УДК 004.056.55 **DOI:10.30837/rt.2020.4.203.06**

М.С. ЛУЦЕНКО

ПОСТКВАНТОВЫЙ АЛГОРИТМ ИНКАПСУЛЯЦИИ КЛЮЧЕЙ CLASSIC MCELIECE

Введение

В 2016 г. Национальный институт стандартов и технологий США (англ. The National Institute of Standards and Technology, NIST) объявил конкурс, целью которого было выбрать алгоритмы постквантовой криптографии (англ. Post-Quantum Cryptography, PQC). Среди алгоритмов шифрования с открытым ключом и алгоритмов инкапсуляции ключей представлены в качестве финалистов третьего раунда: Classic McEliece [1], CRYSTALS-KYBER, NTRU, SABER. В качестве альтернативных финалистов в этой же группе алгоритмов были выбраны: BIKE [13], FrodoKEM, HQC [14], NTRU Prime, SIKE.

Эта версия алгоритма Classic McEliece была представлена на третьем раунде конкурса постквантовой криптографии 10 октября 2020 г. [7] международной группой разработчиков из Великобритании, США, Швейцарии, Нидерландов, Дании, Германии.

Алгоритм Classic McEliece является вариацией алгоритма на основе кодов, который был предложен Робертом Мак-Элисом (англ. Robert J. McEliece). Первая криптосистема с открытым ключом на основе кода была представлена в 1978 г. математиком и инженером в Калифорнийском технологическом институте Робертом Мак-Элисом [2]. В такой системе открытый ключ определяется случайным двоичным кодом Гоппы [3 – 6] (англ. Binary Goppa code). Шифртекст является кодовым словом, объединенным с некоторым вектором ошибок. Личный ключ обеспечивает эффективное декодирование: извлечение кодового слова из зашифрованного текста, выявление и удаление ошибок.

Система Мак-Элиса была разработана как односторонняя (англ. one-wayness against chosen-plaintext attacks, OW-CPA) [3 – 11], что означает, что злоумышленник не может эффективно найти кодовое слово из зашифрованного текста и открытого ключа, когда кодовое слово выбирается случайным образом. Уровень безопасности системы Мак-Элиса оставался на удивление стабильным, несмотря на десятки документов об атаках за 40 лет. Первоначальные параметры криптосистемы Мак-Элиса были разработаны только для $\left(2^{64}\right)$ безопасности, но система легко масштабируется до «избыточных» параметров, которые обеспечивают достаточный запас безопасности по сравнению с достижениями в компьютерных технологиях, включая квантовые компьютеры.

Система Мак-Элиса по времени существования сравнима с алгоритмом RSA [3 – 11]. Как известно, алгоритм RSA не имеет необходимой стойкости против постквантового криптоанализа, в то время как криптосистема Мак-Элиса должна обеспечивать такую стойкость.

Система Мак-Элиса потребовала огромного количества дополнительных доработок спустя годы. Некоторые из них повышают эффективность при сохранении безопасности: это включает в себя «двойное» шифрование с открытым ключом (англ. public-key encryption, PKE), предложенное Нидеррайтером (англ. Harald G. Niederreiter), ускорение программного обеспечения и аппаратных составляющих системы.

Кроме того, теперь известно [3 – 11], как эффективно преобразовать OW-CPA алгоритмы шифрования с открытым ключом в алгоритмы инкапсуляции ключей (англ. key encapsulation mechanisms, KEM), защищенные от атак типа IND-CCA2 на основе модели случайного оракула (англ. random oracle model, ROM). Это преобразование является жестким, сохраняя уровень безопасности, при двух допущениях, которым удовлетворяет криптосистема Мак-Элиса: во-первых, шифрование с открытым ключом является детерминированным (т.е. дешифрование восстанавливает всю использованную случайность); во-вторых, алгоритм не имеет ошибок дешифрования действительных зашифрованных текстов. Более того,

внесенные авторами модификации классического алгоритм обеспечивают аналогичную степень защиты для более широкого класса атак, а именно – атак на основе модели квантового случайного оракула (англ. quantum random oracle model, QROM). Риск того, что атака, специфичная для хэш-функции, может быть быстрее, чем атака ROM или QROM, устраняется стандартной практикой выбора хорошо изученной, "неструктурированной" хэш-функции с высоким уровнем безопасности.

Classic McEliece [1] – это вариация алгоритма Мак-Элиса, которая объединяет все лучшие наработки предыдущих лет воедино. Она представляет собой алгоритм инкапсуляции ключей с обеспечением безопасности против IND-CCA2 на очень высоком уровне, даже против квантовых вычислений. Алгоритм инкапсуляции ключей построен консервативно на основе шифрования на открытом ключе, что позволяет обеспечивать безопасности OW-CPA. А именно – основан на двойной версии алгоритма шифрования на открытом ключе Мак-Элиса, созданной Нидеррайтером, с использованием двоичных кодов Гоппы. Каждый уровень конструкции разработан так, чтобы системы, в которых был внедрен данный алгоритм с высокой вероятностью, обеспечивали долгосрочную безопасность в постквантовый период.

Другая работа включает вариации алгоритма Мак-Элиса, безопасность которых еще не была изучена достаточно тщательно. Например, во многих предложенных вариациях двоичные коды Гоппа заменяются другими семействами кодов, а криптография на основе решеток заменяет понятие «кодовое слово плюс случайные ошибки» на «точку решетки плюс случайные ошибки». Криптография на основе кодов и криптография на основе решеток являются двумя основными типами кандидатов, указанных в требованиях NIST [7, 8] относительно стандартизации постквантовых алгоритмов. В этом документе основное внимание уделяется точности классической системы Мак-Элиса, поскольку она была тщательно изучена.

1. Базовые определения

1.1. Криптография с открытым ключом

Криптосистема с открытым ключом [3 – 11] использует одностороннюю функцию с потайным входом E, которая будет служить функцией шифрования. Вычисление обратной E^{-1}, называемой функцией дешифрования, возможно, только если известен секрет K. Эта концепция односторонней функции формирует основу криптографии с открытым ключом, в которой секретным ключом является K, а открытым ключом – E. Точнее, криптосистема шифрования с открытым ключом должна обеспечивать три алгоритма: $KeyGen$, $Encrypt$ и $Decrypt$. $KeyGen$ – это вероятностный алгоритм с полиномиальным временем, который на входе получает $\{0,1\}^k$, где $k \geq 0$ – параметр безопасности, а на выходе формирует пару открытого / личного ключа (pk, sk). $KeyGen$ также определяет конечное пространство сообщений M_{pk}. $Encrypt$ – это вероятностный алгоритм с полиномиальным временем, который из входных параметров $\{0,1\}^k$, открытого ключа pk и сообщения $x \in M_{pk}$ формирует на выходе шифртекст c. $Decrypt$ – это детерминированный алгоритм с полиномиальным временем, который на основе входных данных $\{0,1\}^k$, sk и шифртекст c восстанавливает сообщение x. Криптосистема должна удовлетворять свойству правильности, что означает, что дешифрование должно однозначно восстанавливать зашифрованное сообщение.

1.2. Двоичный код Гоппы

Двоичный код Гоппы [3 – 11], обозначаемый $\Gamma(g(z), L)$, где $g(z)$ – порождающий полином Гоппы степени t над расширением поля $GF(2^m)$, где m – порядок расширения, а L – диапазон значений код, где $L \subseteq GF\{2^m\}$

$$g(z)=\sum_{i=0}^{t} g_i z^i = g_0 + g_1 z + ... + g_t z^t \quad (1)$$

$$L=\left\{\forall \alpha_i \in GF\left(2^m\right) \setminus g(\alpha_i)=0\right\} \quad (2)$$

с вектором c над $GF(2)$ таким, что $c=(c_1,c_2,...,c_n)$ и

$$R_e(z)=\sum_{i=1}^{t}\frac{c_i}{z-\alpha_i} \quad (3)$$

Если $n=2^m$ – длина кодового слова c, ограниченная диапазоном L, k – размерность, ограниченная $k>n-mt$, и минимальное расстояние $d\geq 2t+1$. Тогда $[n,k,d]$ представляет параметры кода Гоппы $\Gamma(g(z),L)$.

Для двоичного кода Гоппы любой сгенерированный многочлен $g(z)$ называется разделимым, если многочлен не имеет корней с кратностью больше единицы (то есть не имеет повторяющихся корней). Коды Гоппы, состоящие из таких полиномов, называются разделимыми двоичными кодами Гоппы. Для таких кодов существует хотя бы одно значение α_i при котором $g(\alpha_i)=0$.

Для двоичного кода Гоппы любой сгенерированный многочлен $g(z)$ называется неприводимым, если для $\forall \alpha_i$ справедливо $g(\alpha_i)\neq 0$. Коды Гоппы, состоящие из таких полиномов, называются неприводимыми двоичными кодами Гоппы.

Оба типа имеют возможность исправлять максимальное количество ошибок по сравнению с другими семействами кодов.

Минимальное расстояние d кода Гоппы должно быть больше $d\geq 2t+1$. Исправляющая способность кода оценивается параметром t.

Порождающая матрица G двоичного кода Гоппы используется для кодирования и декодирования сообщения, а контрольная сумма с проверкой на четность важна для обнаружения и исправления ошибок.

Порождающая матрица G получается из матрицы проверки на четность H, пространство строк G – это векторы нулевого пространства H по модулю 2, такие, что $GH^T=0$. Матрица проверки на четность H определяется как

$$H_{di}=\sum_{\alpha_i\in L}\sum_{k=1}^{t-(d-1)} g_{t-(k-1)}\alpha_i^{\,t-d+(1-k)}h_i \quad (4)$$

где $1\leq d\leq t$. Причем должно удовлетворяться условие $cH^T=0$, где c – шифротекст.

