УДК 004.421.5
DOI 10.33764/2618-981X-2023-6-129-136

В. Е. Кудряшов[1], А. Н. Фионов[1,2]*

Метод генерации случайных компонент в системе NTRU

[1] Сибирский государственный университет геосистем и технологий, г. Новосибирск, Российская Федерация
[2] Сибирский государственный университет телекоммуникаций и информатики, г. Новосибирск, Российская Федерация
* e-mail: vadkud@inbox.ru

Аннотация. В современном мире безопасность информации зависит от защищенности криптосистем. И в последние годы замечен стремительный рост количества исследовательских работ в области квантовой криптографии. Ведется активное создание первых прототипов квантовых компьютеров и процессоров (существуют идеи реализации на оптических системах, где в роли кубита выступает фотон). Такие устройства значительно превосходят классические компьютеры по скорости обработки операций. Например, задача факторизации больших целых чисел решима за реальное время с использования алгоритма Шора на квантовом компьютере. Это делает небезопасными многие современные системы криптографической защиты информации (RSA, DH, ECDSA, ЭЦП). Поэтому необходимо уже сейчас задумываться о криптографических системах защиты информации, которые будут сохранять свою стойкость даже к атакам с квантовых компьютеров. Одна из таких систем-претендентов – NTRU. В данной статье она рассмотрена более детально, а также предложен метод генерации случайных компонент для ее реализации.

Ключевые слова: шифрование, RSA, квантовый компьютер, NTPU, постквавовая криптография, информационная безопасность

V. E. Kudryashov[1], A. N. Fionov[1, 2]*

The Method of Generation of Random Components in NTRU System

[1] Siberian State University of Geosystems and Technologies, Novosibirsk, Russian Federation
[2] Siberian State University of Telecommunications and Informatics, Novosibirsk, Russian Federation
* e-mail: vadkud@inbox.ru

Abstract. In the modern world, the security of information depends on the security of cryptosystems. And in recent years, there has been a rapid increase in the number of research papers in the field of quantum cryptography. An active creation of the first prototypes of quantum computers and processors is underway (there are ideas for implementation on optical systems, where a photon plays the role of qubit). Such devices are significantly superior to classical computers in terms of processing speed. For example, the problem of factorization of large integers can be solved in real time using Shor's algorithm on a quantum computer. This makes many modern systems of cryptographic information protection (RSA, DH, ECDSA, EDS) insecure. Therefore, it is necessary now to think about cryptographic information protection systems that will remain resistant even to attacks from quantum computers. One such candidate system is NTRU. In this article, it is considered in more detail, and a method for generating random components for its implementation is also proposed.

Keywords: encryption, RSA, quantum computer, NTPUEncrypt, post-quantum cryptography, information secuirity

Введение

Криптосистема NTRU относится к методам криптографии, основанной на решетках. Криптография на решетках – подход к построению алгоритмов асимметричного шифрования с использованием задач теории решеток, то есть задач оптимизации на дискретных аддитивных подгруппах, заданных на множестве [4]. Вместе с другими методами постквантовой криптографии она считается перспективной из-за способности квантового компьютера расшифровывать широко используемые системы асимметричной криптографии, основанные на двух типах задач теории чисел: задачах целочисленной факторизации и задачах дискретного логарифмирования. Сложность алгоритмов взлома, построенных на решетках, чрезвычайно высока, лучшие алгоритмы могут решить эту задачу с трудом за экспоненциальное время.

Постквантовая криптография на решетках основана на труднорешаемых как для квантовых, так и для классических компьютеров задачах на решетках, таких как:

- нахождение кратчайшего вектора;
- нахождение идеального кратчайшего вектора;
- нахождение кратчайшего независимого вектора;
- поиск короткого целого решения.

NTRU был изобретен в 1996 году и представлен миру на конференции CRYPTO. Причиной, послужившей началом разработки алгоритма в 1994 году, стала статья, в которой говорилось о легкости взлома существующих алгоритмов на квантовых компьютерах, которые, как показало время, не за горами. Система полностью отвечает стандартам IEEE P1363 в соответствии со спецификациями решетчатой криптографии с открытым ключом [5].

В отличие от RSA или El Gamal, NTRU работает не над кольцом вычетов по модулю целого числа N, а над кольцом многочленов (полиномов), приводимых по модулю. Например, множество полиномов для работы NTRU можно задать так:

$$(Z / 7Z)[x] / (x^4 - 1), \tag{1}$$

где Z – множество целых чисел.

