СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задачей данной курсовой работы является составление пояснительной записки к алгоритму CRYSTALS-KYBER. Алгоритм основан на схеме шифрования с открытым ключом, предназначенной для обеспечения безопасности как от классических, так и от квантовых атак. Сама идея шифрования основана на квантовой стойкости известной задачи MLWE.

Данный алгоритм является одним из самых перспективных претендентов на роль будущего постквантового стандарта (что подтверждает его проход в 3 этап конкурса NIST). Он обладает неплохой производительностью, его основой является хорошо изученная проблема.

1 Краткое описание алгоритма

Kyber – это механизм инкапсуляции ключей (KEM) с защитой IND-CCA2, основанный на стойкости проблемы обучения с ошибками (LWE) над модульными решетками (MLWE).

Алгоритм Kyber имеет два режима:

1) Kyber.CPAPKE – это режиму соответствует уровень защиты IND-CPA
2) Kyber.CCAKEM – это режиму соответствует уровень защиты IND-CCA2. Kyber.CCAKEM получают с помощью слегка изменённого преобразования Фудзисаки–Окамото (FO)

2 Теоретическая информация и обозначения

β – множество {0…255}, т.е. набор 8-битных целых чисел без знака (байт)

β^k - байтовый массив длины k

β^* - байтовый массив произвольной длины

R – кольцо многочленов $\mathbb{Z}[X]/(X^n+1)$

R_q - кольцо многочленов $\mathbb{Z}_q[X]/(X^n+1)$, где $n=2^{n'-1}$, здесь и далее $n=256$, $n'=9$, $q=3329$

$Compress_q(x,d)=\lceil(2^d/q)*x\rfloor \bmod^+ 2^d$ - функция сжатия

$Decompress_q(x,d)=\lceil(q/2^d)*x\rfloor$ - функция расширения

$PRF:\beta^{32}\times\beta\to\beta^*$ - псевдослучайная функция, используется SHAKE-256

$XOF:\beta^*\times\beta\times\beta\to\beta^*$ - расширение вывода, используется SHAKE-128

$H:\beta^*\to\beta^{32}$ - хеш-функция, используется SHA3-256

$G:\beta^*\to\beta^{32}\times\beta^{32}$ - хеш-функция, используется SHA3-512

$KDF:\beta^*\to\beta^*$ - функция генерации ключа, используется SHAKE-256

NTT - Теоретико-числовые преобразования, критическая по времени функция, требуемая многими постквантовыми криптографическими протоколами, основанными на решетках.

Kyber использует детерминированный подход к выборке элементов в R_q , которые статистически близки к равномерно случайному распределению. Для этого используется функция **Parse**:

Algorithm 1 Parse: $\mathcal{B}^* \to R_q^n$

Input: Byte stream $B = b_0, b_1, b_2 \cdots \in \mathcal{B}^*$
Output: NTT-representation $\hat{a} \in R_q$ of $a \in R_q$

$i := 0$
$j := 0$
while $j < n$ **do**
  $d_1 := b_i + 256 \cdot (b_{i+1} \bmod^+ 16)$
  $d_2 := \lfloor b_{i+1}/16 \rfloor + 16 \cdot b_{i+2}$
  if $d_1 < q$ **then**
    $\hat{a}_j := d_1$
    $j := j + 1$
  end if
  if $d_2 < q$ **and** $j < n$ **then**
    $\hat{a}_j := d_2$
    $j := j + 1$
  end if
  $i := i + 3$
end while
return $\hat{a}_0 + \hat{a}_1 X + \cdots + \hat{a}_{n-1} X^{n-1}$

Рис.1 – Функция Parse

Шум в алгоритме Kyber генерируется с помощью центрированного биноминального распределения и обеспечивается соответствующей функцией:

Algorithm 2 CBD_η: $\mathcal{B}^{64\eta} \to R_q$

Input: Byte array $B = (b_0, b_1, \ldots, b_{64\eta-1}) \in \mathcal{B}^{64\eta}$
Output: Polynomial $f \in R_q$

$(\beta_0, \ldots, \beta_{512\eta-1}) := \mathsf{BytesToBits}(B)$
for i from 0 to 255 **do**
  $a := \sum_{j=0}^{\eta-1} \beta_{2i\eta+j}$
  $b := \sum_{j=0}^{\eta-1} \beta_{2i\eta+\eta+j}$
  $f_i := a - b$
end for
return $f_0 + f_1 X + f_2 X^2 + \cdots + f_{255} X^{255}$

Рис.2 – Функция CBD

В алгоритме требуется представлять в виде байтовых массивов многочлены (и наоборот), поэтому были созданы две функции Encode и Decode:

Algorithm 3 $\mathsf{Decode}_\ell\colon \mathcal{B}^{32\ell} \to R_q$

Input: Byte array $B \in \mathcal{B}^{32\ell}$
Output: Polynomial $f \in R_q$
$(\beta_0, \ldots, \beta_{256\ell-1}) \coloneqq \mathsf{BytesToBits}(B)$
for i from 0 to 255 **do**
$f_i \coloneqq \sum_{j=0}^{\ell-1} \beta_{i\ell+j} 2^j$
end for
return $f_0 + f_1 X + f_2 X^2 + \cdots + f_{255} X^{255}$

Рис.3 – Функция Decode

3 Описание режимов

Kyber.CPAPKE

Kyber.CPAPKE похож на схему шифрования LPR которая была представлена для Ring-LWE Любашевским, Пейкертом и Регевым в презентации на Eurocrypt 2010. Данная схема основана на самой первой схеме шифрования на основе проблемы LWE. Главное отличие в том, что базовое кольцо не являлось кольцом $\mathbb{Z}_q$ и в том, что и секрет, и векторы ошибок имели малые коэффициенты.

Идея использования полиномиального кольца (вместо $\mathbb{Z}_q$) взята у криптосистемы NTRU, представленной Хоффштейном, Пайфером и Сильверманом, в то время как симметрия между секретом и ошибкой уже использовалась в очень похожих криптографических схемах.

Основное отличие от схемы шифрования LPR заключается в использовании Module-LWE вместо Ring-LWE. Кроме того, Kyber использует подход, принятый Алкимом, Дукасом, Пеппельманом и Швабе для генерации открытой матрицы **A**. Кроме того, алгоритм предусматривает сокращение зашифрованных текстов, с помощью округления младших битов, как при обучении, с помощью схем, основанных на округлении (learning-with-

rounding-based), что является распространенной практикой для уменьшения размера зашифрованного текста в других схемах на основе LWE.

Kyber.CPAPKE имеет свои параметры: $n = 256, k, q = 3329, \eta_1, \eta_2, d_u, d_v$

Параметры *n* и *q* должны быть именно такими.

Algorithm 4 KYBER.CPAPKE.KeyGen(): key generation

Output: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) := \mathrm{G}(d)$
$N := 0$
for i from 0 to $k-1$ **do** ▷ Generate matrix $\hat{\mathbf{A}} \in R_q^{k \times k}$ in NTT domain
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, j, i))$
  end for
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{s} \in R_q^k$ from B_{η_1}
  $\mathbf{s}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{e} \in R_q^k$ from B_{η_1}
  $\mathbf{e}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
$\hat{\mathbf{s}} := \mathsf{NTT}(\mathbf{s})$
$\hat{\mathbf{e}} := \mathsf{NTT}(\mathbf{e})$
$\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
$pk := (\mathsf{Encode}_{12}(\hat{\mathbf{t}} \bmod^+ q) \| \rho)$ ▷ $pk := \mathbf{As} + \mathbf{e}$
$sk := \mathsf{Encode}_{12}(\hat{\mathbf{s}} \bmod^+ q)$ ▷ $sk := \mathbf{s}$
return (pk, sk)

Рис.4 – Генерация ключа для Kyber.CPAPKE

Algorithm 5 KYBER.CPAPKE.Enc(pk, m, r): encryption

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Input: Message $m \in \mathcal{B}^{32}$
Input: Random coins $r \in \mathcal{B}^{32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$

$N := 0$
$\hat{\mathbf{t}} := \mathsf{Decode}_{12}(pk)$
$\rho := pk + 12 \cdot k \cdot n/8$
for i from 0 to $k-1$ **do** ▷ Generate matrix $\hat{\mathbf{A}} \in R_q^{k \times k}$ in NTT domain
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}^T[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, i, j))$
  end for
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{r} \in R_q^k$ from B_{η_1}
  $\mathbf{r}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(r, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{e}_1 \in R_q^k$ from B_{η_2}
  $\mathbf{e}_1[i] := \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(r, N))$
  $N := N + 1$
end for
$e_2 := \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(r, N))$ ▷ Sample $e_2 \in R_q$ from B_{η_2}
$\hat{\mathbf{r}} := \mathsf{NTT}(\mathbf{r})$
$\mathbf{u} := \mathsf{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$ ▷ $\mathbf{u} := \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
$v := \mathsf{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + e_2 + \mathsf{Decompress}_q(\mathsf{Decode}_1(m), 1)$ ▷ $v := \mathbf{t}^T\mathbf{r} + e_2 + \mathsf{Decompress}_q(m, 1)$
$c_1 := \mathsf{Encode}_{d_u}(\mathsf{Compress}_q(\mathbf{u}, d_u))$
$c_2 := \mathsf{Encode}_{d_v}(\mathsf{Compress}_q(v, d_v))$
return $c = (c_1 \| c_2)$ ▷ $c := (\mathsf{Compress}_q(\mathbf{u}, d_u), \mathsf{Compress}_q(v, d_v))$