Кодирование сообщения в двоичном коде Гоппы [6]. Сообщение можно закодировать, разбив его на блоки длиной k бит и умножив каждый блок на порождающую матрицу G:

$$(m_1,m_2,...,m_k)\cdot G=(c_1,c_2,...,c_k) \quad (5)$$

Полученное сообщение (y) будет иметь вид

$$y=mG+e \quad (6)$$

***Алгоритм исправления ошибок неприводимого кода Гоппы* [6].**

1) Рассчитать синдром $S(z)$:

$$S(z)=\sum_{i=1}^{n}\frac{y_i}{z-\alpha_i} \quad (7)$$

2) Вычислить $\sigma(z)$, как показано ниже:

– Использовать расширенный алгоритм Евклида для определения $h(z)$, такого что

$$\sigma(z)h(z) \equiv 1 \pmod{g(z)} \tag{8}$$

Если $h(z) = z$, то процесс завершен и $\sigma(z) = z$, в противном случае продолжить:

– вычислить $d(z)$ так, чтобы

$$d^2(z) \equiv h(z) + z \pmod{g(z)} \tag{9}$$

– определить $a(z)$ и $b(z)$ такое, что

$$d(z)b(z) \equiv a(z) \pmod{g(z)} \tag{10}$$

– вычислить

$$\sigma(z) = a^2(z) + zb(z) \tag{11}$$

3) Рассчитать набор местоположений ошибки:

$$B = \{i \mid \sigma(\alpha_i) = 0\} \tag{12}$$

4) Вектор ошибок $e = (e_1, e_2, ..., e_n)$ определяется $e_i = 1$ для $i \in B$ и нулями в противном случае.

5) Сформированный шифротекст c имеет вид

$$c = y - e \tag{13}$$

Расшифровка сообщения [6]. Когда ошибки исправлены, сообщение может быть декодировано; закодированное сообщение в уравнении (6) можно представить в виде матрицы:

$$G^T \cdot \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \tag{14}$$

Для вычисления сообщения можно применить метод Гаусса, чтобы удалить порождающую матрицу $\lceil n/8 \rceil$:

$$\left(G^T \left| \begin{matrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{matrix} \right. \right) \square \ldots \square \cdot \left(\begin{array}{c|c} I_k & \begin{matrix} m_1 \\ \vdots \\ m_k \end{matrix} \\ \hline P & \end{array} \right) \tag{15}$$

где I_k _ это единичная матрица с размером $k \times k$, а p – это матрица с размером $(n-k) \times (k+1)$.

1.3. Свойство неразличимости шифротекста

К криптосистемам шифрования и инкапсуляции ключей с открытым ключом выдвигаются следующие требования [3 – 5, 8, 9]:

– семантическая безопасность (также известная как полиномиальная безопасность / неразличимость шифров): если злоумышленник имеет некоторую информацию об открытом тексте, он не может получить на основе этих знаний дополнительный данные о параметрах криптосистемы из шифротекста. Вычислительно невозможно различить два сообщения, одно из которых было зашифровано. Это означает, что схемы шифрования должны быть вероятностными;

– отсутствие гибкости. Из любого заданного зашифрованного текста злоумышленник не может получить новый зашифрованный текст таким образом, чтобы открытые тексты, лежащие в основе двух зашифрованных текстов, были значимо связаны. С другой стороны, злоумышленник может использовать множество видов атак: он может просто получить доступ к общедоступным данным, а затем зашифровать любой открытый текст по своему выбору (атаки на основе подобранного открытого текста) или, кроме того, использовать алгоритм дешифрования (атаки на основе неадаптивно / адаптивно подобранного шифротекста).

Криптосистема может быть семантически защищенной от атак с подобранным открытым текстом или даже атак с неадаптивным подобранным зашифрованным текстом (CCA1), при этом оставаясь гибкой. Однако защита от атак с адаптивно подобранным шифротекстом (CCA2) эквивалентна отсутствию гибкости.

Свойство неразличимости шифротекста определяет криптостойкость алгоритма к атаке на основе подобранного открытого текста. Обеспечение такого свойства неразличимости на основе открытого теста (IND-CPA) считается основным требованием для большинства доказуемо защищенных криптосистем с открытым ключом [3 – 5, 8, 9], хотя некоторые схемы также обеспечивают криптографическую стойкость против атак на основе подобранного шифротекста и адаптивных атак на основе подобранного шифротекста. Такие свойства неразличимости обозначаются, как IND-CCA1 и IND-CCA2 соответственно [3 – 5, 8, 9].

Для оценки уровня криптостойкости в классической и постквантовой криптографии для каждого алгоритма и его вариации используется обеспечиваемый уровень криптостойкости и соответствующие основные параметры преобразований. При описании требований к алгоритмам, подаваемым на конкурс, были определены три уровня криптографической стойкости [7, 8]:

– Уровень 1: любая атака, которая взламывает IND-CCA-стойкий алгоритм, должна требовать вычислительных ресурсов, сравнимых или превышающих требуемые для поиска ключа на блочном шифре с 128-битным ключом (например, AES-128);

– Уровень 3: если существует атака на IND-CCA-криптостойкий алгоритм, то для проведения такой атаки должны обеспечиваться вычислительные ресурсы, соизмеримые или превышающие требуемые для поиска ключа на блочном шифре с 192-битным ключом (например, AES-192);

– Уровень 5: любая атака, которая нарушает криптостойкость IND-CCA-стойкой схемы, должна требовать вычислительных ресурсов, сравнимых или превышающих требуемые для поиска ключа на блочном шифре с 256-битным ключом (AES-256).

1.4. Классическая и квантовая случайная модель оракула

За последние несколько лет, после многочисленных атак доказанная безопасность стала представлять наибольший интерес. Такое доказательство проводится в рамках теории сложности: предпринимаются попытки полиноминально свести хорошо установленную сложно решаемую проблему к атаке. Следовательно, эффективный злоумышленник не сможет решить сложную проблему: иначе это приведет к противоречию. Очень мало доказано схем с использованием только таких полиномиальных редукций без каких-либо других предположений. Кроме того, они с трудом достигают эффективности. В последние годы так называемая «случайная модель оракула» активизировала исследования, предоставив интересный инструмент для доказательства безопасности очень эффективных схем. Действительно, эта модель, в которой идеализированы некоторые конкретные криптографические объекты,

а именно хеш-функции, которые считаются действительно случайными, помогли обеспечить доказательства безопасности для многих схем шифрования и схем цифровой подписи.

Криптографическая хеш-функция – это функция $h:\{0,1\}^m \to \{0,1\}^n$ (где обычно $n<m$), которую трудно проанализировать. Интуитивно понятно, что желательно, чтобы единственный способ узнать информацию об $h(x)$ для любого конкретного $x \in \{0,1\}^n$ _ это фактически вычислить $h(x)$ и для каждого x, для которого h еще не вычислено, $h(x)$ должен выглядеть как новое случайное значение в $\{0,1\}^n$.

Модель случайного оракула – это способ формального моделирования этих интуитивных требований при анализе безопасности криптографических схем (таких как схемы шифрования или цифровой подписи), которые используют криптографическую хеш-функцию h как составную часть внутренней структуры. В модели случайного оракула хэш-функция h моделируется оракул, который необходимо запросить об $x \in \{0,1\}^m$, чтобы объект узнал хэш-значение $h(x)$, а случайный ответ оракула $h(x)$ относится к равномерно случайной функции $h:\{0,1\}^m \to \{0,1\}^n$.

Эта модель оказывается очень полезной для анализа безопасности криптографических схем. Например, это позволяет вести учет вычислительных возможностей злоумышленников, и поэтому возможно точно определить хеш-значения, которые доступны злоумышленнику. Тогда есть вероятность, что криптографическая схема, которая доказано безопасна в модели случайного оракула, будет доказуемо безопасна в принципе.

Оказывается, что модель случайного оракула необходимо пересмотреть в контексте квантовых атак, то есть когда рассматриваются злоумышленники, которые имеют в своем распоряжении неограниченные возможности проведения квантовых вычислений. Такая гипотетическая возможность позволила бы злоумышленнику вычислить суперпозиции хеш-функции h по нескольким x, то есть он мог бы создавать состояния, которые зависят от $\sum_x \alpha_x |x\rangle|h(x)\rangle$ для разных x, вычисляя h только один раз. Однако возможности злоумышленника относительно квантового криптоанализа ограничены рядом теорем о запрете (например, теоремой о запрете клонирования).

1.5. Криптосистема McEliece (1978)

Используя тот факт, что алгоритм быстрого декодирования существует для общего кода Гоппы, в то время как его не существует для общего линейного кода, в 1978 г. Робертом Мак-Элисом [2] было предложено создать криптосистему с открытым ключом, которая была бы достаточно безопасной, но в то же время обеспечивала чрезвычайно высокую скорость передачи данных. По мнению автора, такая криптосистема идеально подходит для использования в многопользовательских коммуникационных сетях, таких как те, которые предусмотрены НАСА для распространения данных, полученных из космоса, для передачи данных по незащищенным каналам передачи данных.

Криптосистема Мак-Элиса – это асимметричная криптосистема с открытым ключом, основанная на теории алгебраического кодирования. Криптосистема Мак-Элиса использует неприводимый двоичный код Гоппы, который до сих пор считался криптоустойчивым, особенно с параметром [1024, 524, 101], предложенным Мак-Элисом. Криптосистема с таким набором параметров страдает от большой матрицы открытых ключей, что затрудняет ее применение.