Из формулы (1) можно понять, что это множество многочленов, приводимых по модулю ($x^4 - 1$); у которых коэффициенты – целые числа, приводимые по модулю 7 [6].

Для реализации алгоритма задаются 6 парамтеров: N, p, q, d, d_f, d_g. Задаются также 3 призвольных набора многочленов, отвечающих принципу (1): L_f, L_g, L_r.

Рассмотрим метод генерации открытого и секретного ключей:

1. Из набора L_f выбирается произвольный многочлен f степени $(N - 1)$, который имеет d_f «1» и $(d_f - 1)$ «-1» (остальные «0») в качестве коэффиентов. Выбираться продолжается до тех пор, пока не будет получен многочлен, который имеет инверсии:

$$f_p = f^{(-1)} \ mod(p, x^n - 1), \tag{2}$$

$$f_q = f^{(-1)} \ mod(p, x^q - 1). \tag{3}$$

2. Из набора L_g выбирается произвольный многочлен g степени $(N - 1)$, который имеет d_g «1» и d_g «-1» (остальные «0») в качестве коэффиентов.

3. Открытый ключ вычисляется по формуле:

$$h = p * g \otimes f_q \ mod(q, x^n - 1). \tag{4}$$

4. Секретный ключ – это пара полиномов $(f; f_p)$.

Рассмотрим процесс шифрования сообщения:

1. Представляем отправляемое сообщение в виде полинома m степени $(N - 1)$.

2. Из набора L_r выбираем произвольный многочлен r степени $(N - 1)$, который имеет d «1» и d «-1» (остальные «0») в качестве коэффиентов. Полином называется «ослепляющим».

3. Зашифрованное сообщение вычисляется по формуле:

$$c = r \otimes h + m \ mod(q, x^n - 1). \tag{5}$$

Рассмотрим процесс расшифрования сообщения:

1. Вычисляем полином a по формуле:

$$a = c \otimes f \ mod(q, x^n - 1). \tag{6}$$

2. Переводим многочлен a в канонический вид по модулю q.

3. Расшифрованное сообщение вычисляется по формуле:

$$m = a \otimes f_p \ mod(p, x^n - 1). \tag{7}$$

Работу алгоритма расшифрования можно проверить, если подставить в формулу (6) формулу (5). Тогда мы получим a, равное:

$$a = m \otimes f \ mod(p, x^n - 1). \tag{8}$$

В результате преобразования формулы (7) с учетом формулы (8) получим:

$$m = m \otimes f \otimes f_p \ mod(p, x^n - 1). \tag{9}$$

Применив к формуле (9) формулу (2), получим $m = m$.

Рассмотрев весь алгоритм работы NTRU, можно выделить проблему генерации случайных полиномов с ограничениями по количеству опередлнных ко-

эффициентов. Это шаги 1-2 в алгоритмах генерации ключей и шаг 2 в алгоритме шифрования. Можно сформулировать следующую цель: разработать метод генерации «случайных» компонент для реализации алгоритма NTRU.

Методы и материалы

Перед тем как перейти к предложенному методу генерации случайных полиномов, вспомним, как работает арифметичекое кодирование.

Например, у нас есть алфавит A = {a; b; c}. Процесс кодирования/декодирования можно представить в виде схемы (рис. 1), где P – распределение вероятностей, а Q – распределение кумулятивных вероятностей.

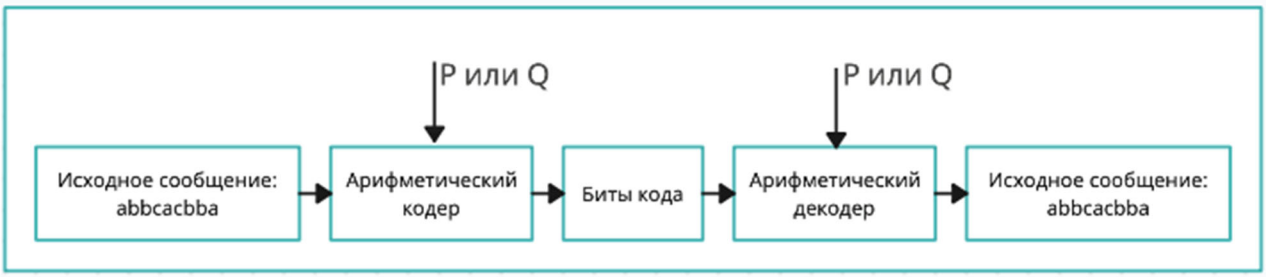

Рис. 1. Процес кодирования/декодирования

Предположим, у нас стоит задача закодировать сообщение (0 1 0 -1 1). Заменим буквы в алфавите A на нужные нам коэффициенты {-1; 0; 1}. Распределение вероястностей можно увидеть в табл. 1 [7]. То есть если в схему из рис. 1 передать алфавит A с распределением вероятностей из табл. 1, то мы сможем закодировать и однозначно декодировать исходное сообщение из нашей задачи.

