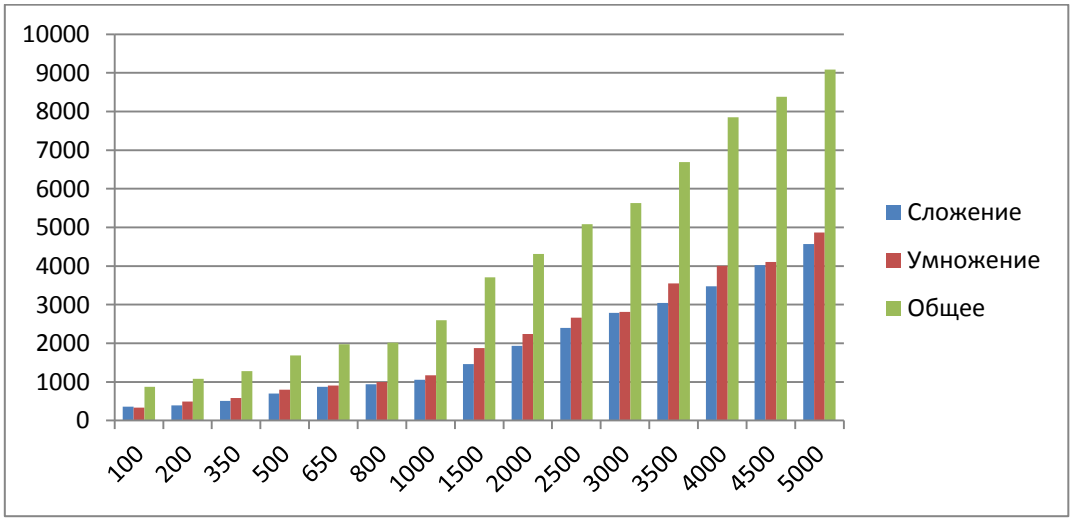

Рис. 2. Диаграмма времени работы алгоритма MD6

Fig. 2. MD6 algorithm runtime

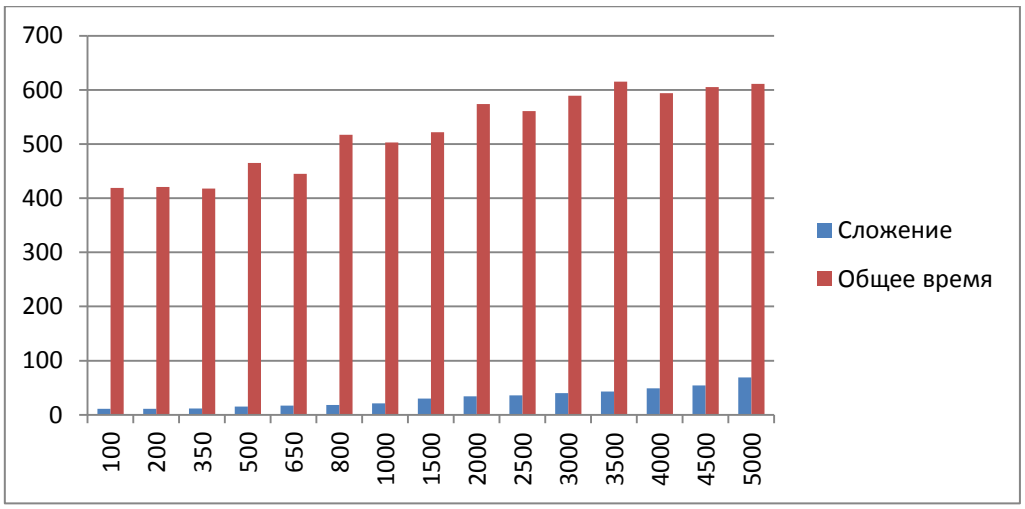

Рис. 3. Диаграмма времени работы алгоритма SHA-2
Fig. 3. SHA-2 algorithm runtime diagram

