МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА 51

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

ассистент		М. Н. Исаева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

NTRU

по дисциплине: КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 5912		М. С. Фомин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 Описание NTRU

2 Определения

Решётки – это дискретные подгруппы вещественного векторного пространства $\mathbb{R}^n$, линейные оболочки которых совпадают с ним: $L = \left\{ \sum_{i=1}^{n} \lambda_i b_i \mid \lambda_i \in \mathbb{Z} \right\}$.

Линейная оболочка $V(X)$ подмножества X линейного пространства V — пересечение всех подпространств V, содержащих X. Линейная оболочка является подпространством V.

Факторкольцо — общеалгебраическая конструкция, позволяющая распространить на случай колец конструкцию факторгруппы.

Факторгруппа — множество смежных классов группы по её нормальной подгруппе, само являющееся группой с определённой специальным образом групповой операцией.

KEM – key encapsulation mechanism – механизм инкапсуляции ключей.

DPKE – deterministic public key encryption scheme – детерминированная схема шифрования с открытым ключом.

$(\mathbb{Z}/n)^{\times}$ это мультипликативная группа целых чисел по модулю n.

Φ_1 – многочлен вида $(x-1)$.

Φ_n – это многочлен $(x^n-1)(x-1) = x^{n-1} + x^{n-2} + \ldots + 1$.

R – это факторкольцо $\mathbb{Z}[x]/(\Phi_1\Phi_n)$.

S – факторкольцо вида $\mathbb{Z}[x]/(\Phi_n)$.

$R/3$ – это факторкольцо $\mathbb{Z}[x]/(3, \Phi_1\Phi_n)$.

R/q – это факторкольцо $\mathbb{Z}[x]/(q, \Phi_1\Phi_n)$. Каноничный R/q-представитель из $a \in \mathbb{Z}[x]$ – это уникальный многочлен $b \in \mathbb{Z}[x]$ степени не больше $n-1$ с коэффициентами из множества $\{-q/2, -q/2+1, \ldots, q/2-1\}$ такой, что $a \equiv b \left(\mathrm{mod}(q, \Phi_1\Phi_n)\right)$.

$S/2$ – это факторкольцо $\mathbb{Z}[x]/(2, \Phi_n)$. Каноничный $S/2$-представитель из

$a \in \mathbb{Z}[x]$ – это уникальный многочлен $b \in \mathbb{Z}[x]$ степени не больше $n-2$ с коэффициентами из множества $\{0,1\}$ такой, что $a \equiv b\left(\mathrm{mod}\left(2, \Phi_n\right)\right)$.

$S/3$ – это факторкольцо $\mathbb{Z}[x]/(3, \Phi_n)$. Каноничный $S/3$-представитель из $a \in \mathbb{Z}[x]$ – это уникальный многочлен $b \in \mathbb{Z}[x]$ степени не больше $n-2$ с коэффициентами из множества $\{-1,0,1\}$ такой, что $a \equiv b\left(\mathrm{mod}\left(3, \Phi_n\right)\right)$.

S/q – это факторкольцо $\mathbb{Z}[x]/(q, \Phi_n)$. Каноничный S/q-представитель из $a \in \mathbb{Z}[x]$ – это уникальный многочлен $b \in \mathbb{Z}[x]$ степени не более $n-2$ с коэффициентами из множества $\{-q/2, -q/2+1, ..., q/2-1\}$ такой, что $a \equiv b\left(\mathrm{mod}\left(q, \Phi_n\right)\right)$.

Многочлен является тернарным (троичным), если его коэффициенты принадлежат множеству $\{-1,0,1\}$.

Тернарный многочлен $\mathrm{v} = \sum_i v_i x^i$ обладает свойством неотрицательной корреляции, если $\sum_i v_i v_{i+1} \geq 0$.

T – это множество ненулевых тернарных многочленов степени не больше $n-2$. Другими словами, T - это множество каноничных $S/3$-представителей.

T_+ – это подмножество множества T, состоящее из многочленов со свойством неотрицательной корреляции.

$T(d)$ для четного положительного числа d является подмножеством множества T, состоящим из многочленов, которые имеют ровно $d/2$ коэффициентов, равных $+1$, и $d/2$ коэффициентов, равных -1.