Криптосистема Мак-Элиса – одна из самых многообещающих криптосистем с открытым ключом, способная противостоять атакам на основе квантовых вычислений. Фактически, в отличие от криптосистем, использующих целочисленную факторизацию или дискретные логарифмы, криптостойкость данного алгоритма полагается на сложность декодирования полных линейных кодов (общая задача декодирования является NP-сложной).

После того, как Диффи и Хейлман ввели понятие криптосистемы с открытым ключом, в которой безопасность связи достигается без необходимости периодического распространения личного ключа отправителю и получателю. Это свойство делает такие системы идеально подходящими для использования в многопользовательских сетях связи.

Криптосистема Мак-Элиса состоит из трех основных алгоритмов:

– случайной генерации открытого и личного ключей;
– случайного шифрования открытого текста;
– детерминированного алгоритма расшифрования шифротекста.

Общесистемными параметрами для всех алгоритмов являются длина кода n, длина битовой последовательности открытого текста k, исправляющая способность кода t.

Каждому неприводимому многочлену степени t над $GF\left(2^m\right)$ соответствует двоичный неприводимый код Гоппы длины $n=2^m$, размерности $k \geq n-tm$ способный исправить любую комбинацию t ошибок или меньше. Более того, существует быстрый алгоритм декодирования этих кодов (алгоритм, предложенный Паттерсоном, время работы $\mathrm{O}(nt)$).

Предположим, что изначально в системе уже выбраны общесистемные параметры, а именно n и t, а также случайным образом выбран неприводимый полином степени t над $GF\left(2^m\right)$. Поскольку вероятность того, что случайно выбранный многочлен степени t является неприводимым, составляет около $1/t$, и поскольку существует быстрый алгоритм проверки неприводимости, предложенный Берлекампом, этот выбор будет легко сделать.

Генерация ключей

Входные параметры:

отсутствуют

Выходные параметры:

открытый ключ $\left(G', t\right)$,

личный ключ $\left(S, G, P\right)$

Алгоритм:

1) Сформировать $k \times n$ порождающую матрицу G для кода, которая может быть в канонической (например, приведённого ступенчатого вида), форме.
2) Сформировать случайную $k \times k$ невырожденную матрицу S.
3) Сформировать случайную $n \times n$ матрицу перестановок P.
4) Вычислить матрицу G' размером $k \times n$: $G' = SGP$, которая будет генерировать линейный код с той же скоростью и минимальным расстоянием, что и матрица G. Матрица G' будет общедоступной порождающей матрицей.
5) Сформировать наборы параметров в качестве открытого ключа $\left(G', t\right)$ и личного ключа $\left(S, G, P\right)$.

Схематически данный алгоритм представлен на рис. 1.

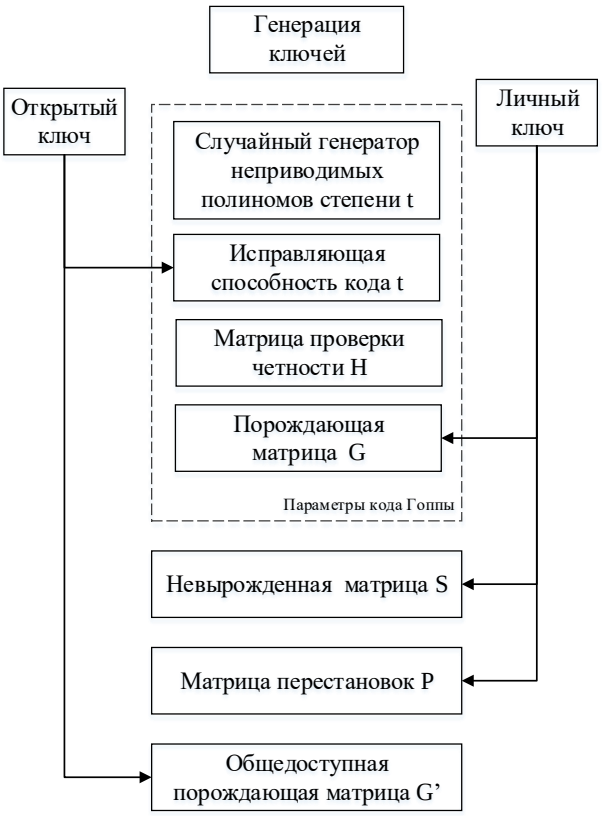

Рис. 1. Схема алгоритма случайной генерации открытого и личного ключей

Шифрование открытого текста

Входные параметры:

открытый текст m, общедоступная порождающая матрица G'.

Выходные параметры:

шифротекст c.

Алгоритм:

1) Представить открытый текст m в виде битовых последовательностей длины k.

2) Сформировать так называемый вектор ошибок – случайный вектор z длины n и веса t.

3) Вычислить шифротекст $c = mG' + z$.

Схематически данный алгоритм представлен на рис. 2.

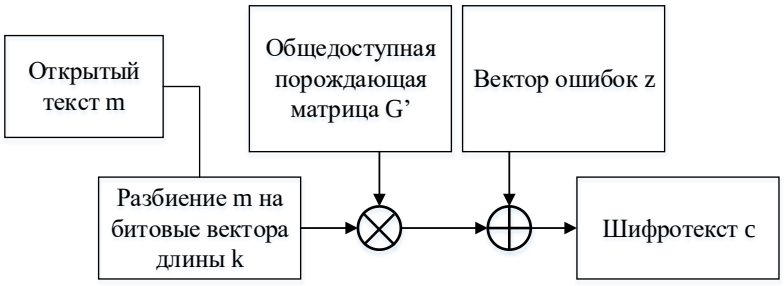

Рис. 2. Схема алгоритма шифрования открытого текста

Алгоритм расшифрования шифротекста

Входные параметры:

шифротекст c'.

Выходные параметры:

открытый текст m.

Алгоритм:

1) Вычислить обратную матрицу перестановок P^{-1}.
2) Вычислить $c' = cP^{-1}$.
3) Использовать алгоритм декодирования кодов Гоппы для восстановления m' из полученного c'.
4) Вычислить обратную матрицу S^{-1}, которая используется в алгоритме Паттерсона.
5) Вычислить открытый текст $m = m'S^{-1}$.

Схематически данный алгоритм представлен на Рис. 3.

Автором доказана корректность системы, т.е. правильность восстановления открытого текста после расшифрования $Decrypt\left(Encrypt\left(m\right)\right) = m$. Сформированный шифротекст имеет вид

$$c = mG' + z = mSGP + z \tag{16}$$

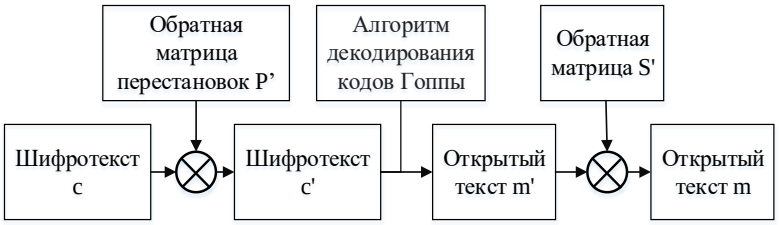

Рис. 3. Схема алгоритма расшифрования шифротекста

При расшифровывании формируется

$$c' = cP^{-1} = mSG + zP^{-1} \tag{17}$$

Как указывалось ранее, z имеет вес t, соответственно вес произведения zP^{-1} будет не более t.

Следовательно, при помощи кода Гоппы будет исправлено t ошибок. Расстояние Хемминга для составляющих шифротекста не превышает t, т.е. $d\left(mSG,\ cP^{-1}\right) \le t$, что доказывает соответствие исходного и расшифрованного текстов $m = m'S^{-1} = mSS^{-1}$.

2. Общая спецификация алгоритма Classic McEliece

2.1. Обозначения и параметры

Общесистемные параметры

n — Длина кода, $n \le q, n \in \mathbb{N}$

k — Размер кода, $k = n - mt$

t — Гарантируется возможность коррекции ошибок, $t \ge 2, mt < n$

q — Размер используемого конечного поля

m — $\log_2 q$, $m > 0, m \in \mathbb{N}$

μ — Неотрицательное целое число $\nu \ge \mu \ge 0,\ \nu \le k + \mu$

ν — Неотрицательное целое число

$f\left(z\right)$ — Неприводимый полином степени m, $f\left(z\right) \in F_2[z]$. Это определяет представление $F_2[z]/\ f\left(z\right)$ поля F_q

$F\left(y\right)$ — Неприводимый полином степени t, $F\left(y\right) \in F_q[y]$. Это определяет представление $F_q[y]/\ F\left(y\right)$ поля $F_{q^t} = F_{2^{mt}}$

Параметры симметричной криптографии

H — Криптографическая хеш-функция

l — Длина выхода криптографической хеш-функции, $l > 0$

σ_1 — Неотрицательное целое число, $\sigma_1 \geq m$

σ_2 — Неотрицательное целое число, $\sigma_2 \geq 2m$

G — Генератор псевдослучайных бит, который отображает строку из l бит в строку из $\left(n+\sigma_2 q+\sigma_1 t+l\right)$ бит

Параметры личного ключа

g — Многочлен в поле $F_q[x]$

α_i — Элемент конечного поля F_q

Γ — $\left(g,\alpha_1,...,\alpha_n\right)$

s — Битовая строка длины n

Параметры открытого ключа

T — Матрица из $\left(n-k\right)\times k$ элементов над полем F_2

Параметры шифротекста

e — Битовая строка длины n и веса Хэмминга t

C — Зашифрованный текст, содержащий сеансовый ключ

C_0 — Битовая строка длины $n-k$

C_1 — Битовая строка длины l

Элементы поля F_2^n, такие как кодовые слова и векторы ошибок, всегда рассматриваются как векторы-столбцы.

