

Создание постквантового протокола KEM на LWR

Цели и задачи:

Разработка отечественного постквантового протокола KEM на базе LWR.

Задачи:

- Анализ существующих решений
- Выбор оптимальной математической базы
- Создать математическую часть алгоритма
- Прототипирование и тестирование алгоритма

Практическая значимость:

Алгоритм вносит вклад в развитие отечественной криптографической базы, предоставляя возможность для дальнейшей стандартизации и внедрения защищённых решений в государственных и коммерческих информационных системах.

Термины:

KEM - механизм инкапсуляции ключей

LWR - обучение с округлением. Криптографическая схема

Кубит - квантовый бит. Имеет состояния суперпозиции

Актуальность:

В условиях стремительно развивающихся технологий в сфере квантовых компьютеров, ставится под угрозу современная криптография. По расчётам экспертов, компьютер с 4000 логическими кубитами способен взломать RSA-2048, что уже выглядит реальной перспективой. Развитие технологий требует перехода на постквантовые схемы, способные обеспечить надёжную защиту

Новизна:

Представленный протокол является первым Российским KEM, основанным на задаче LWR. Использование LWR позволяет упростить реализацию за счёт отказа от шумовых распределений, а также улучшить устойчивость к побочным каналам. Протокол сочетает подходы LWR и корректирующей подсказки (hint), что отличает его от зарубежных аналогов.

Аналитика:

National Institute of Standards and Technology		
Алгоритм	Основа	Статус
Kyber (FIPS 203)	LWE	Стандарт (2024)
HQC (в разработке)	Коды исправляющие ошибки	Резервный (2025)
Saber	LWR	В разработке
Технический комитет 26		
Кодиум	Коды исправляющие ошибки	В разработке
Российские разработки		
???	Решётки	Ранняя разработка
???	На изогениях	Ранняя разработка
???	Гибридные схемы	Ранняя разработка

На данный момент полностью стандартизирован только Kyber. Разработка и стандартизация заняли 8 лет (с 2016г).

Среди российских KEM относительно хорошие результаты показывает только Кодиум. Но у него большие открытые ключи (несколько КБ) и большое время работы (почти в 3 раза дольше, чем у зарубежных протоколов).

Остальные алгоритмы находятся только в ранней стадии разработки и в основном закрыты из-за их связи с гос структурами.

Алгоритмы взлома:

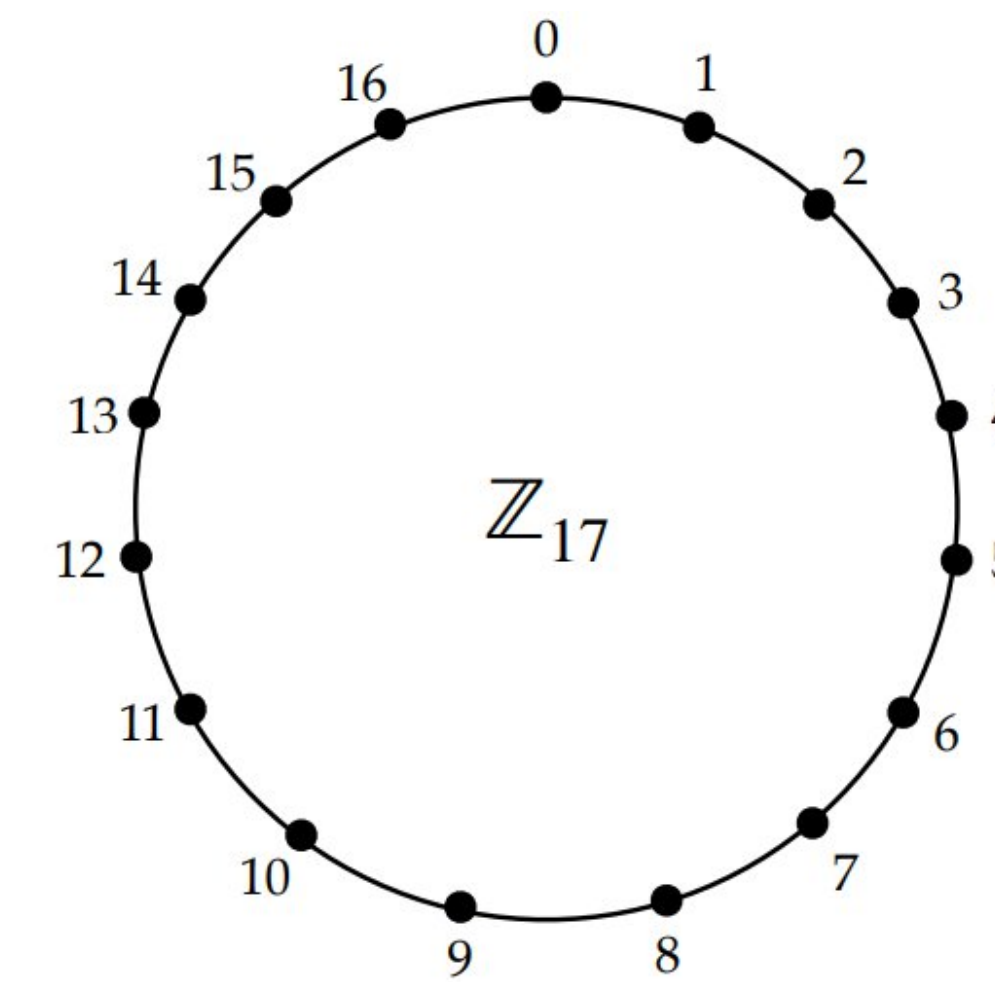
Алгоритм Шора

Задача факторизации сводится к поиску периода функции $f(x) = a^x \bmod N$, где N — число для разложения, a — случайное число, взаимно простое с N .
Позволяет факторизовать число N за полиномиальное время ($O(\log^3 N)$), используя $O(\log N)$ кубитов.

Алгоритм Гровера

Использует квантовую амплитудную интерференцию
Изменяет асимптотику перебора с $O(2^N)$ на $O(2^{N/2})$

$q = 2^{12}$ - модуль кольца
 $p = 2^5$ - точность после округления
 $n = 2^3$ - степень многочленов
 $d = \frac{q}{p}$ - делитель округления
 $offset = \frac{d}{2}$ - смещение
 $R = \frac{Z_q[x]}{x^n + 1}$ - кольцо многочленов
threshold = 126 - порог для подсказки



Ключевые формулы:

Округление: $Round(x) = \left\lfloor \frac{x + offset}{d} \right\rfloor \bmod p$

Умножение полиномов: $c_k = \sum_{i+j \equiv k \pmod n} a_i b_j (-1)^{\lfloor \frac{i+j}{n} \rfloor}$

Генерация ключей:

Секретный ключ: $s(x) \in R, s_i \in \{0, 1\}$

Публичный ключ:

Генерация $a(x) \in R$

Вычисление: $b(x) = Round(a(x) * s(x)) \bmod (x^n + 1)$

Публичный ключ $pk = (a(x), b(x))$

Инкапсуляция:

Сообщение $m \in \{0, 1\}^n$

Кодирование: $m^{enc}(x) = encode(m), m_i^{enc} = m_i * \frac{p}{2}$

Эфемерный секрет: $s'(x) \in R, s'_i \in \{0, 1\}$.

Вычисление: $u(x) = Round(a(x) s'(x) \bmod (x^n + 1))$.

Подсказка:

$b'(x) = b(x) \cdot s'(x) \bmod (x^n + 1)$

$h(x) = hint(b'(x))$

$h(x) = \begin{cases} 1, & \text{если } x \bmod d \geq threshold \\ 0, & \text{иначе} \end{cases}$

Шифротекст:

$v(x) = Round(b'(x)) + m^{enc}(x) \bmod p$

$ct = (u(x), v(x), h(x))$

Общий ключ: $K = Hash(serialize(m))$

Декапсуляция:

$w'(x) = u(x) s(x) \bmod (x^n + 1)$

$w(x) = Round(w'(x))$

Коррекция: $w'_i = \begin{cases} w_i + 1, & \text{если } h_i = 1 \text{ и } w_i < p - 1, \\ w_i - 1, & \text{если } h_i = 0 \text{ и } w_i > 0, \\ w_i, & \text{иначе.} \end{cases}$

Восстановление: $m_i^{enc(recovered)} = v_i - w'_i \bmod p$

Декодирование: $\hat{m}_i = \begin{cases} 1, & \text{если } m_i^{enc(recovered)} \geq \frac{p}{2}, \\ 0, & \text{иначе.} \end{cases}$

Общий ключ: $K' = Hash(serialize(m))$

Тестирование:

Ключевые аспекты проверки:

- Корректность восстановления ключа:**
При базовых настройках успешность составила 35%.
После изменения коэффициентов достигнута успешность 67%.
- Производительность:**
Время генерации ключей: <10 мс.
Скорость инкапсуляции/декапсуляции: <50 мс.

Инфраструктура:

Для анализа использовались локальные машины и облачные среды.
Данные визуализированы через *matplotlib* (графики зависимости).

Выводы:

Тестирование подтвердило, что протокол:

- Корректно функционирует в заданных условиях.
- Не всегда устойчив к ошибкам округления и устойчив к сбоям передачи данных.
- Для промышленного внедрения требуется:
Масштабирование параметров ($n \geq 512$).

