Криптография на решётках

Шифрование, основанное на решетках (или **решеточное шифрование**, **lattice-based encryption**), представляет собой криптографический метод, использующий математические структуры, называемые решетками. Этот подход стал популярным в последние десятилетия, особенно в контексте квантовой криптографии, поскольку предполагается, что он может быть более устойчивым к атакам с использованием квантовых компьютеров.

### Что такое решетка?

Решетка — это структура, образованная набором точек в многомерном пространстве. Математически решетку можно описать как множество линейных комбинаций базисных векторов. В двухмерном пространстве, например, решеткой может быть множество точек, полученных с помощью всех возможных целых чисел, умноженных на два базисных вектора.

Пример решетки в двумерном пространстве:

* Пусть базис решетки состоит из двух векторов  Тогда решетка будет включать все возможные целые линейные комбинации этих векторов, такие как (0,0),(1,0),(0,1),(1,1),(−1,0) и так далее.

шифрования с использованием задач теории решёток, то есть задач оптимизации на дискретных аддитивных подгруппах, заданных на множестве

**Итак, что же такое решётки?**

### Формальное определение решётки

Решётка L в n-мерном пространстве Rn задаётся как множество всех векторов, которые можно получить из линейных комбинаций n-мерных базисных векторов b1,b2,…,bk ​ с целыми коэффициентами:

L(B)={Z1Z1+Z2Z2+…+ZkKk∣Zi∈Z}

матрица базисных векторов.

* Если k=n, решётка называется **полной**.
* Если k<n, это **решётка с меньшей размерностью**, которая образует подпространство.

### Визуализация решётки

Решётку можно представить как регулярный "узор" точек в пространстве. Например:

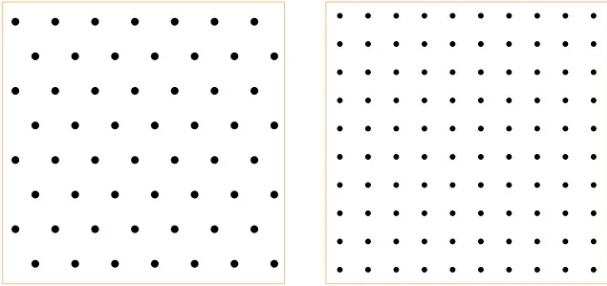
* В двумерном пространстве это может быть сетка точек на плоскости.
* В трёхмерном пространстве решётка напоминает трёхмерную сетку.

### Свойства решёток

1. **Периодичность**: Точки решётки повторяются с определённой регулярностью.
2. **Базис неоднозначен**: Для одной и той же решётки может существовать множество различных наборов базисных векторов.
3. **Кратчайший вектор**: Найти кратчайший ненулевой вектор в решётке (задача SIVP) — одна из важных задач в криптографии.

**Представление**

 Решётку можно представить как любую равномерно распределённую сетку точек, простирающуюся до бесконечности. Например, вот две разные двумерные решётки.



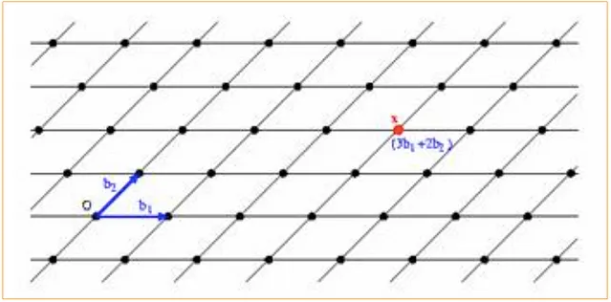
**Что такое вектор** - ~~и так все знают, так как проходили выш. Мат на первом курс-~~

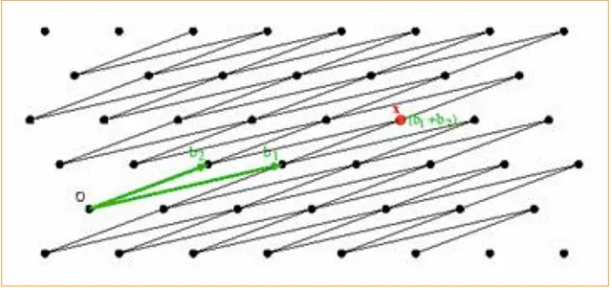
[математический объект](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82), характеризующийся величиной и направлением.

**Базис**: Решётки — это бесконечно большие объекты, но у компьютеров есть только ограниченный объём памяти для работы с ними. Поэтому нам понадобится краткий способ представления решёток, если мы собираемся использовать их в криптографии. Для этого мы используем так называемый базис решётки. Базис — это небольшая совокупность векторов, которые можно использовать для воспроизведения любой точки в сетке, образующей решётку.

Важно отметить, что в отличие от бесконечной сетки точек, базис — это простой конечныйобъект, который мы можем представить в памяти компьютера.