2.2. Выбор типа и размера параметров

Выбор параметров симметричной криптографии:

- l-битовая строка $H\left(x\right)$ определяется как первые l битов вывода $SHAKE256\left(x\right)$. Байтовые строки здесь рассматриваются как битовые строки с прямым порядком байтов.
- Длина битовой строки $l = 256$.
- Целое число σ_1 равно 16. (Во всех наборах параметров $m \leq 16$, поэтому $\sigma_1 \geq m$).
- Целое число σ_2 равно 32.
- $\left(n+\sigma_2 q+\sigma_1 t+l\right)$-битовая строка $G\left(\delta\right)$ определяется как первые $\left(n+\sigma_2 q+\sigma_1 t+l\right)$ вывода $SHAKE256\left(64,\delta\right)$. Здесь $64,\delta$ означает 33-байтовую строку, которая начинается с байта 64 и продолжается δ.

Все входы $H\left(x\right)$, используемые в криптосистеме Classic McEliece, начинаются с байта 0, 1 или 2 и, таким образом, не перекрывают входы $SHAKE256\left(x\right)$, используемые в G.

2.3. Представление параметров в виде байтовых строк

Векторы над полем F_2**.** Если r кратно 8, то r-битовый вектор $\upsilon = \left(\upsilon_0,\upsilon_1,...,\upsilon_{r-1}\right) \in F_2^r$ представляется в виде следующей последовательности из $r/8$ байтов:

$$\begin{pmatrix} \upsilon_0 + 2\upsilon_1 + 4\upsilon_2 + ... + 128\upsilon_7,\ \upsilon_8 + 2\upsilon_9 + 4\upsilon_{10} + ... \\ \qquad\qquad + 128\upsilon_{15},\ \upsilon_{r-8} + 2\upsilon_{r-7} + 4\upsilon_{r-6} + ... + 128\upsilon_{r-1} \end{pmatrix} \tag{18}$$

Если r не кратно 8, то r-битный вектор $\upsilon = \left(\upsilon_0,\upsilon_1,...,\upsilon_{r-1}\right) \in F_2^r$ дополняется 0 справа до длины между $r+1$ и $r+7$, в таком случае r станет кратно 8.

Сеансовые ключи. Сеансовый ключ K элемент поля F_2^l, представляется $\lceil l/8 \rceil$ байтовой строкой.

Шифротекст. Шифротекст C состоит из двух компонентов: $C_0 \in F_2^{n-k}$ и $C_1 \in F_2^l$. Шифротекст представлен как конкатенация $\lceil mt/8 \rceil$-байтовой строки, представляющей C_0 и $\lceil l/8 \rceil$-байтовой строки, представляющей C_1.

Хеш-входы. В алгоритме присутствуют три типа хеш-входов: $(2, \upsilon)$, $(1, \upsilon, C)$ и $(0, \upsilon, C)$. Здесь $\upsilon \in F_2^n$ и C представляет шифротекст.

Первые 0, 1 и 2 параметры представляются как байты. Вектор υ представляет собой следующие $\lceil n/8 \rceil$ байт. Если шифротекст присутствует, то он представляется как следующие $\lceil mt/8 \rceil + \lceil l/8 \rceil$ байт.

Открытые ключи. Открытый ключ T, который является матрицей $mt \times k$, представлен в виде строк. Каждая строка T представляется как $\lceil k/8 \rceil$-байтовая строка, и открытый ключ представлен как $mt\lceil k/8 \rceil$-байтовая конкатенация этих строк.

Элемент поля. Каждый элемент поля $F_q \cong F_2[z]/\, f(z)$ имеет форму $\sum_{i=0}^{m-1} c_i z^i$, где $c_i \in F_2$. Представление вектора поля это вектор $(c_0, c_1, ..., c_{m-1}) \in F_2^m$.

Неприводимые полиномы. Неприводимые полиномы $g = g_0 + g_1 x + ... + g_{t-1} x^{t-1} + x^t$ степени t представлены $t\lceil m/8 \rceil$ байтами, конкатенацией элементов поля $g_0, g_1, ..., g_{t-1}$.

Упорядоченное поле. Представление последовательности $(\alpha_1, ..., \alpha_q)$ из q различных элементов поля F_q будет последовательность q элементов поля.

Авторы описывают представление, которое упрощает быстрые алгоритмы декодирования с постоянным временем.

«Локальная сеть Бенеша» (англ. in-place Benes network) представляет собой серию из $2m-1$ этапов замены, применяемых к массиву из $q = 2^m$ объектов $(\alpha_0, \alpha_1, ..., \alpha_{q-1})$. Первый этап меняет местами α_0 и α_1, меняет местами α_2 и α_3 меняет местами α_4 и α_5 и т. д., как указано последовательностью из $q/2$ управляющих бит (1 означает замену мест, 0 означает отсутствие необходимости в замене). Второй этап меняет местами α_0 и α_2, меняет местами α_1 и α_3, меняет местами α_4 и α_6, и т.д., как указано следующими $q/2$ управляющими битами. Это продолжается до стадии m, на которой меняются α_0 и $\alpha_{q/2}$, меняются на α_1 и $\alpha_{q/2+1}$, и т.д. На $(m+1)$ этапе точно все происходит так же как и на $(m-1)$ этапе (с новыми управляющими битами), $(m+2)$-й этап аналогичен $(m-2)$-му этапу, и так далее вплоть до $(2m-1)$ этапа.

Определим π как перестановку $\{0,1,...,q-1\}$ такую, что $a_{i+1} = \sum_{j=0}^{m-1} \pi(i)_i \cdot z^{m-1-j}$ для всех $i \in \{0,1,...,q-1\}$. Порядок $(\alpha_1, ..., \alpha_q)$ представлен как последовательность из $(2m-1)2^{m-1}$ управляющих бит для локальной сети Бенеша для π. Этот вектор представлен в виде $\lceil (2m-1)2^{m-4} \rceil$ байт, как описано выше. Каждая перестановка имеет несколько вариантов управляющих битовых векторов. Чтобы упростить тестирование, требуется, чтобы переста-

новка π была преобразована конкретно в управляющие биты, определенные для π. Программные биты управления считыванием не проверяют уникальность.

В качестве простой защиты от ошибок при вычислении управляющих битов для π авторы рекомендуют разработчикам после вычисления проверять, что применение сети Бенеша дает π, и перезапускать генерацию ключа, если этот тест не прошел.

Выбор столбца. Часть личного ключа, сгенерированного функцией *KeyGen*, представляет собой последовательность $c=\left(c_{n-k-\mu+1},...,c_{n-k}\right)$ из μ целых чисел в порядке возрастания от $n-k-\mu+1$ до $n-k-\mu+\nu$. Эта последовательность c представлена как строка размером $\lceil \nu/8 \rceil$ байт, формат представления в порядке от младшего к старшему (обратном порядке байт):

$$\sum_{i=0}^{\mu-1} 2^{c_{n-k-\mu+1}-(n-k-\mu+1)} \tag{19}$$

Однако для $(\mu,\nu)=(0,0)$ последовательность c вместо этого представлена как 8 байтовая строка, которая является обратным порядком байтов $2^{32}-1$, т. е. 4 байта значения 255, за которыми следуют 4 байты значения 0.

Специальная обработка $(\mu,\nu)=(0,0)$ разработана таким образом, чтобы личный ключ, использующий $(\mu,\nu)=(0,0)$, был совместим с программным обеспечением декапсуляции, использующим $(\mu,\nu)=(32,64)$, когда все остальные параметры совпадают.

Личный ключ. Личный ключ (δ,c,g,α,s) представляет собой конкатенацию пяти частей:

- $\lceil l/8 \rceil$- байтовая строка $\delta \in F_2^l$.
- строка, представляющая выбор столбцов c. Эта строка имеет $\lceil \nu/8 \rceil$ байтов или 8 байтов, если $(\mu,\nu)=(0,0)$.
- $t\lceil m/8 \rceil$- байтовая строка полинома g.
- $\lceil (2m-1)2^{m-4} \rceil$ байт, представляющие поле порядка α.
- $\lceil n/8 \rceil$- байтовая строка $s \in F_2^n$.

3. Classic McEliece: алгоритм инкапсуляции ключей

3.1. Генерация неприводимого полинома

Входные параметры:

строка $d_{0,}d_{1,},...,d_{\sigma_1 t-1}$ из $\sigma_1 t$ бит.

Выходные параметры:

неприводимый полином $g \in F_q[x]$ степени t.

Алгоритм:

1) Определить $\beta_j=\sum_{i=0}^{m-1} d_{\sigma_1 j+iz^i}$ для каждого $j \in \{0,1,...,t-1\}$. В каждой группе σ_1 входных бит, используются только m первых бит. Алгоритм игнорирует оставшиеся биты строки.
2) Определить $\beta=\beta_0+\beta_1 y+...+\beta_{t-1}y^{t-1} \in F_q[y]/F(y)$.
3) Вычислить минимальный полином g из β над полем F_q. По определению g неприводимый полином, такой что $g(\beta)=0$.
4) Вернуть g если g имеет степень t. В противном случае вернуть ошибку выполнения.

3.2. Генерация упорядоченных полей

Входные параметры:

строка $\sigma_2 q$.

Выходные параметры:

последовательность $(\alpha_1, \alpha_2, ..., \alpha_q)$ из q различных элементов в поле F_q.

