Линейный конгруэнтный генератор (ЛКГ) — это способ создания \*\*псевдослучайных чисел\*\*. Это как машина, которая выдает числа, похожие на случайные, но на самом деле они следуют строгому математическому правилу.

- \*\*\(a\)\*\* (множитель) делает числа менее предсказуемыми. Если \(a\) слишком маленькое или неудачное, последовательность может быть короткой или повторяющейся.

- \*\*\(c\)\*\* (прибавка) добавляет разнообразия. Без \(c\) (если \(c = 0\)), последовательность может "застревать" в определенных значениях.

- \*\*\(n\)\*\* (модуль) задает диапазон чисел. Чем больше \(n\), тем больше разных чисел можно получить.

1. \*\*Текущее число (\(X\_n\))\*\*:

- Это число, с которого начинается расчет. Первое число называется \*\*seed\*\* (начальное значение). - На каждом шаге текущее число используется для вычисления следующего.

2. \*\*\(a\) (множитель)\*\*:

- Это число, на которое умножается текущее число. Оно "перемешивает" значение, чтобы последовательность выглядела случайной.

3. \*\*\(c\) (прибавка)\*\*:

- Это число, которое добавляется после умножения. Оно помогает сделать последовательность более разнообразной.

4. \*\*\(n\) (модуль)\*\*:

- Это число, которое ограничивает результат. После вычислений мы берем остаток от деления на \(n\) (операция \(\mod n\)). Это значит, что все числа в последовательности будут от 0 до \(n-1\).

- В вашем случае \(n = 11979\), так что числа будут от 0 до 11978.

5. \*\*\(\mod n\)\*\*:

- Это как "обрезка" результата. Например, если после умножения и прибавки получилось 15000, а \(n = 11979\), то \(\mod 11979\) дает остаток: \(15000 - 11979 = 3021\). Значит, следующее число — 3021.

RC4 — это способ шифрования текста, который придумал Рон Ривест в 1987 году. Он создает длинную последовательность чисел, которая выглядит случайной (называется псевдослучайный поток, или ПСП), и использует её, чтобы "замаскировать" текст. RC4 часто применяется там, где нужно быстро шифровать данные, например, в старых версиях Wi-Fi (WEP) или в некоторых протоколах.

RC4 состоит из двух главных частей: \*\*KSA\*\* (подготовка таблицы) и \*\*PRGA\*\* (генерация чисел для шифрования).

#### 1. Подготовка: KSA (Key-Scheduling Algorithm) — "перемешиваем таблицу"

Сначала RC4 создает таблицу \(S\), которая будет основой для генерации чисел.

- \*\*Шаг 1: Создаем таблицу \(S\)\*\*:

- У нас \(n = 8\), значит, таблица \(S\) имеет размер \(2^8 = 256\) (числа от 0 до 255).

- Заполняем её по порядку: \(S = [0, 1, 2, ..., 255]\).

- \*\*Шаг 2: Берем ключ\*\*:

- Ваш ключ: \(122, 125, 48, 84, 201\). Это массив из 5 чисел (байтов).

- Ключ короткий (5 чисел), а таблица большая (256 чисел), поэтому ключ будет повторяться: \(122, 125, 48, 84, 201, 122, 125, 48, ...\) столько раз, сколько нужно.

- \*\*Шаг 3: Перемешиваем таблицу \(S\)\*\*:

- Берем два указателя: \(i\) и \(j\). Начинаем с \(i = 0\), \(j = 0\).

- Проходим по всем числам от 0 до 255 (256 шагов):

- \(j = (j + S[i] + \text{ключ}[i \mod \text{длина ключа}]) \mod 256\).

- Меняем местами \(S[i]\) и \(S[j]\).

- Простыми словами: на каждом шаге мы берем число из таблицы \(S\), добавляем к нему значение ключа и текущее \(j\), "обрезаем" результат (через \(\mod 256\)), чтобы он был в пределах 0–255, и меняем местами два числа в таблице. Это делает таблицу \(S\) "перемешанной" и зависимой от ключа.

#### 2. Генерация чисел: PRGA (Pseudo-Random Generation Algorithm) — "делаем числа для шифрования"

Теперь, когда таблица \(S\) готова, мы используем её, чтобы генерировать псевдослучайные числа (ПСП).