В схеме (рис. 1) заменим арифметический кодер/декодер на омофонный. Омофонное кодирование представляет собой вид рандомизации сообщений, в котором буква источника заменяется специальными символами (омофонами), выбираемыми случайным образом так, чтобы сделать кодовую последовательность неотличимой от последовательности равновероятных и независимых нулей и единиц [8].

При посимвольном омофонном кодировании для каждого символа сообщения выбирается соответствующий ему интервал и производится омофонное кодирование этого интервала. Основная идея арифметического кодирования заключается в том, что кодирование интервала на каждом шаге не производится [9]. Вместо этого на интервале, соответствующем первому символу сообщения, рассматривается новое распределение символов алфавита источника, в котором выбирается интервал, соответствующий второму символу сообщения и т.д. Иными словами, каждый последующий символ сообщения сужает текущий интервал до интервала, соответствующего этому символу. В результате получается интервал, соответствующий всему сообщению. Для рандомизации сообщения необходимо произвести омофонное кодирование этого заключительного интервала [10]. Для обозначения описанного метода далее в тексте будет использо-

ваться аббревиатура АКРИ (арифметическое кодирование с разделением интервала).

Таблица 1

Распределение вероятностей

	0	1	0	-1	1	
p_{-1}	1	1	1	1	0	0
p_0	2	1	1	0	0	0
p_1	2	2	1	1	1	0
N	5	4	3	2	1	0
q_{-1}	0	0	0	0	0	0
q_0	1	1	1	1	0	0
q_1	3	2	2	1	0	0

Дело в том, что омофонный кодер гарантирует на выходе получение полностью случайной кодовой последовательности. Благодаря этому свойству, можно получить случайные коэффициенты полинома из случайной последовательности бит при задании кумулятивного распределения вероятностей (количества «-1», «0» и «1») для декодера АКРИ [11]. А случайную последовательность бит можно получить из генератора случайных бит (рис. 2).