Рис.5 – Шифрования для Kyber.CPAPKE

Algorithm 6 KYBER.CPAPKE.Dec(sk, c): decryption

Input: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$
Output: Message $m \in \mathcal{B}^{32}$

$\mathbf{u} := \mathsf{Decompress}_q(\mathsf{Decode}_{d_u}(c), d_u)$
$v := \mathsf{Decompress}_q(\mathsf{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
$\hat{\mathbf{s}} := \mathsf{Decode}_{12}(sk)$
$m := \mathsf{Encode}_1(\mathsf{Compress}_q(v - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \mathsf{NTT}(\mathbf{u})), 1))$ ▷ $m := \mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1))$
return m

Рис.6 – Дешифрование для Kyber.CPAPKE

Kyber.CCAKEM

Данный алгоритм получен из схемы шифрования с открытым ключом IND-CPA, которая описана в предыдущем подразделе, с помощью слегка измененного преобразования Фудзисаки-Окамото.

Algorithm 7 KYBER.CCAKEM.KeyGen()

Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Output: Secret key $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$
1: $z \leftarrow \mathcal{B}^{32}$
2: $(pk, sk') := \text{KYBER.CPAPKE.KeyGen}()$
3: $sk := (sk' \| pk \| \text{H}(pk) \| z)$
4: **return** (pk, sk)

Рис.7 – Генерация ключа для Kyber.CCAKEM

Algorithm 8 KYBER.CCAKEM.Enc(pk)

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$
Output: Shared key $K \in \mathcal{B}^*$
1: $m \leftarrow \mathcal{B}^{32}$
2: $m \leftarrow \text{H}(m)$ ▷ Do not send output of system RNG
3: $(\bar{K}, r) := \text{G}(m \| \text{H}(pk))$
4: $c := \text{KYBER.CPAPKE.Enc}(pk, m, r)$
5: $K := \text{KDF}(\bar{K} \| \text{H}(c))$
6: **return** (c, K)

Рис.8 – Инкапсуляция для Kyber.CCAKEM

Algorithm 9 KYBER.CCAKEM.Dec(c, sk)

Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$
Input: Secret key $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$
Output: Shared key $K \in \mathcal{B}^*$
1: $pk := sk + 12 \cdot k \cdot n/8$
2: $h := sk + 24 \cdot k \cdot n/8 + 32 \in \mathcal{B}^{32}$
3: $z := sk + 24 \cdot k \cdot n/8 + 64$
4: $m' := \text{KYBER.CPAPKE.Dec}(\mathbf{s}, (\mathbf{u}, v))$
5: $(\bar{K}', r') := \text{G}(m' \| h)$
6: $c' := \text{KYBER.CPAPKE.Enc}(pk, m', r')$
7: **if** $c = c'$ **then**
8: **return** $K := \text{KDF}(\bar{K}' \| \text{H}(c))$
9: **else**
10: **return** $K := \text{KDF}(z \| \text{H}(c))$
11: **end if**
12: **return** K

Рис.9 – Декапсуляция для Kyber.CCAKEM

4 Параметры Kyber

Всего определено 3 набора параметров: Kyber512, Kyber768, Kyber1024.

Помимо входных параметров можно увидеть дополнительный параметр, который был получен в ходе тестирования (δ): вероятность ошибки декапсуляции ключа.

Таблица 1. Параметры алгоритма Kyber

	n	k	q	η_1	η_2	(d_u, d_v)	δ
Kyber512	256	2	3329	3	2	(10, 4)	2^{-139}
Kyber768	256	3	3329	2	2	(10, 4)	2^{-164}
Kyber1024	256	4	3329	2	2	(11, 5)	2^{-174}

Значение n равно 256, поскольку цель состоит в том, чтобы инкапсулировать ключи с 256 битами энтропии, то есть использовать обычный текст размера 256 бит.

Меньшие значения n потребовали бы кодировать несколько ключевых бит в один полиномиальный коэффициент, что потребовало бы внесения меньших значений шума, а это напрямую повлияло бы на безопасность всего алгоритма.

Если же брать значения n больше 256, то это бы помешало масштабированию безопасности с помощью параметра k.