Алгоритм:

1) Берутся первые α_2 входных битов $b_0, b_1, ..., b_{\sigma_2 - 1}$ как σ_2-разрядное целое число $a_0 = b_0 + 2b_1 + ... + 2^{\sigma_2 - 1} b_{\sigma_2 - 1}$, затем берутся следующие σ_2 бит как σ_2-разрядное целое число a_1 и так далее до a_{q-1}.
2) Если $a_0, a_1, ..., a_{q-1}$ не различны, возвращается ошибка,
3) Сортируются пары (a_i, i) в лексикографическом порядке, чтобы получить пары $\left(a_{\pi(i),} \pi(i)\right)$, где π – перестановка $\{0,1,...,q-1\}$.
4) Определить $\alpha_{i+1} = \sum_{j=0}^{m-1} \pi(i)_j \cdot z^{m-1-j}$, где $\pi(i)$ определяет j наименее значащий бит в $\pi(i)$. Следует отметить, что конечное поле F_q построено как $F_2[z] / f(z)$.
5) Вернуть $(\alpha_1, \alpha_2, ..., \alpha_q)$.

3.3. Генерация ключей

Входные параметры:

не принимает никаких входных данных (помимо параметров).

Выходные параметры:

открытый ключ и личный ключ.

Алгоритм:

1) Генерируется строка δ с нормальным распределением длиной l бит, т.н. порождающее значение.
2) Вычисляется $E = F(\delta)$, строка длиной $(n + \sigma_2 q + \sigma_1 t + l)$ бит.
3) Определяется строка δ' как последние l бит от E.
4) Определяется s как первые n бит от E.
5) Вычисляется $\alpha_1, ..., \alpha_q$ из $\sigma_2 q$ последующих бит от E используя описанный выше алгоритм генерации упорядоченных полей. Если генерация завершается ошибкой, устанавливается $\delta \leftarrow \delta'$ и алгоритм выполняется заново.
6) Вычисляется g из следующих $\sigma_1 t$ от E используя описанный выше алгоритм генерации неприводимого полинома. Если генерация завершается ошибкой устанавливается $\delta \leftarrow \delta'$ и алгоритм выполняется заново.
7) Определяется $\Gamma = (g, \alpha_1, \alpha_2, ..., \alpha_n)$ (следует отметить, что $\alpha_{n+1}, ..., \alpha_q$ не используется здесь).
8) Вычисляется $(T, c_{n-k-\mu+1}, ..., c_{n-k}, \Gamma') \leftarrow MATGEN(\Gamma)$. Если генерация завершается ошибкой устанавливается $\delta \leftarrow \delta'$ и алгоритм выполняется заново.
9) Определяется Γ' как $(g, \alpha'_1, \alpha'_2, ..., \alpha'_n)$.
10) Формируется T как открытый ключ и как $(\delta, c, g, \alpha, s)$ личный ключ, где $c = (c_{n-k-\mu+1}, ..., c_{n-k})$ и $\alpha = (\alpha'_1, ..., \alpha'_n, \alpha_{n+1}, ..., \alpha_q)$.

3.4. Генерация вектора фиксированного веса

Входные параметры:

не принимает никаких входных данных.

Выходные параметры:

вектор $e \in F_2^n$ веса t. Алгоритм использует предварительно вычисленное целое число $\tau \geq t$.

Алгоритм:

1) Сгенерировать $\sigma_1 \tau$ случайных бит $b_0, b_1, ..., b_{\sigma_1 \tau - 1}$.

2) Определить $d_j = \sum_{i=0}^{m-1} b_{\sigma_1 j + i} 2^i$ для каждого $j \in \{0,1,...,\tau - 1\}$.

3) Определить $a_{0,} a_1, ..., a_{t-1}$ как первые t записей в $d_{0,} d_1, ..., d_{\tau - 1}$ в диапазоне $\{0,1,...,n-1\}$. Если получается определить менее чем t таких последовательностей, алгоритм выполняется заново.

4) Если $a_{0,} a_1, ..., a_{t-1}$ не все различны, алгоритм выполняется заново.

5) Определить $e = (e_{0,} e_1, ..., e_{n-1}) \in F_2^n$ как вектор веса t, такой, что $e_{a_i} = 1$ для каждого i.

6) Результатом является e.

Целое число τ определяется как t, если $n = q$; как $2t$, если $q/2 \leq n \leq q$; как $4t$, если $q/4 \leq n \leq q/2$; и т.д. Все выбранные параметры имеют $q/2 \leq n \leq q$, поэтому $\tau \in \{t, 2t\}$.

3.5. Инкапсуляция

Входные параметры:

открытый ключ T.

Выходные параметры:

шифротекст C и сеансовый ключ K.

Алгоритм:

1) Используя алгоритм генерации вектора фиксированного веса сгенерировать вектор $e \in F_2^n$ веса t.

2) Вычислить $C_0 = ENCODE(e, T)$.

3) Вычислить $C_1 = H(2, e)$.

4) Сформировать $C = (C_0, C_1)$.

5) Вычислить $K = H(1, e, C)$.

6) Сформировать шифротекст C и сеансовый ключ K.

3.6. Алгоритм кодирования

Входные параметры:

вектор-столбец $e \in F_2^n$ с весом t; и открытый ключ T, то есть матрица $(n-k) \times k$.

Выходные параметры:

вектор $C_0 \in F_2^{n-k}$.

Алгоритм:

1) Определить $H = (I_{n-k} \mid T)$.

2) Вычислить $C_0 = He \in F_2^{n-k}$.

3.7. Алгоритм декодирования

Входные параметры:

вектор $C_0 \in F_2^{n-k}$, набор параметров Γ'.

Выходные параметры:

вектор-столбец e, $C_0 = He$.

Существует два возможных исхода алгоритма:

– Если $C_0 = ENCODE(e,T)$, то $DECODE(C_0,\Gamma') = e$. Другими словами, если существует вектор $e \in F_2^n$ весом t, такой что выполняется $C_0 = He \in F_2^{n-k}$ с $H = (I_{n-k} \mid T)$, то алгоритм декодирования завершится успешно.

– Если C_0 не имеет формы He для любого вектора $e \in F_2^n$ весом t, то алгоритм декодирования завершится ошибкой.

Алгоритм:

1) Расширить C_0 до $\nu = (C_0, 0, ..., 0) \in F_2^n$ добавлением k нулей.
2) Найти уникальное кодовое слово c в коде Гоппы определенное Γ' с расстоянием $\le t$ от ν, если такого кодового слова нет, вернуть ошибку.
3) Установить $e = \nu + c$.
4) Если $wt(e) = t$ и $C_0 = He$ вернуть e, в обратном случае вернуть ошибку.

4. Параметры Classic McEliece

Существует пять предлагаемых наборов параметров в систематической форме: mceliece348864, mceliece460896, mceliece6960119, mceliece6688128 и mceliece8192128. Также авторами было предложено пять наборов параметров, использующих полусистематическую форму: mceliece348864f, mceliece460896f, mceliece6960119f, mceliece6688128f и mceliece8192128f.

В табл. 1 приведены общесистемные параметры вариаций криптосистемы Classic McEliece. В табл. 2 приведены фактические размеры входных и выходных параметров для вариаций алгоритма. Все данные приведены в байтах.

Таблица 1

Основные параметры криптосистемы Classic McEliece

Алгоритм	m	n	t	Полиномы
mceliece348864	12	3488	64	$f(z) = z^{12} + z^3 + 1,\ F(y) = y^{64} + y^3 + y + z$
mceliece348864f	12	3488	64	$f(z) = z^{12} + z^3 + 1,\ F(y) = y^{64} + y^3 + y + z,\ (\mu,\nu) = (32,64)$
mceliece460896	13	4608	96	$f(z) = z^{13} + z^4 + z^3 + 1,\ F(y) = y^{96} + y^{10} + y^9 + y^6 + 1$
mceliece460896f	13	4608	96	$f(z) = z^{13} + z^4 + z^3 + 1,\ F(y) = y^{96} + y^{10} + y^9 + y^6 + 1,\ (\mu,\nu) = (32,64)$
mceliece6688128	13	6688	128	$f(z) = z^{13} + z^4 + z^3 + z + 1,\ F(y) = y^{128} + y^7 + y^2 + y + 1$
mceliece6688128f	13	6688	128	$f(z) = z^{13} + z^4 + z^3 + z + 1,\ F(y) = y^{128} + y^7 + y^2 + y + 1,\ (\mu,\nu) = (32,64)$
mceliece6960119	13	6960	119	$f(z) = z^{13} + z^4 + z^3 + z + 1,\ F(y) = y^{119} + y^8 + 1$
mceliece6960119f	13	6960	119	$f(z) = z^{13} + z^4 + z^3 + z + 1,\ F(y) = y^{119} + y^8 + 1,\ (\mu,\nu) = (32,64)$
mceliece8192128	13	8192	128	$f(z) = z^{13} + z^4 + z^3 + 1,\ F(y) = y^{128} + y^7 + y^2 + y + 1$
mceliece8192128f	13	8192	128	$f(z) = z^{13} + z^4 + z^3 + 1,\ F(y) = y^{128} + y^7 + y^2 + y + 1,\ (\mu,\nu) = (32,64)$

Каждый из этих десяти наборов параметров имеет эталонную реализацию; реализацию с использованием 64-битной векторизации; реализацию с использованием 128-битных векторных инструкций Intel / AMD); и реализацию с использованием 256-битных векторных инструкций Intel / AMD). Эти четыре реализации являются интероперабельными и формируют идентичные тестовые векторы.