3 Параметризация

Для NTRU определен следующий набор параметров: $(n, p, q, L_f, L_g, L_r, L_m, \text{Lift})$, где n, p, q – взаимно простые положительные целые числа; L_f, L_g, L_r, L_m – множества целочисленных многочленов; Lift – инъекция $L_m \to \mathbb{Z}[x]$, для которой $\underline{S3}(\text{Lift}(m)) = m$ для всех $m \in L_m$. Набор параметров корректен, если $(p \cdot r \cdot g + \text{f} \cdot \text{Lift}(m)) \bmod (\Phi_1 \Phi_n)$ имеет коэффициенты в множестве $\{-q/2, -q/2+1, ..., q/2-1\}$ для всех $(\text{f, g, r, m}) \in (L_f, L_g, L_r, L_m)$.

Числом n характеризуется размер выбираемых в качестве ключей многочленов. Числа p и q не обязательно должны быть простыми, но $\text{НОД}(p, q)$ должен равняться 1. Следует отметить, что параметр p служит для определения интервала, которому обязаны принадлежать все коэффициенты многочленов, используемых в криптосистеме. А если точнее, пространство сообщений L криптосистемы NTRU определяется как:

$$L_m = \left\{ M(x) \in \mathbb{R} \mid \text{для всех коэффициентов } M \text{ лежащих в } \left[-\frac{p-1}{2}; \frac{p-1}{2} \right] \right\}.$$

4 DPKE и KEM

DPKE (детерминированная схема шифрования с открытым ключом) параметризуется взаимно простыми положительными целыми числами n, p, q, выборочными пространствами L_f, L_g, L_r, L_m и инъекцией Lift: $L_m \to \mathbb{Z}[x]$. Разработчики рекомендуют два узко определенных семейства наборов параметров, которые называют ntru-hps и ntru-hrss.

Наборы параметров ntru-hps следуют примеру Хоффстейна, Пифера и Сильвермана, использующему выборочные пространства с фиксированным весом, и допускают несколько вариантов выбора q для каждого n [2].

Наборы параметров ntru-hrss следуют примеру Хюльсинга, Рейнвельда, Шанка и Швайба, использующим произвольные весовые выборочные пространства и фиксированный q как функцию n [1].

KeyGen′(*seed*)	Encrypt($\mathbf{h}$, ($\mathbf{r}$, $\mathbf{m}$))	Decrypt(($\mathbf{f}$, $\mathbf{f}_p$, $\mathbf{h}_q$), $\mathbf{c}$)
1. $(\mathbf{f}, \mathbf{g}) \leftarrow$ Sample_fg(*seed*)	1. $\mathbf{m}' \leftarrow$ Lift($\mathbf{m}$)	1. if $\mathbf{c} \not\equiv 0 \pmod{(q, \mathbf{\Phi}_1)}$ return $(0, 0, 1)$
2. $\mathbf{f}_q \leftarrow (1/\mathbf{f}) \bmod (q, \mathbf{\Phi}_n)$	2. $\mathbf{c} \leftarrow (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}') \bmod (q, \mathbf{\Phi}_1\mathbf{\Phi}_n)$	2. $\mathbf{a} \leftarrow (\mathbf{c} \cdot \mathbf{f}) \bmod (q, \mathbf{\Phi}_1\mathbf{\Phi}_n)$
3. $\mathbf{h} \leftarrow (3 \cdot \mathbf{g} \cdot \mathbf{f}_q) \bmod (q, \mathbf{\Phi}_1\mathbf{\Phi}_n)$	3. return $\mathbf{c}$	3. $\mathbf{m} \leftarrow (\mathbf{a} \cdot \mathbf{f}_p) \bmod (3, \mathbf{\Phi}_n)$
4. $\mathbf{h}_q \leftarrow (1/\mathbf{h}) \bmod (q, \mathbf{\Phi}_n)$		4. $\mathbf{m}' \leftarrow$ Lift($\mathbf{m}$)
5. $\mathbf{f}_p \leftarrow (1/\mathbf{f}) \bmod (3, \mathbf{\Phi}_n)$		5. $\mathbf{r} \leftarrow ((\mathbf{c} - \mathbf{m}') \cdot \mathbf{h}_q) \bmod (q, \mathbf{\Phi}_n)$
7. return $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{h})$		6. if $(\mathbf{r}, \mathbf{m}) \in \mathcal{L}_r \times \mathcal{L}_m$ return $(\mathbf{r}, \mathbf{m}, 0)$
		7. else return $(0, 0, 1)$

Рисунок 1 – DPKE из реализации NTRU

KEM (механизм инкапсуляции ключей), который создали разработчики, имеет строгое доказательство безопасности IND-CCA2 в модели случайного оракула (ROM) в предположении, что их же DPKE является безопасным OW-CPA. Он также имеет строгое доказательство безопасности IND-CCA2 в квантовой доступной модели случайного оракула (QROM) при нестандартном предположении, сформулированном Сайто, Ксагавой и Ямакавой [4]. KEM отклоняет недействительные зашифрованные тексты, возвращая псевдослучайный ключ вместо символа ошибки — метод, называемый неявным отклонением.