Вот приме 2 разных базисов:



****

**Как** **выглядит сложная задача о решётке?**

1. **Основная идея**: решеточные схемы используют математические проблемы, связанные с решетками, которые трудно решать даже с помощью квантовых компьютеров. Одной из таких проблем является задача нахождения кратчайшего вектора в решетке

2. **Процесс шифрования**:

* **Генерация ключей**: Пара ключей состоит из публичного и секретного ключей. Публичный ключ генерируется таким образом, чтобы задача нахождения определенных векторов в решетке была сложной.
* **Шифрование**: Для шифрования сообщения выбирается случайный вектор, который затем используется для создания зашифрованного сообщения. Процесс добавления шума к сообщению делает его трудным для расшифровки без знания секретного ключа.
* **Расшифровка**: Для расшифровки используется секретный ключ и свойства решетки, позволяющие найти оригинальное сообщение, несмотря на добавленный шум.

3. **Проблемы решеток**:

* **Shortest Vector Problem (SVP)**: Найти наименьший вектор в решетке — одна из наиболее сложных задач в решеточной криптографии.
* **Learning With Errors (LWE)**: Еще одна важная проблема, которая лежит в основе шифрования на решетках, заключается в том, чтобы научиться восстанавливать информацию из зашумленных данных.

**Преимущества решеточного шифрования**

**\***Устойчивость к квантовым атакам - предполагаемая стойкость против квантовых вычислительных атак.

\*Поддержка более сложных операций - позволяет выполнять вычисления над зашифрованными данными без их расшифровки

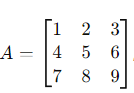
**Как происходит процесс шифрования? (Рассмотрим сразу на примере строки “Hi”)**

**1) Генерация ключей**

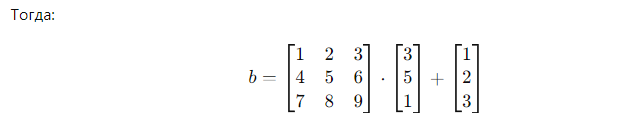
Предположим, что секретный и публичный ключи генерируются следующим образом:

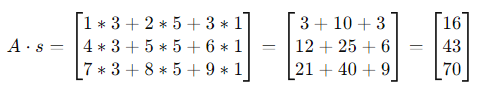
* **Секретный ключ** s — это случайный вектор. Пусть s=[3,5,1]
* **Публичный ключ** состоит из двух элементов: матрицы A и вектора b. Для простоты пусть A будет случайной матрицей размером 3x3, а b будет вычислен с использованием секретного ключа s.

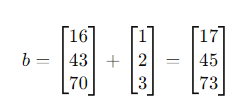
Пусть a & b вычисляются следующим образом:



е – вектор шума (ошибки), добавляемый для защиты. Допустим он равен e = [0,1,2]









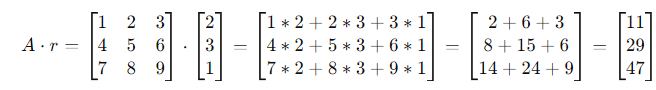
2) Шифрование сообщения:  
\*Представим строку “Hi” в числовом значении (ASCII): [72, 105]

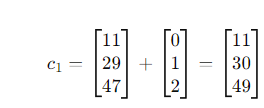
\*Наше сообщение будет представлено как вектор m = [72, 105]

\* Чтобы зашифровать сообщение выбираем случайный вектор r = [2, 3, 1]

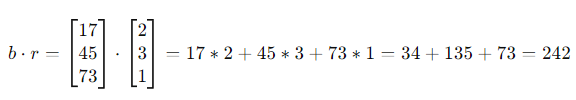
\* Теперь вычисляем зашифрованное сообщение используя публичный ключ А и случайный вектор r:

1. Первая часть зашифрованного сообщения будет вычисляться как: 

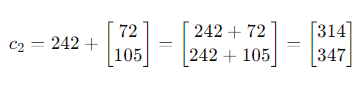
e=[0,1,2], Тогда A\*r : 



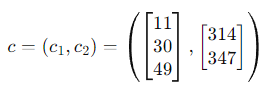
2. Вторая часть зашифрованного сообщения c2 будет вычисляться как: 



Добавляем сообщение m:



Таким образом финальное зашифрованное сообщение будет выглядеть: c = (c1, c2)



**3. Расшифровка сообщения**

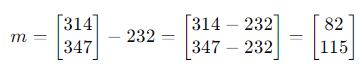
Для расшифровки используется секретный ключ s = [3, 5, 1]

1. Извлекаем первую часть зашифрованного сообщения и вычисляем s\*c1

2. Теперь извлекаем вторую часть зашифрованного сообщения

и вычитаем s\*c1 из c2:



3. Переводим эти числа в ASCII и получаем сообщение “Hi”

Реализация алгоритма на python с использование numpy