Параметр q выбирается как небольшое просто число, удовлетворяющее условию $n|(q-1)$. Такое условие необходимо для быстрого умножения на основе NTT. Есть два меньших простых числа, для которых это свойство тоже справедливо: 257 и 769.

Однако для этих простых чисел незначительная вероятность ошибки декапсуляции была бы недостижима, поэтому было выбрано следующее по величине простое число, т.е. q = 3329.

Параметр k выбирается для фиксации размерности алгебраической решетки, которая делится на n. Данный параметр является основным механизмом масштабирования уровня безопасности.

Остальные параметры выбраны для обеспечения баланса между безопасностью и вероятностью отказа. Стоить обратить внимания на тот факт, что все 3 набора параметров обеспечивают вероятность отказа меньше чем 2^{-128}

Параметр η_1 определяет уровень шума s и e в алгоритме 4, а в алгоритме 5 определяет уровень шума r.

Параметр η_2 определяет уровень шума e_1 и e_2 в алгоритме 5.

5 Ожидаемый уровень безопасности

Ранее упоминалось, что все наборы параметров Kyber имеют вероятность сбоя дешифрования. Эти сбои представляют собой угрозу безопасности, и поэтому вероятность их возникновения должна быть небольшой. Но поскольку в классической модели случайного оракула вероятность сбоя дешифрования является теоретико-информационной, авторы не видят необходимости в ее уменьшении по мере повышения уровня безопасности.

Стоит обратить внимание, что для Kyber768 и Kyber1024 вероятность ошибки меньше 2^{-160}. Это настолько малая величина, что данный аспект безопасности можно исключить из рассмотрения.

Что же касается уровней безопасности по требованиям NIST, варианты Kyber можно оценить по 5-балльной шкале как 1, 3 и 5:

1) Kyber512 – 1 балл, уровень безопасности соответствует AES-128
2) Kyber768 – 3 балла, уровень безопасности соответствует AES-192
3) Kyber1024 – 5 баллов, уровень безопасности соответствует AES-256

6 Достоинства и сравнения с другими алгоритмами и схемами

6.1 Достоинства

Начнем с того, что алгоритм имеет очень конкурентоспособные скорости, небольшой размер входных параметров и работает на хорошо изученной проблеме.

Также стоит отметить следующие преимущества:

1) **Простота реализации**: Оптимизированные реализации сосредоточены только на быстрых NTT преобразованиях и быстрой перестановке Keccak. Это даст очень конкурентоспособную производительность для всех наборов параметров Kyber.

2) **Масштабируемость**: переход от одного уровня безопасности к другому осуществляется крайне просто, всего лишь нужно поменять размерность матрицы, выборки шума и округление шифртекста с помощью различных параметров функции сжатия.

6.2 Сравнение с SIDH

Интересной альтернативой механизма инкапсуляции ключей на основе решетки является суперсингулярная изогения Диффи-Хеллмана (SIDH).

Очевидным преимуществом SIDH являются размеры открытых ключей и зашифрованных текстов, которые при подходящем сжатии примерно в 3 раза меньше, чем открытые ключи и зашифрованные тексты Kyber.

Недостатком SIDH является то, что он работает более чем в 2 раза медленнее, чем Kyber. К тому же схема довольно новая, что затрудняет проведение окончательных сравнений.

6.3 Сравнение с алгоритмами на основе кодов

При рассмотрении алгоритмов на основе кодов следует различать классические схемы, такие как схемы МакЭлиса и Нидеррайтера, на основе двоичных кодов Гоппы, и схемы с менее консервативным (но более эффективным) выбором кодов. Алгоритмы на основе кода Гоппы являются проверенными временем и могут считаться консервативным стандартом постквантового примитива. Тем не менее их развертывание во множестве сценариев затруднено большим размером ключа и большим временем генерации данного ключа.

Другим вариантом являются квазициклические коды с малой плотностью проверок на четность (QC-MDPC). Эти коды обеспечивают более близкую конкуренцию с точки зрения производительности, но при высоких уровнях защиты не могут обеспечивать (вероятно) достаточно малую вероятность отказа.

6.4 Сравнение с схемами на основе LWE

Если кто-то не хочет использовать алгебраические структуры в задаче LWE, то есть 2 пути.

Первым вариантом является использование одной из версий оригинальной схемы, придуманной Регевым. Но такой подход делает открытый и секретный ключи невероятно большими по сравнению с алгоритмов Kyber: порядка 1 мегабайта каждый. Шифртекст при этом оказывается примерно одного размера с шифртекстом Kyber. Учитывая размеры ключей, такая схема была бы крайне неэффективной при обмене ключами.