KeyGen $(seed)$	Encapsulate $(\mathbf{h})$	Decapsulate $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q, s), \mathbf{c})$
1. $(\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{h}) \leftarrow \mathsf{KeyGen}'(seed)$	1. $coins \leftarrow_{\$} \{0,1\}^{256}$	1. $(\mathbf{r}, \mathbf{m}, fail) \leftarrow \mathsf{Decrypt}((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{c})$
2. $s \leftarrow_{\$} \{0,1\}^{256}$	2. $(\mathbf{r}, \mathbf{m}) \leftarrow \mathsf{Sample\_rm}(coins)$	2. $k_1 \leftarrow H_1(\mathbf{r}, \mathbf{m})$
7. return $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q, s), \mathbf{h})$	3. $\mathbf{c} \leftarrow \mathsf{Encrypt}(\mathbf{h}, (\mathbf{r}, \mathbf{m}))$	3. $k_2 \leftarrow H_2(s, \mathbf{c})$
	4. $k \leftarrow H_1(\mathbf{r}, \mathbf{m})$	4. if $fail = 0$ return k_1
	5. return $(\mathbf{c}, k)$	5. else return k_2

Рисунок 2 – KEM из реализации NTRU

Набором параметров для nru-hps являются: $\mathrm{Lift}(m) = m$ и $L_f = T,\ L_g = T(q/8-2),\ L_r = T,\ L_m = T(q/8-2)$. Выбор L_g гарантирует, что $h \equiv 0 \pmod{q, \Phi_1}$, и наряду с выбором L_m это гарантирует, что $c \equiv 0 \pmod{q, \Phi_1}$. Весовой параметр $q/8-2$ является самым большим, который совместим с идеальной корректностью.

Набором параметров для nru-hrss являются: $\mathrm{Lift}(m) = \Phi_1 \cdot \underline{S3}(m / \Phi_1)$ и $L_f = T_+,\ L_g = \{\Phi_1 \cdot v : v \in T_+\},\ L_r = T,\ L_m = T$. Выбор L_g гарантирует, что $h \equiv 0 \pmod{q, \Phi_1}$, и наряду с выбором Lift это гарантирует, что $c \equiv 0 \pmod{q, \Phi_1}$.

6 Процесс шифрования и дешифрования

6.1 Построение ключей

Генерация пары ключей DPKE_Key_Pair:

На вход подаются из битовой строки битовые блоки (coins) длины sample_key_bits.

На выходе получаем байтовый массив – *packed_private_key* длины dpke_private_key_bytes, а также байтовый массив – *packed_public_key* длины dpke_public_key_bytes.

Процедура:

1. $(\mathrm{f},\mathrm{g}) = \mathrm{Sample\_fg}(coins)$;
2. $\mathrm{f}_p = S3\_\mathrm{inverse}(\mathrm{f})$;
3. $(\mathrm{h},\mathrm{h}_q) = \mathrm{DPKE\_Public\_Key}(\mathrm{f},\mathrm{g})$;
4. $packed\_private\_key = \mathrm{pack\_S3}(\mathrm{f}) \,||\, \mathrm{pack\_S3}(\mathrm{f}_p) \,||\, \mathrm{pack\_Sq}(\mathrm{h}_q)$;
5. $packed\_public\_key = \mathrm{pack\_Rq0}(\mathrm{h})$;
6. выход – ($packed\_private\_key, packed\_public\_key$).

Генерация открытого ключа DPKE_Public_Key:

На вход подаются многочлены $\mathrm{f} \in L_f$ и $\mathrm{g} \in L_g$.