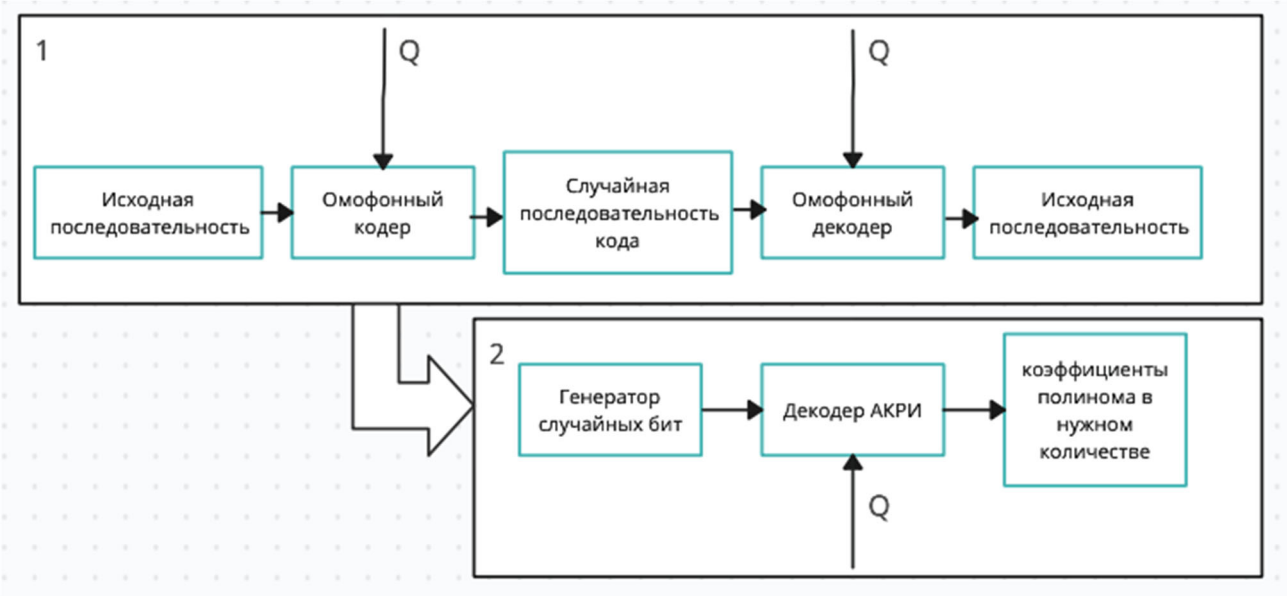

Рис. 2. Алгоритм генерации случайных коэффициентов заданного количества

Результаты

Попробуем сгенерировать необходимые нам наборы коэффициентов для многочленов f, g и r по рекомендуемым входным параметрам NTRU для максимального уровня стойкости ($N = 503$, $d_f = 155$, $d_g = 100$, $d = 65$) [7].

На рис. 3 показан результат работы программы для первого полинома *f*, который имеет всего 503 элемента, из них 155 «1», 154 «-1» и остальные «0». Т.е. мы в результате получаем готовый случайный полином с нужным количеством «0», «1» и «-1»: $-x^{501}+x^{500}-x^{499}-x^{497}$ …-1. Также программа считает, сколько случайных бит понадобилось извлечь из входной последовательности. Для полинома *f* это 798 случайных бит. Скорость работы программы меньше 1 секунды.

```
MacBook-Air-Vadim:~ kite$ ./genntru 503 155 154
6549ce9721ae61a473375ad19850296c881a57f1b4719656ade609f8b680882ad3ad4bdaa1860139e79a
b26075182dea936f7c1068db9d28e707909231472bdb83311ad30f43d645a1e1925af94b7e4f8caee0e8
6fa279659bdac88c9849b8c2cf4d16899bf78a858c01b0b5c6987e8cb8ac84270c86f588146cd7036ac7
c662
0 -1 1 -1 0 1 0 -1 1 0 -1 0 0 -1 1 1 1 -1 0 0 -1 0 -1 0 1 -1 1 1 -1 0 1 0 -1 -1 0 0
1 0 0 1 0 -1 -1 1 1 0 0 1 0 1 0 0 -1 1 -1 -1 -1 0 0 1 0 1 0 -1 0 1 0 1 0 -1 -1 1 1 -
1 -1 1 1 1 0 0 0 1 1 1 -1 -1 0 1 0 -1 -1 0 0 0 1 -1 0 1 1 1 0 1 1 0 0 0 -1 1 -1 0 0
-1 0 0 0 1 -1 0 1 0 0 0 0 0 1 -1 0 0 1 -1 0 -1 0 1 0 1 0 0 0 0 1 0 1 1 -1 1 -1 0 0 1
 1 -1 0 -1 -1 -1 0 1 1 0 -1 -1 -1 0 1 1 0 0 1 0 0 0 0 -1 -1 -1 0 -1 -1 -1 -1 1 0 -1
0 1 0 0 0 0 -1 1 1 0 1 1 -1 -1 1 1 0 1 1 0 1 0 -1 1 1 1 1 0 -1 0 0 1 0 -1 0 1 1 0 0
0 0 0 -1 -1 -1 0 1 -1 1 -1 -1 0 -1 0 1 1 1 -1 0 -1 1 1 1 0 0 1 1 1 0 0 0 -1 1 -1 0 -
1 0 1 1 -1 -1 -1 0 1 -1 -1 -1 -1 0 -1 0 -1 -1 0 1 -1 0 0 1 0 0 1 0 0 0 0 0 1 0 -1 0
1 1 0 0 -1 1 1 0 -1 0 1 -1 1 1 0 -1 1 0 0 -1 1 0 -1 1 1 -1 -1 0 1 0 0 1 -1 0 0 0 -1
1 -1 1 1 1 -1 -1 1 1 -1 0 0 -1 -1 -1 0 -1 0 0 0 -1 0 1 -1 1 0 -1 -1 0 0 -1 0 0 1 1 1
 -1 1 1 -1 0 -1 0 -1 1 1 0 -1 1 0 -1 1 -1 -1 0 0 1 1 1 1 1 0 0 1 0 1 1 -1 -1 0 0 0 -
1 0 1 -1 0 -1 -1 0 1 0 1 1 1 1 1 -1 1 0 -1 -1 0 -1 0 -1 -1 -1 1 0 -1 1 0 -1 -1 1 0 1
 -1 -1 -1 -1 0 0 -1 -1 -1 1 0 -1 0 1 1 0 -1 -1 0 0 0 0 -1 1 -1 -1 0 0 0 0 0 -1 1 0 -
1 -1 0 -1 0 1 0 -1 0 -1 -1 0 -1 -1 1 1 1 1 1 0 1 1 -1 1 0 0 -1 0 -1
bits = 798
```