Другим вариантом является схема алгоритма Frodo. В данном алгоритме и шифртекст, и ключи имеют размер порядка 11 килобайт. Это примерно в 10 раз больше, чем в алгоритме Kyber. Время работы алгоритма Frodo примерно в 10 раз больше времени работы Kyber.

Таким образом, существующие схемы на основе LWE уступают по нескольким параметрам алгоритму Kyber. Но забывать о них не стоит, они являются хорошими резервными вариантами на случай появления эффективных алгоритмов решения проблемы MLWE.

6.5 Сравнение с схемами на основе RLWE

Другой альтернативой MLWE являются схемы на основе RLWE.

RLWE – это частный случай задачи MLWE, где ширина матрицы **A** над кольцом R всегда равно 1. Следовательно, чтобы повысить или понизить уровень безопасности, придется менять размер кольца, тогда как в алгоритме Kyber размер кольца фиксирован, а размер модуля меняется.

Как было упомянуто выше, одно из преимуществ подхода, который был выбран для Kyber, заключается в том, что нужно иметь только одну хорошую реализацию для реализации операций над кольцом. Изменение размера модуля просто предполагает выполнение большего количества (или меньшего количества) одних и тех же операций над кольцом.

А изменение кольца, потребовало бы повторной реализации всех операций.

Еще одним преимуществом работы с “малым” кольцом постоянной степени является то, что оно обеспечивает более точный компромисс между производительностью и безопасностью.

Самый простой и эффективный способ реализации RLWE – это работа над кольцами $\mathbb{Z}[X]/(X^n+1)$, где n - степень двойки. Поскольку n является единственным параметром, который непосредственно влияет на уровень безопасности, его ограничение степенями двойки может потребовать превышения необходимой границы безопасности. Например, уровень безопасности Kyber768 при таком подходе недостижим, придется брать выше требуемого.

Конечно, можно было бы работать непосредственно по модулю многочлена любой желаемой степени (с ограничением, заключающимся в том, что он должен быть неприводимым по $\mathbb{Z}$), но тогда безопасность немного снизилась бы из-за геометрии числовых полей, отличных от степени двойки.

Единственное преимущество RLWE перед Kyber заключается в том, что если **A** представляет собой матрицу $k \times k$, то для ее извлечения из начального значения требуется в k раз больше выходных данных XOF, чем для матрицы 1×1.

6.6 Сравнение с алгоритмом NTRU

По сравнению с Kyber, NTRU обладает всеми преимуществами и недостатками RLWE, но, кроме того, имеет еще два отрицательных момента. Первая генерация ключа NTRU значительно дороже, чем в RLWE, когда кольцо не поддерживает NTT. Причина в том, что генерация ключа NTRU требует полиномиальное деление, тогда как генерация ключа RLWE требует только умножения (если кольцо поддерживает NTT, то деление происходит ненамного медленнее, чем умножение).

Вторым возможным недостатком NTRU является то, что геометрия его базовой решетки приводит к атакам, которых не существует против схем RLWE или MLWE.

Хотя это свойство, по-видимому, не помогает в атаках на небольшие параметры, которые используются для определения криптосистемы NTRU, но это может указывать на возможную слабость, которая может быть использована в дальнейшем.

Одним из возможных достоинств использования NTRU является небольшое преимущество в производительности при шифровании (инкапсуляции), но, учитывая недостатки, это нельзя считать хорошим компромиссом.

Кроме того, невозможно определить эффективную версию Module-NTRU, которая позволила бы использовать преимущества Kyber, описанные выше в разделе 6.1.

ЗАКЛЮЧЕНИЕ

Результатом курсовой работы является составленная пояснительная записка к алгоритму CYSTALS-CYBER, который основан на схеме шифрования с открытым ключом, предназначенной для обеспечения безопасности как от классических атак, так и от квантовых. В ходе написания были описаны все ключевые моменты алгоритма, такие как: построения ключей, шифрование, дешифрование, описание предлагаемых параметров, анализ стойкости, анализ сложности реализации. Помимо этого, были выявлены и раскрыты его преимущества и недостатки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. In 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018. IEEE, 2018.
2. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 75(3):565–599, 2015.
3. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Advances in Cryptology - CRYPTO ’99, pages 537–554, 1999.
4. Jeffrey Hoffstein, Jull Pipher, and Joseph H. Silverman. NTRU: a ring-based public key cryptosystem. In Joe P. Buhler, editor, Algorithmic number theory, volume 1423 of LNCS, pages 267–288. Springer, 1998.
5. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In STOC ’05 Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pages 84–93. ACM, 2005.
6. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56(6):34, 2009.