На выходе получаем многочлен h, который удовлетворяет равенству $\underline{\mathrm{Rq}}(\mathrm{h} \cdot \mathrm{f}) = 3 \cdot \mathrm{g}$, и многочлен h_q, который удовлетворяет равенству $\underline{\mathrm{Sq}}(\mathrm{h} \cdot \mathrm{h}_q) = 1$.

Процедура:

1. $\mathrm{G} = 3 \cdot \mathrm{g}$;
2. $v_0 = \mathrm{Sq}(\mathrm{G} \cdot \mathrm{f})$;
3. $v_1 = \mathrm{Sq\_inverse}(v_0)$;
4. $\mathrm{h} = \mathrm{Rq}(v_1 \cdot \mathrm{G} \cdot \mathrm{G})$;
5. $\mathrm{h}_q = \mathrm{Rq}(v_1 \cdot \mathrm{f} \cdot \mathrm{f})$;
6. выход – $(\mathrm{h},\mathrm{h}_q)$.

6.2 Шифрование

Процедура шифрования DPKE_Encrypt:

На вход подается байтовый массив *packed_public_key* длины dpke_public_key_bytes и байтовый массив *packed_rm* длины dpke_plaintext_bytes.

На выходе имеем байтовый массив *packed_ciphertext* длины dpke_ciphertext_bytes.

Процедура:

1. Разделить *packed_rm* на $packed\_r \,||\, packed\_m$: *packed_r* длины packed_s3_bytes и *packed_m* длины packed_s3_bytes;
2. $\mathrm{r} = \underline{\mathrm{S3}}(\mathrm{unpack\_S3}(packed\_r))$;
3. $\mathrm{m}_0 = \mathrm{unpack\_S3}(packed\_m)$;
4. $\mathrm{m}_1 = \mathrm{Lift}(\mathrm{m}_0)$;
5. $\mathrm{h} = \mathrm{unpack\_Rq0}(packed\_public\_key)$;
6. $\mathrm{c} = \mathrm{Rq}(\mathrm{r} \cdot \mathrm{h} + \mathrm{m}_1)$;
7. $packed\_ciphertext = \mathrm{pack\_Rq0}(\mathrm{c})$;
8. выход – *packed_ciphertext.*

6.3 Дешифрование

Процедура дешифрования DPKE_Decrypt:

На вход подается байтовый массив *packed_private_key* длины dpke_private_key_bytes и *packed_ciphertext* длины dpke_ciphertext_bytes.

На выходе получаем байтовый массив *packed_rm* длины dpke_plaintext_bytes и бит *fail*.

Процедура:

1. Разделить *packed_private_key* на $packed\_f \,||\, packed\_fp \,||\, packed\_hq$: *packed_f* длины packed_s3_bytes, *packed_fp* длины packed_s3_bytes и *packed_hq* длины packed_s3_bytes;
2. $\mathrm{c} = \mathrm{unpack\_Rq0}(packed\_ciphertext)$;

3. $\mathrm{f} = \underline{\mathrm{S3}}(\mathrm{unpack\_S3}(packed\_f))$;

4. $\mathrm{f}_p = \mathrm{unpack\_S3}(packed\_fp)$;

5. $\mathrm{h}_q = \mathrm{unpack\_Rq0}(packed\_hq)$;

6. $v_1 = \underline{\mathrm{Rq}}(\mathrm{c} \cdot \mathrm{f})$;

7. $\mathrm{m}_0 = \underline{\mathrm{S3}}(v_1 \cdot \mathrm{f}_p)$;

8. $\mathrm{m}_1 = \mathrm{Lift}(\mathrm{m}_0)$;

9. $\mathrm{r} = \underline{\mathrm{Sq}}((\mathrm{c} - \mathrm{m}_1) \cdot \mathrm{h}_q)$;

10. $packed\_rm = \mathrm{pack\_S3}(\mathrm{r}) \,||\, \mathrm{pack\_S3}(\mathrm{m}_0)$;

11. если $\mathrm{r} \in L_m$ и $\mathrm{m}_0 \in L_m$, то $fail = 0$;

12. иначе $fail = 1$;

13. выход – ($packed\_rm, fail$).

6.4 Инкапсуляция ключей

Генерация пары ключей Key_Pair:

На вход подается битовая строка *seed* длины key_seed_bits.

На выходе имеем байтовый массив *packed_private_key* длины kem_private_key_bytes и *packed_public_key* длины kem_public_key_bytes.