Рис. 3. Случайные коэффициенты для полинома *f* в нужном количестве

На рис. 4, 5 показаны результаты работы программы для полиномов *g* и *r* соответсвенно.

```
MacBook-Air-Vadim:~ kite$ ./genntru 503 100 100
cf4d16899bf78a858c01b0b5c6987e8cb8ac84270c86f588146cd7036ac7c6620f32d152e57a9420ec26
a12cb8f9f000bd5d26c77401b8c097b0e938d46d0a5a28a80b156726f11ee6ac1aef5c563693aa4056a9
e8de861c44ec66b4642d5e8c
1 -1 0 -1 -1 -1 1 0 0 0 0 0 0 -1 0 -1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 -1 0 0 0 0 1 1
 0 0 0 -1 0 -1 0 1 0 1 1 0 0 -1 -1 0 0 0 0 0 1 0 1 0 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0
 0 -1 0 0 0 1 0 -1 1 0 0 0 0 0 0 0 -1 0 1 0 0 1 0 0 -1 -1 1 0 0 0 1 -1 1 0 1 0 0 1 0
 0 0 0 1 0 1 1 0 -1 0 -1 1 0 0 1 0 -1 0 -1 0 0 -1 0 0 -1 0 0 -1 0 0 0 0 0 1 -1 0 0 0 0
 -1 0 0 0 0 0 -1 -1 0 1 -1 0 -1 0 0 0 0 0 1 0 -1 0 0 0 0 0 -1 0 1 -1 0 1 0 -1 0 0 1 1
-1 0 0 -1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 -1 0 1 0 1 0 0 0 -1 -1 0 0 1 -1 0 0 -1 1 1
 -1 1 0 0 1 0 0 1 -1 0 0 0 -1 1 0 0 -1 -1 -1 0 0 -1 0 0 0 1 1 -1 0 0 1 0 -1 1 0 1 0
0 0 -1 -1 0 0 1 0 0 0 1 0 1 0 -1 1 0 1 0 -1 0 -1 1 0 -1 0 1 0 0 -1 0 0 0 0 0 0 0 1 0
 1 -1 0 0 0 1 0 -1 1 0 0 1 0 0 1 -1 0 0 1 0 0 0 0 -1 0 0 0 0 0 0 -1 1 0 0 0 0 0 0 0 0
0 1 0 -1 0 1 -1 0 0 0 0 0 1 0 0 1 -1 0 -1 0 0 -1 0 0 0 0 0 0 1 0 0 -1 0 1 1 1 0 1 1
0 -1 0 -1 1 0 0 1 0 1 0 0 1 0 0 0 -1 -1 0 -1 1 1 -1 0 0 1 -1 0 0 0 0 0 0 -1 -1 0 0 0
 -1 0 0 1 -1 0 1 -1 0 1 1 0 1 -1 0 0 0 -1 0 1 0 1 0 -1 0 0 0 -1 -1 0 0 1 0 0 0 0 0 -
1 -1 1 0 1 -1 -1 -1 -1 -1 0 0 0 0 0 1 0 0 0 -1 0 0 1 0 0 0 0 -1 0 -1 0 -1 0 -1 0 1 1
 -1 0 1 1 0 1
bits = 693
```