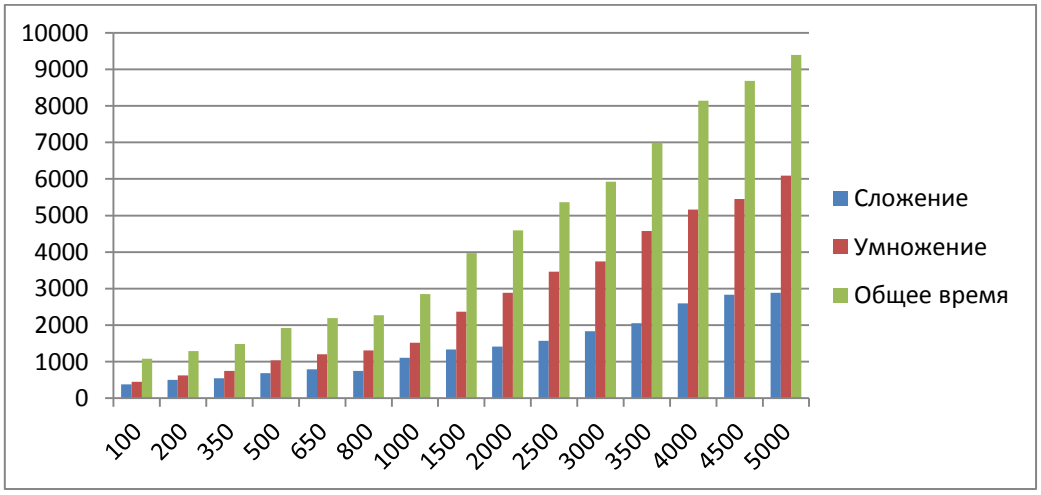

Рис. 4. Диаграмма времени работы алгоритма ГОСТ Р 34.11-2012
Fig. 4. The time diagram of the algorithm GOST R 34.11-2012

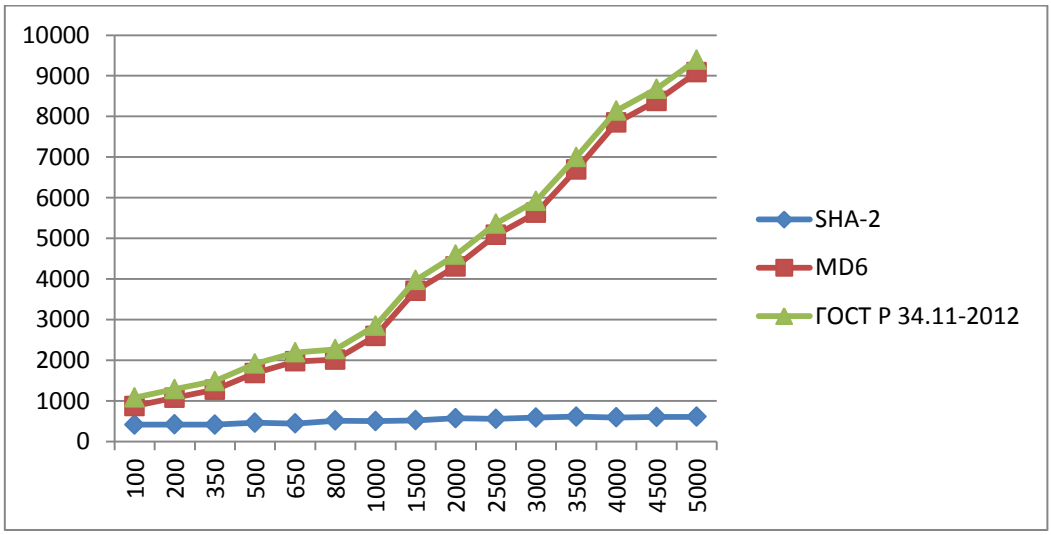

Рис. 5. Время, потраченное на реализацию алгоритмов
Fig. 5. The time spent on the implementation of the algorithms

Заключение. Использование алгоритмов хеширования позволит обеспечить безопасность передачи данных в системах экологического мониторинга. В связи с развитием вычислительной техники появление новых методов усиления безопасности передаваемых данных, таких как MD6, SHA-2, ГОСТ Р 34.11-2012, позволяет создать АИС с высоким уровнем защиты передаваемых данных. В дальнейшем предполагается исследовать алгоритмы защиты данных на основе нейронных сетей, что позволит усилить устойчивость систем экологического мониторинга.

СПИСОК ЛИТЕРАТУРЫ

1. *Корепанова Н.Л., Лебедева М.А., Павленко Л.О.* Исследование алгоритмов криптографии в системах контроля окружающей среды // Системы контроля окружающей среды. 2017. Вып. 8 (28). С. 55–64.

2. *Халимов Г.З.* Универсальное хеширование по максимальной кривой третьего рода // Научные ведомости БелГУ. Сер. История. Политология. Экономика. Информатика. 2011. № 1 (96). Вып. 17/1. С. 137–145.

3. *Халимов О.Г., Буханцов А.Д., Халимов Г.З.* Построение кривых Гурвица для универсального хеширования // Научные ведомости БелГУ. Сер. История. Политология. Экономика. Информатика. 2014. № 1 (172). Вып. 29/1. С. 153–160.

ANALYSIS OF HASHING ALGORITHMS FOR ENVIRONMENTAL CONTROL SYSTEMS

N.L. Korepanova, D.A. Sitnikov

Sevastopol State University,
RF, Sevastopol, Universitetskaya St., 33

Mathematical models of hashing to protect data from unauthorized access and study of the main characteristics of their effectiveness are presented. In the process of testing the algorithms, measurements were made of the implementation time for the operation of homomorphic properties and the total time of hashing algorithms. It is concluded that the SHA-2 algorithm has a significant performance advantage in comparison with the MD6 and GOST R 34.11-2012 algorithms.
Keywords: information security; data protection; hashing algorithms; hash function.