Процедура:

1. Разделить *seed* на $fg\_bits \,||\, prf\_key$: fg_bits длины sample_key_bits и prf_key длины prf_key_bits;
2. $(packed\_dpke\_private\_key, packed\_public\_key) = \mathrm{DPKE\_Key\_Pair}(fg\_bits)$;
3. $packed\_private\_key = packed\_dpke\_private\_key \,||\, \mathrm{bits\_to\_bytes}(prf\_key)$;
4. $(packed\_private\_key, packed\_public\_key)$.

Процедура инкапсуляции Encapsulate:

На вход подается байтовый массив *packed_public_key* длины kem_public_key_bytes.

На выходе получаем битовую строку *shared_key* длины

kem_shared_key_bits и байтовый массив *packed_ciphertext* длины kem_ciphertext_bytes.

Процедура:

1. Пусть *coins* будут строкой однородных случайных битов из sample_plaintext_bits;
2. $(\mathrm{r,m}) = \mathrm{Sample\_rm}(coins)$;
3. $packed\_rm = \mathrm{pack\_S3(r)} \,||\, \mathrm{pack\_S3(m)}$;
4. $shared\_key = \mathrm{Hash}(\mathrm{bytes\_to\_bits}(packed\_rm, 8 \cdot \mathrm{dpke\_plaintext\_bytes}))$;
5. $packed\_ciphertext = \mathrm{DPKE\_Encrypt}(packed\_public\_key, packed\_rm)$.

Процедура декапсуляции Decapsulate:

На вход подается байтовый массив *packed_private_key* длины kem_private_key_bytes и байтовый массив *packed_ciphertext* длины kem_ciphertext_bytes.

На выходе получаем битовую строку *shared_key* длины kem_shared_key_bits.

Процедура:

1. Разделить *packed_private_key* на $packed\_f \,||\, packed\_fp \,||\, packed\_hq \,||\, prf\_key$: *packed_f* длины packed_s3_bytes, *packed_fp* длины packed_s3_bytes, *packed_hq* длины packed_s3_bytes и *prf_key* длины packed_sq_bytes;
2. $(packed\_rm, fail) = \mathrm{DPKE\_Decrypt}(packed\_private\_key, packed\_ciphertext)$;
3. $shared\_key = \mathrm{Hash}(\mathrm{bytes\_to\_bits}(packed\_rm, 8 \cdot \mathrm{dpke\_plaintext\_bytes}))$;
4. $random\_key = \mathrm{Hash}(\mathrm{bytes\_to\_bits}(prf\_key, \mathrm{prf\_key\_bits}) \,||\, \mathrm{bytes\_to\_bits}(packed\_ciphertext, 8 \cdot \mathrm{kem\_ciphertext\_bytes}))$;
5. если $fail = 0$, выход – *shared_key;*
6. иначе выход – *random_key*.

7 Рекомендуемые параметры

Изначально был выбран набор параметров ntruhrss701, поскольку $n = 701$ обеспечивает самый высокий уровень безопасности среди наборов параметров ntru-hrss с $q := 2^{\lceil 7/2+\log_2(n) \rceil} \leq 8192$. При выборе наборов параметров ntru-hps разработчики также пытались максимально повысить безопасность при минимизации q:

Разработчики решили рекомендовать только наборы параметров nru-hps со значениями $n/3 \leq q/8-2 \leq 2n/3$, где $q/8-2$ – это вес векторов в L_g и L_m. Параметры веса за пределами этого диапазона рассматриваются как потенциальная угроза безопасности.

Таким образом, реализованную систему необходимо использовать со следующими параметрами (таблица 1):

Таблица 1 – Рекомендуемые параметры

	ntruhps2048509	ntruhps2048677	ntruhps4096821	ntruhrss701
n	509	677	821	701
q	2048	2048	4096	8192
Hash	SHA3_256	SHA3_256	SHA3_256	SHA3_256
sample_fixed_type_bits	15240	20280	24630	-
sample_iid_bits	4064	5408	6560	5600
sample_key_bits	19304	25688	31190	11200
sample_plaintext_bits	19304	25688	31190	11200
packed_s3_bytes	102	136	164	140
packed_rq0_bytes	699	930	1230	1138
packed_sq_bytes	699	930	1230	1138
dpke_public_key_bytes	699	930	1230	1138
dpke_private_key_bytes	903	1202	1558	1418
dpke_plaintext_bytes	204	272	328	280
dpke_ciphertext_bytes	699	930	1230	1138
kem_public_key_bytes	699	930	1230	1138

kem_private_key_bytes	935	1234	1590	1450
kem_ciphertext_bytes	699	930	1230	1138
kem_shared_key_bits	256	256	256	256