Таблица 2

Размеры входных и выходных параметров для криптографических функций криптосистемы Classic McEliece. Все размеры представлены в байтах

Алгоритм	Категория	Открытый ключ	Личный ключ	Шифртекст	Сессионный ключ
mceliece348864	1	261120	6492	128	32
mceliece348864f	1	261120	6492	128	32
mceliece460896	3	524160	13608	188	32
mceliece460896f	3	524160	13608	188	32
mceliece6688128	5	1044992	13932	240	32
mceliece6688128f	5	1044992	13932	240	32
mceliece6960119	5	1047319	13948	226	32
mceliece6960119f	5	1047319	13948	226	32
mceliece8192128	5	1357824	14120	240	32
mceliece8192128f	5	1357824	14120	240	32

Для наглядности на гистограмме (рис. 4) приведены размеры входных и выходных параметров. Так как значения параметров разнятся на несколько порядков, для наглядного представления на гистограммах данные длины личных и открытых ключей, а также длины соответствующих шифротекстов представлены в логарифмическом масштабе. Суть использования такого масштабирования заключается в преобразовании длин данных следующим образом: $x = \log_{10} X$, где X – параметр, такой как длина открытого или личного ключа, длина шифротекста, который подлежит масштабированию; x – результат вычисления десятичного логарифма над масштабируемым значением.

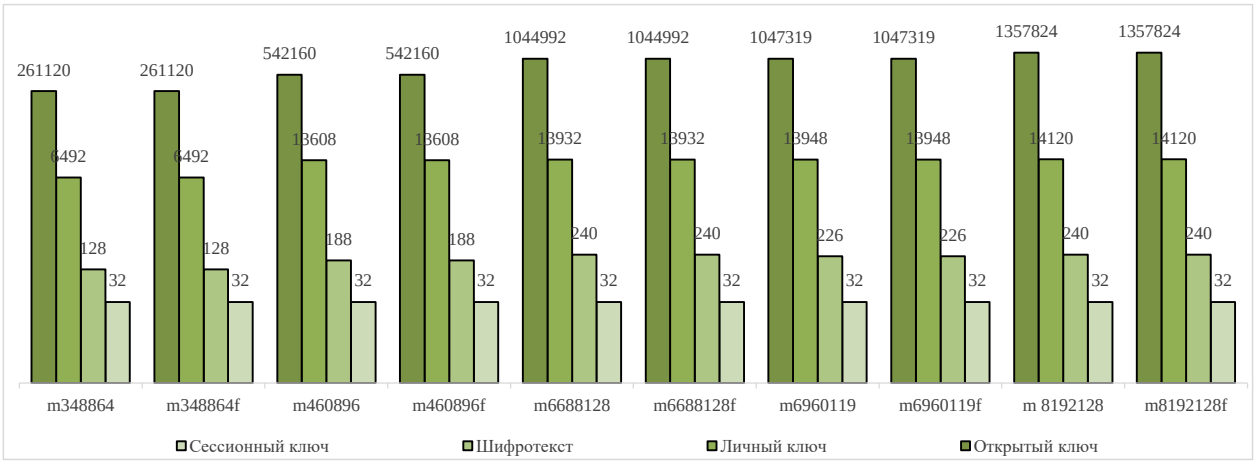

Рис. 4. Размеры входных и выходных параметров для криптографических функций криптосистемы Classic McEliece. Все размеры представлены в байтах

5. Анализ показателей быстродействия

Влияние выбора основных параметров на производительность. Размер зашифрованного текста составляет $n-k$ бит. Обычно соотношение $R = k/n$ выбирается около 0.8, поэтому размер зашифрованного текста составляет около $0.2n$ бит, то есть $n/40$ байтов, плюс 32 байта для подтверждения.

Размер открытого ключа составляет $k(n-k)$ бит. Для $R \approx 0.8$ это, примерно, $0.16n^2$ бит, то есть $n^2/50$ байтов. Для генерации открытого ключа используется $n^{3+o(1)}$ операций со стандартным методом Гаусса. Существуют асимптотически более быстрые

матричные алгоритмы. В операциях с личным ключом используется всего $n^{1+o(1)}$ операций со стандартными алгоритмами.

Сравнительный анализ показателей быстродействия. Данная оценка показателей быстродействия представлена в формате количества циклов процессора, затраченных на выполнение операции формирования ключей, инкапсуляции и деинкапсуляции ключей. Все измерения происходили без применения каких-либо технологий оптимизации производительности.

В частности, приводится оценка быстродействия, которая была проведена самими авторами при подаче алгоритма на всех трех раундах конкурса постквантовой криптографии. Программные измерения были выполнены с использованием пакета SUPERCOP на платформе Intel Xeon E3-1275 v3 (Haswell) с тактовой частотой 3,50 ГГц [1]. Объем оперативной памяти платформы 32 ГБ и работает под управлением Ubuntu 16.04. Для вариаций с полусистематической формой скорость выполнения операций инкапсуляции и деинкапсуляции не указана.

Приведенные оценки быстродействия сравнимы при условии игнорирования остальных (помимо приведенных показателей используемого центрального процессора) характеристик используемых вычислительных систем. Такая оценка носит исключительно первичный ознакомительный анализ возможностей производительности перечисленных выше алгоритмов. Как указывают сами авторы показатели производительности представленных ими оптимизированных реализаций алгоритма возможно еще улучшить.

В табл. 3 приведены показатели производительности для вариаций алгоритма инкапсуляции ключей Classic McElice, представленном на втором раунде конкурса. В табл. 4 приведены показатели производительности для Classic McElice, представленном на третьем раунде конкурса [1]. Данные приведены в циклах процессора, которые требуется выполнить для проведения каждой операции.

Как упоминалось ранее, каждая попытка генерации ключа (для систематических вариантов) успешна с вероятностью около 29 %, поэтому общее время генерации ключа варьируется. Однако окончательная успешная генерация ключей требует постоянного времени, и она использует отдельные случайные числа из неудачных попыток генерации ключей.

Для наглядности ниже на рис. 5 – 8 приведены гистограммы параметров быстродействия для всех вариаций алгоритма.

Таблица 3

Показатели производительности для вариаций алгоритма Classic McElice (раунд 2)

Название	Формирование ключей	Инкапсуляция	Деинкапсуляция
mceliece348864	208145574	46895	137503
mceliece348864f	82258215	-	-
mceliece460896	612458270	85410	274759
mceliece460896f	283980350	-	-
mceliece6688128	1344459257	156750	321536
mceliece6688128f	625501207	-	-
mceliece6960119	1202081992	156826	303207
mceliece6960119f	565430070	-	-
mceliece8192128	1277898472	185146	324803
mceliece8192128f	678901745	-	-

Таблица 4

Показатели производительности для вариаций алгоритма Classic McElice (раунд 3)

Название	Формирование ключевых данных	Инкапсуляция	Деинкапсуляция
mceliece348864	58034411	44350	134745
mceliece348864f	36641040	-	-
mceliece460896	215785433	117782	271694
mceliece460896f	117067765	-	-
mceliece6688128	556495649	151721	323957
mceliece6688128f	284584602	-	-
mceliece6960119	438217685	161224	301480
mceliece6960119f	246508730	-	-
mceliece8192128	514489441	178093	326531
mceliece8192128f	316202817	-	-

Чтобы адекватно продемонстрировать эти показатели, все данные на гистограммах приведены с использованием логарифмического масштаба, так как параметры разнятся на несколько порядков. Следует отметить, что меньшие значения циклов, затрачиваемых на выполнение одной операции, являются предпочтительными. В то же время, большие значения количества затрачиваемых циклов указывают на низкую скорость выполнения операции.

На рис. 5 приведена сводная гистограмма всех показателей быстродействия, показывающая общее соотношение скорости выполнения трех операций (формирование ключей, инкапсуляция и деинкапсуляция ключей) для каждого из вариаций алгоритмов, представленных на третьем раунде. Все данные указаны в затраченных на выполнение операции циклах.

Примерно сравнимую скорость выполнения операций инкапсулиции и деинкапсуляции имеют все вариации. Однако стоит отметить достаточно большой разрыв в производительности между формированием ключей и инкапсуляцией (деинкапсуляцией).

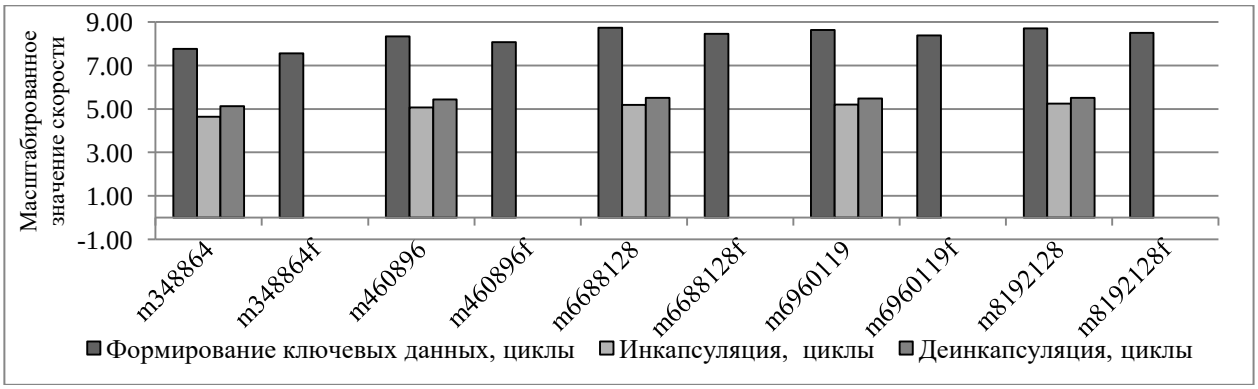

Рис. 5. Гистограмма показателей быстродействия (в логарифмическом масштабе)

Как указывалось, авторы для каждого из трех раундов представляли эталонные и оптимизированные версии алгоритма. Однако если прирост производительности в сравнении с первым раундом, по мнению авторов, был почти двойной. Прирост производительности же для реализации третьего раунда по сравнению со вторым менее значителен. Небольшого ускорения удалось добиться для операции формирования ключей, для остальных же операций скорость сравнима, а для вариации mceliece460896 из третьего раунда скорость инкапсуляции уменьшилась почти на треть. На рис. 6 и 7 приведена сводная гистограмма показателей быстродействия, показывающая общее соотношение скорости выполнения реализаций, представленных на втором и третьем раунде конкурса.