Рис. 4. Случайные коэффициенты для полинома *g* в нужном количестве

```
MacBook-Air-Vadim:~ kite$ ./genntru 503 65 65
9e7c1d2333be6b3925da298edf424a511fbb8b571aace82a55daa2ee3ed7c6623552538d4387991fdd55
4b0eff4842998df6dc64ea69ccca06439b53c26d078dcdb8b3c06827ee17063c618305f4b477da939d68
e5cc53cf451593c64b99c0d2
0 0 0 0 1 0 -1 0 0 0 0 0 0 0 -1 0 0 0 -1 -1 0 0 0 0 0 0 0 -1 1 0 1 0 0 0 0 0 0 0 0
 0 0 0 0 -1 -1 0 1 0 -1 1 -1 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 -1 0 1 1 0 0 0 0 -1 0
0 0 0 0 0 -1 0 0 0 0 -1 0 0 -1 0 0 1 0 0 0 1 0 0 0 1 -1 1 -1 0 0 0 0 1 0 0 0 0 -1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 -1 0 1 0 0 -1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 1 0 -1 1 0 1
 0 -1 1 0 0 0 0 0 0 -1 0 0 -1 0 0 0 -1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 -1 0 -1 0 0 0 0 0 -1 1 0 0 0 0 1 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 1 -1 0 0 0 0 0 0 0 0 -1 1 0 0 0 1 0 0 -1 0 0 -1 0 0 0 0 -1 0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 -1 0 0 0 0 0 0 1 -1 0 0 1 0 0 0 0 0 0 0 0 -1 0 0 0 0 1 0 1 -1 1 0 0 0 0
0 0 0 -1 -1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 1 0 0 -1 0 -1 0
 0 0 0 0 0 0 -1 0 0 0 -1 1 0 0 0 0 -1 1 -1 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 0 0 0 -1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 -1 1 1 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 -1 0 -1 0 0 0 0 0 0 -1 0 1 1 0 0 0 0 0 -1 -1 0 0 0
 0 0 1 0 -1 0 1 1 0 -1 0 -1 -1 0 0 0 0 -1 -1 0 0 0 0 -1 1 -1 0
bits = 551
```

Рис. 5. Случайные коэффициенты для полинома *r* в нужном количестве

Заключение

Появление квантовых компьютеров ставит под угрозу безопасность всей информации, защищемой методами классической криптографии (RSA, ЭЦП, Эль-Гамаль, DH) [12]. Необходимо уже сейчас задумываться о системах защиты (например NTRU), которые будут стойки к атакам с подобных устройств. Одна из сложностей реализации системы NTRU залючается в том, что несколько раз нужно из набора многочленов выбрать «произвольный» многочлен с заданным во входных параметрах количеством определенных коэффициентов. В данной статье предложен метод генерации случайных компонент для системы NTRU, который позволяет это решить. Метод основан на свойстве арифметического кодирования с разделением интервала (АКРИ) получить в результате полностью случайную закодированную последовательность кода.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Буковшин, В.А., Чуб, П.А., Черкесова, Л.В., Короченцев, Д.А., Поркшеян, В.М. Анализ современных постквантовых алгоритмов шифрования / В.А. Буковшин – Научное обозрение №4: Технические науки, 2005. – 128 с.

2. Дрон, К.К. О перспективах совместного использования методов квантовой и классической криптографии / К.К Дрон. – Вестник Хакасского государственного университета им НФ Катанова. – 2018. – № 24. – С. 8-11.

3. Кузьмин, Т. В. Криптографические методы защиты информации / Т.В. Кузьмин. – Москва: Огни, 2013. – 192 с.

4. Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman. NTRU: A Ring Based Public Key Cryptosystem. In Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, J.P. Buhler (ed.), Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin, 1998, 267–288.

5. Bernstein D.J., Chuengsatiansup Ch., Lange T., van Vredendaal Ch. NTRU Prime: reducing attack surface at low cost. / D.J. Bernstein – URL: https://eprint.iacr.org/2016/461.pdf (дата обращения: 15.10.2020).

6. Авдошин, С. Дискретная математика. Модулярная алгебра, криптография, кодирование [Текст] / С. Авдошин – Москва: СИНТЕГ, 2016. – 260 с.

7. Fionov A. Universal homophonic coding // 2001 IEEE International Symposium on Information Theory (ISIT 2001). – Washington, DC, USA, June 24-29, 2001. – P. 116.

8. Jendal H. N., Kuhn Y. J. B., Massey J. L. An information-theoretic treatment of homophonic substitution // Advances in Cryptology Eurocrypt- 89. – Berlin: Springer-Verlag, 1990. – P. 382–394 (Lecture Notes in Computer Science; V. 434).

9. Rissanen J. J., Langdon G. G. Arithmetic coding // IBM J. Res. Dev. – 1979. – V. 23, No2. – P. 149–162.

10. Фионов А. Н. Эффективный метод рандомизации сообщений на основе арифметического кодирования. – 1997. – Т. 4, No 2. – С. 51–74.

11. Fionov A. Random number generation via homophonic coding [Text] // 2000 IEEE International Symposium on Information Theory (ISIT 2000). – Sorrento, Italy, June 25–30, 2000. – P. 354.

12. Хоффман, Л. Дж. Современные методы защиты информации / Л. Дж. Хоффман – СПб: Питер**,** 2014. – 264 с.