8 Ожидаемый уровень безопасности

Категории безопасности NIST 1, 3 и 5 определяются относительно «вычислительных ресурсов», которые требуются для поиска ключа в блочном шифре с 128-, 192- или 256-битным ключом соответственно. В конкурсе предложений говорится, что «вычислительные ресурсы могут быть измерены с использованием множества различных показателей» и что пороговые значения должны быть удовлетворены «в отношении всех показателей, которые NIST считает потенциально важными для практической безопасности». NIST не указал исчерпывающий набор соответствующих метрик, поэтому разработчики решили предоставить две оценки безопасности: одну относительно нелокальных моделей вычислений и одну относительно локальных моделей вычислений.

Атаки на блочные шифры с поиском ключей хорошо работают в локальных моделях вычислений, и кажется маловероятным, что нелокальность может быть использована для значительного повышения производительности. То же самое нельзя сказать об атаках на NTRU. Некоторые из лучших алгоритмов атак, например, алгоритмы сита для задачи кратчайшего вектора, достигают значительно лучшей производительности в нелокальных моделях, чем в локальных. Тем не менее, атакам в локальных моделях не уделяется такого же внимания, и ситуация нестабильна.

На рисунке 3 представлены результаты оценок безопасности для нелокальных и локальных моделей вычислений:

Security categories relative to **non-local** models	
Parameter set	Category
ntruhps2048509	–
ntruhrss701	1
ntruhps2048677	1
ntruhps4096821	3

Security categories relative to **local** models	
Parameter set	Category
ntruhps2048509	1
ntruhrss701	3
ntruhps2048677	3
ntruhps4096821	5

Рисунок 3 – оценки безопасности

9 Преимущества и недостатки

По словам разработчиков, их система обладает следующими преимуществами:

1. Система работает корректно. KEM IND-CCA2 всегда устанавливает ключ; он никогда не прерывается из-за сбоя дешифрования. Это упрощает анализ схемы и делает ее привлекательной заменой KEM, которые используются сегодня.
2. Система хорошо изучена. Среди предположений, лежащих в основе постквантовых криптосистем, хорошо изучена безопасность OWCPA NTRU. NTRU и аналогичные системы часто использовались для оценки новых методов восстановления решетки. Эта история конкретного криптоанализа придает NTRU некоторую уверенность.
3. Гибкость системы. Базовый DPKE может быть параметризован для различных вариантов использования с различными требованиями к размеру, безопасности и эффективности.
4. Простота. DPKE имеет только два основных параметра n и q, и может быть полностью описан в терминах простой целочисленной полиномиальной арифметики. Преобразование в безопасный KEM IND-CCA2 концептуально просто.
5. Система достаточно быстрая, компактная и бесплатная.

Также у системы есть ряд ограничений:

1. Несмотря на описанные преимущества, NTRU вряд ли будет самой быстрой, вряд ли будет самой компактной и вряд ли будет самой безопасной системой из предложенных. Однако по продуктам этих мер NTRU будет конкурентоспособен.
2. Выбор оптимальных параметров для NTRU в настоящее время ограничен плохим пониманием неасимптотического поведения новых алгоритмов для SVP. Это ограничение характерно для всех криптосистем на основе решеток.

ЗАКЛЮЧЕНИЕ

Результатом курсовой работы является составленная пояснительная записка к алгоритму NTRU, который основан на решетчатой криптосистеме и создан в качестве альтернативы RSA и криптосистемам на эллиптических кривых (ECC). В ходе написания были подробно описаны все ключевые моменты алгоритма, такие как: построения ключей, шифрование, дешифрование, описание предлагаемых параметров, анализ стойкости. Также были выявлены и раскрыты его преимущества и недостатки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe. High-speed key encapsulation from NTRU. In Wieland Fischer and Naofumi Homma, editors, Cryptographic Hardware and Em-bedded Systems - CHES 2017, LNCS. Springer, 2017.
2. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A new high speed public key cryptosystem, 1996. draft from at CRYPTO '96 rump session.
3. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, Algorithmic Number Theory - ANTS-III, volume 1423 of LNCS, pages 267-288. Springer, 1998.
4. Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 520-551. Springer, 2018.