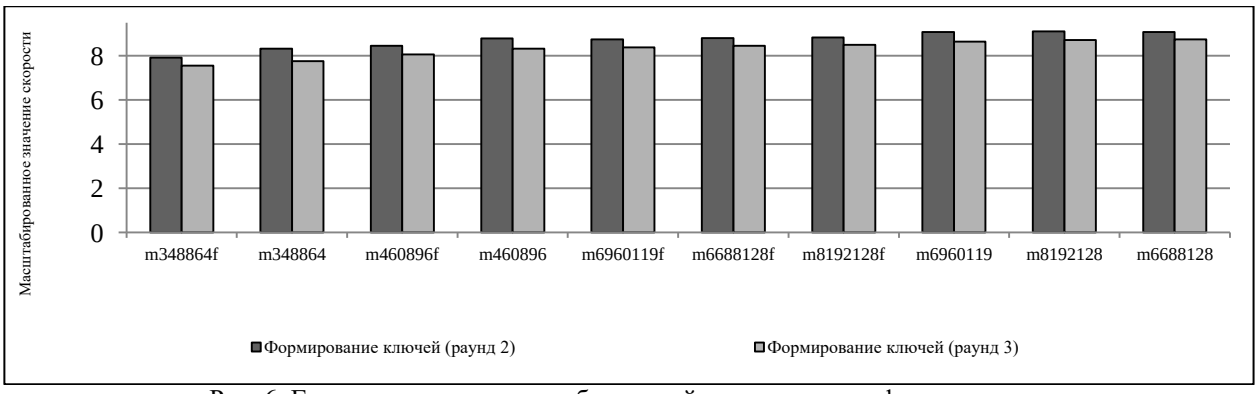

Рис. 6. Гистограмма показателя быстродействия: скорость формирования ключевых данных (раунд 2 и 3), в циклах (в логарифмическом масштабе)

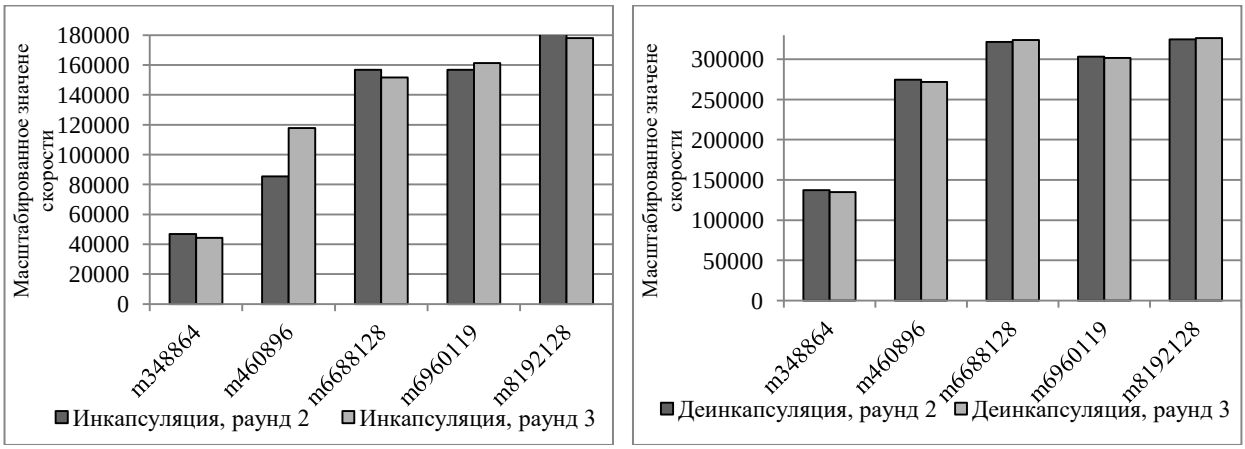

Рис. 7. Гистограмма показателей быстродействия: скорость инкапсуляции (раунд 2 и 3) и скорость деинкапсуляции (раунд 2 и 3), в циклах (в логарифмическом масштабе)

В рамках данного исследования проведен сравнительный анализ с алгоритмами BIKE и HQC. Для всех трех алгоритмов программные измерения были выполнены с использованием пакета SUPERCOP на платформе Intel Xeon E-2124 (CoffeeLake) с тактовой частотой 3,3 ГГц. На рис. 8 приведена гистограмма показателей быстродействия алгоритмов Classic McElice (раунд 3), BIKE и HQC.

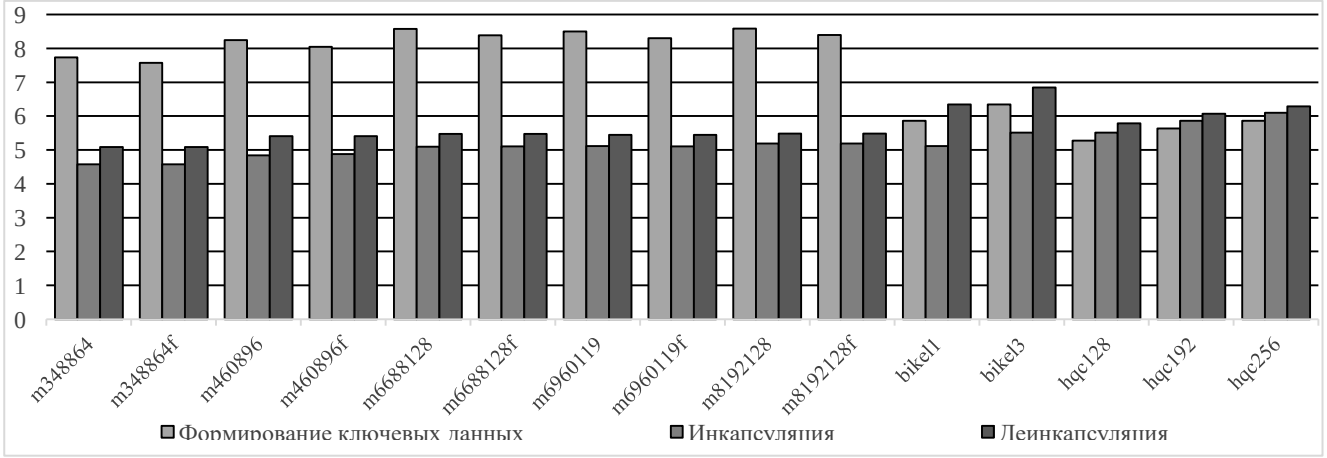

Рис. 8. Гистограмма показателей быстродействия алгоритмов Classic McElice, BIKE и HQC, в циклах (в логарифмическом масштабе)

Схема инкапсуляции ключей BIKE (BIt Flipping Key Encapsulation) представлена группой ученых из университетов стран Франции, США, Израиля, Германии [13]. Схема основана на квазициклических кодах с проверкой на четность (QC-MDPC) с умеренной плотно-

стью. Алгоритм обладает IND-CPA криптостойкостью. В третьем раунде авторами подана модификация алгоритма BIKE-2, представленного на предыдущих раундах. В основе алгоритма BIKE-2 лежит криптосистема Нидеррайтера с проверочной матрицей на четность. Алгоритм обеспечивает все три уровня безопасности.

Схема HQC (Hamming Quasi-Cyclic) подана на конкурс группой ученых из университетов стран Франции, США, Израиля, Германии [14]. Алгоритм обладает IND-CCA2 криптостойкостью. Основными особенностями HQC являются маленький размер открытого ключа и эффективные реализации на основе классических алгоритмов декодирования.

Стоит отметить, что, несмотря на медленную операцию формирования ключей для Classic McElice по сравнению с BIKE и HQC, скорость инкапсуляции и деинкапсулиции у рассматриваемого алгоритма меньше.

В требованиях указано, что алгоритм должен быть интероперабельным и масштабируемым. Поэтому в данном исследовании был проведен анализ производительности для довольно слабой вычислительной платформы. Измерения были выполнены с использованием пакета SUPERCOP на четырехядерной платформе Cortex-A9 с тактовой частотой 1,2 ГГц. Ожидаемо, скорость выполнения всех операций упала, для формирования ключей потребовалась на четверть больше операция процессора, остальным операциям требовалось на 1/5. Авторы отмечают, что совершенствование производительности для малоресурсных платформ является перспективным направлением исследований. На рис. 9 приведена гистограмма показателей быстродействия полученных при измерениях.

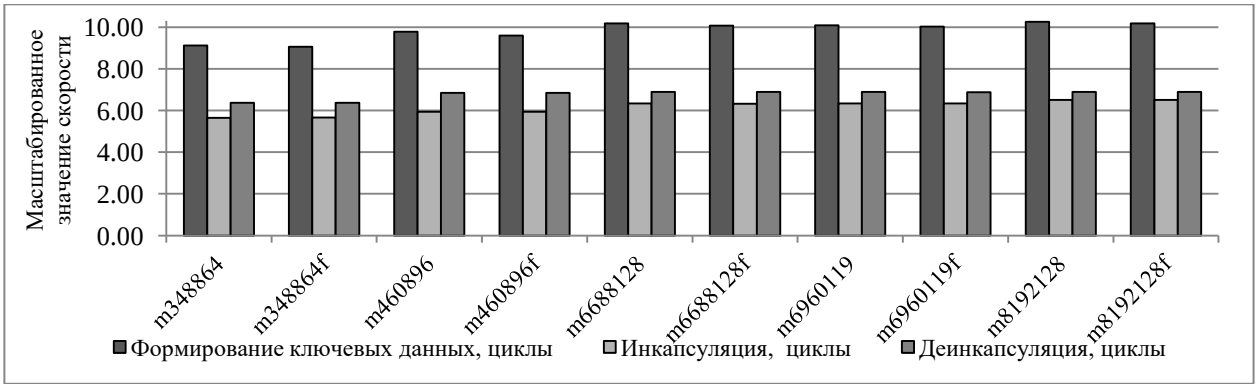

Рис. 9. Гистограмма показателей быстродействия алгоритма Classic McElice для вычислительной платформы Cortex-A9, в циклах (в логарифмическом масштабе)

6. Внесенные изменения

В алгоритм по мере продвижения конкурса постквантовой криптографии Classic McEliece вносились изменения. В частности, перед началом второго раунда конкурса NIST (30 марта 2019 г.) был существенно список наборов параметров m для поддержки необходимых уровней безопасности. Список параметров расширили следующим образом [1]:

- $(8192, 6528, 128)$, шифротексты длиной 240 байт: принимают как n, так и t как степени двойки; как и в первом раунде.
- $(6960, 5413, 119)$, шифротексты длиной 226 байт: оптимальная безопасность в пределах 2^{20} байт для открытого ключа; как и в первом раунде.
- $(6688, 5024, 128)$, шифротексты длиной 240 байт: оптимальная безопасность в пределах 2^{20} байт, если требуется, чтобы n и t были кратны 32.
- $(4608, 3360, 96)$, шифротексты длиной 188 байт: оптимальная безопасность в пределах 2^{19} байт, если требуется, чтобы n и t были кратны 32.

– $(3488,\ 2720,\ 64)$, шифротексты длиной 128 байт: оптимальная безопасность в пределах 2^{18} байтов, если требуется, чтобы n и t были кратны 32.

Поскольку Classic McEliece имеет такие маленькие зашифрованные тексты, можно было бы значительно сэкономить от изменения преобразования CCA, чтобы исключить подтверждение открытого текста (32 байта). Однако подтверждение в виде открытого текста имеет преимущества в плане безопасности: например, оно останавливает атаки с выбранным зашифрованным текстом на более раннем этапе, чем неявное отклонение, что упрощает защиту побочных каналов. Поэтому авторы решили сохранить существующее преобразование CCA.

Также возможно изменить параметры для дальнейшего уменьшения размеров зашифрованного текста на каждом уровне безопасности за счет использования более длинных ключей.

Был изменен формат случайного формирования пары ключей так, чтобы при необходимости можно было дополнительно сжимать ключи до 256 битной строки. Также был описан в качестве возможного будущего предложения альтернативный метод генерации ключей, который является более сложным (и требует больших сжатых личных ключей), но производительность которого значительно выше.

При подаче документации (от 10 октября 2020 г.) к третьему раунду наборы параметров для полусистематической формы алгоритма были задокументированы и реализованы в качестве возможных вариаций алгоритма.

Как и во 2-м раунде генерация ключа определяется с использованием различных случайных объектов, таких как случайный неприводимый многочлен g над F_q степени t. В 3-м раунде указывается явное отображение из случайной строки байтов в пары ключей с помощью явных отображений из случайных байтовых строк к случайным объектам, таким как g. Это оставляет открытой возможность указания дополнительных способов отображений в будущем с проверкой безопасности.

Отображение пар ключей начинается с 32 байтов случайности, что позволяет сжать личный ключ до 32-байтового порождающего значения. Порождающее значение расширяется с помощью $SHAKE256$ [12]. Выборка отклонения при генерации ключей обрабатывается путем детерминированного сопоставления каждого порождающего значения с новым порождающего значения. Личный ключ теперь имеет следующий формат: 256 бит (окончательного) порождающего значения, генерирующего ключ (а не ранее отклоненных порождающих значений); 64-битная строка веса 32, определяющая столбцы, используемые для полусистематической формы (или совместимая константа для систематической формы); многочлен g; управляющие биты для $(\alpha_1,...,\alpha_n)$; и n-битная строка s, используемая для неявного отклонения. Это оставляет открытой возможность указания других форматов личных ключей в будущем, таких как сжатые форматы (порядок объектов в этом формате разработан, чтобы обеспечить сжатие посредством простого усечения с эффективной декомпрессией) или список $(\alpha_1,...,\alpha_n)$ вместо управляющих битов для сред, где перестановка через RAM не является проблемой. Любой другой формат личного ключа с эффективными алгоритмами преобразования в этот формат личного ключа и обратно будет иметь такую же математическую безопасность.

Выводы

Проведен анализ финалиста конкурса постквантовой криптографии NIST PQC алгоритма инкапсуляции ключей Classic McEliece. Рассмотрены математические модели алгоритма Мак-Элиса и его современной вариации Classic McEliece. Рассмотрены характеристики входных и выходных параметров, а также основные показатели криптографической стойкости и быстродействия. Оценка быстродействия формировалась на основе эталонных реали-

заций на нескольких вычислительных платформах. Также проведен анализ изменений, внесенных на протяжении трех раундов конкурса. На основе этого проведен сравнительный анализ показателей быстродействия для вариаций алгоритма, представленных на трех раундах.

В ходе исследований установлено, что схема удовлетворяет формальным требованиям к кандидатам на постквантовые схемы инкапсуляции ключей, т.е. имеет различные варианты алгоритма, которые обеспечивают все три уровня крипто стойкости (1-й, 3-й и 5-й). Алгоритму свойственны длинные открытые ключи вследствие того, что вероятность успешного исполнения отдельных функций формирования ключа достаточно низка, формирование ключей – наиболее медленная и ресурсоемкая операция. Несмотря на это, операции инкапсуляции и деинкапсуляции по сравнению с остальными конкурсантами более производительны.

Перспективным направлением исследования является улучшение свойств масштабируемости без потери показателей производительности и криптографической стойкости.

Список литературы:

1. Classic McEliece: conservative code-based cryptography [Электронный ресурс]. Режим доступа: https://classic.mceliece.org/nist/mceliece-20201010.pdf
2. McEliece R.J. A public-key cryptosystem based on algebraic coding theory // Prog. Rep., Jet Prop. Lab., California Inst. Technol, 1978. P. 114 – 116.
3. Горбенко І.Д. Прикладна криптологія. Теорія. Практика. Застосування: монографія / І.Д. Горбенко, Ю.І. Горбенко. Харків : Форт, 2012. 870 с.
4. Есин В.І. Безпека інформаційних систем і технологій / В.І. Есин, О.О. Кузнецов, Л.С. Сорока. Харків : ХНУ ім. В.Н. Каразіна, 2013. 632 с.
5. Горбенко І. Д. Постквантова криптографія та механізми її реалізації / І. Д. Горбенко, О. О. Кузнєцов, О. В. Потій, Ю. І. Горбенко, Р. С. Ганзя, В. А. Пономар // Радіотехніка. 2016. Вип. 186. С. 32-52.
6. Гоппа В. Д.. Введение в алгебраическую теорию информации. Москва : Наука, Физматлит, 1995. 112 с.
7. Post-Quantum Cryptography [Электронный ресурс]. Режим доступа: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions
8. Report on Post-Quantum Cryptography [Электронный ресурс]. https://csrc.nist.gov/publications/detail/nistir/8105/final
9. Daniel J. Bernstein Johannes Buchmann Erik Dahmen. Post-Quantum Cryptography. [Электронный ресурс]. Режим доступа: https://www.researchgate.net/profile/Nicolas_Sendrier/publication/226115302_Code -Based_ Cryptography/links/540d62d50cf2df04e7549388/Code-Based-Cryptography.pdf.
10. Menezes J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. Boca Raton, Florida. CRC Press. 1997. 816 p.
11. Katz, Jonathan; Lindell, Yehuda. Introduction to Modern Cryptography: Principles and Protocols // Chapman and Hall/CRC, 2007. 552 pages.
12. FIPS PUB 180-4, Secure Hash Standard (SHS) [Электронный ресурс]. Режим доступа: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
13. BIKE: Bit Flipping Key Encapsulation [Электронный ресурс]. Режим доступа: https://bikesuite.org/files/v4.1/BIKE_Spec.2020.10.22.1.pdf
14. Hamming Quasi-Cyclic (HQC) [Электронный ресурс]. Режим доступа: http://pqc-hqc.org/doc/hqc-specification_2020-10-01.pdf
15. SUPERCOP [Электронный ресурс]. Режим доступа: https://bench.cr.yp.to/supercop.html

Поступила в редколлегию 00.00.2020

Сведения об авторах:

Луценко Мария Сергеевна – аспирант кафедры безопасности информационных систем и технологий, Харьковский национальный университет имени В. Н. Каразина, Украина; e-mail: lutsenko.maria.kh@gmail.com, ORCID: https://orcid.org/0000-0003-2075-5796