- \*\*Шаг 1: Инициализация\*\*:

- Берем два указателя: \(i = 0\), \(j = 0\).

- Делаем копию таблицы \(S\), чтобы не испортить оригинал (в коде это \(S\_copy\)).

- \*\*Шаг 2: Генерируем числа\*\*:

- Для каждого числа, которое нам нужно:

- \(i = (i + 1) \mod 256\) (увеличиваем \(i\) и "обрезаем" до 0–255).

- \(j = (j + S[i]) \mod 256\).

- Меняем местами \(S[i]\) и \(S[j]\).

- Берем индекс \(t = (S[i] + S[j]) \mod 256\).

- Выдаем число \(S[t]\) — это одно число из ПСП.

Простыми словами: мы "ходим" по таблице \(S\), каждый раз меняя местами два числа и выбирая одно случайное число (\(S[t]\)). Эти числа и есть наш псевдослучайный поток (ПСП).

#### 3. Шифрование и дешифрование

- \*\*Шифрование\*\*:

- Берем текст, например, "Hello, RC4!".

- Преобразуем его в байты (например, в ASCII: \(H = 72, e = 101, l = 108, ...\)).

- Генерируем ПСП с помощью PRGA (ровно столько чисел, сколько байтов в тексте).

- Делаем операцию XOR (исключающее ИЛИ) между байтами текста и ПСП:

- Например, если байт текста \(72\), а байт ПСП \(139\), то \(72 \oplus 139 = 203\).

- Получаем зашифрованный текст (в байтах).

1. \*\*Особенность потоковых шифров\*\*:

Потоковые шифры шифруют данные побитно или побайтно, генерируя псевдослучайный поток (ПСП), который комбинируется с открытым текстом (обычно через XOR).

2. \*\*Преимущества и недостатки синхронных и асинхронных потоковых шифров\*\*:

Преимущества: независимость ПСП от шифруемого текста, высокая скорость.

Недостатки: требует синхронизации отправителя и получателя, чувствительность к потере данных.

Преимущества: ПСП зависит от шифруемого текста, не требует синхронизации.

Недостатки: сложнее в реализации, может быть медленнее.

3. \*\*Свойства генератора псевдослучайных чисел для криптографии\*\*:

- Длинный период (чтобы числа не повторялись быстро).

- Хорошее распределение (числа должны быть равномерно распределены).

- Непредсказуемость (без знания ключа нельзя предсказать следующее число).

- Высокая криптостойкость (устойчивость к атакам).

4. \*\*Характеристика линейных конгруэнтных генераторов (ЛКГ)\*\*:

ЛКГ генерирует числа по формуле \(X\_{n+1} = (a \cdot X\_n + c) \mod n\).

- Преимущества: простота, скорость.

- Недостатки: низкая криптостойкость, предсказуемость.

5. \*\*Найти \(a\), \(c\), \(n\) для ЛКГ, если \(x\_0 = 1\), \(x\_1 = 12\), \(x\_2 = 3\), \(x\_3 = 6\)\*\*:

Формула: \(x\_{i+1} = (a \cdot x\_i + c) \mod n\).

- \(x\_1 = 12 = (a \cdot 1 + c) \mod n \implies 12 = (a + c) \mod n\).

- \(x\_2 = 3 = (a \cdot 12 + c) \mod n \implies 3 = (12a + c) \mod n\).

- \(x\_3 = 6 = (a \cdot 3 + c) \mod n \implies 6 = (3a + c) \mod n\).

Решаем: из уравнений получаем \(n = 15\), \(a = 11\), \(c = 1\).

Проверка: \(x\_1 = (11 \cdot 1 + 1) \mod 15 = 12\), \(x\_2 = (11 \cdot 12 + 1) \mod 15 = 3\), \(x\_3 = (11 \cdot 3 + 1) \mod 15 = 6\).

6. \*\*Общая схема генератора ПСП на основе регистров сдвига с линейной обратной связью (РСЛОС)\*\*:

Схема: регистр из \(n\) ячеек, биты сдвигаются вправо, новый бит на входе — XOR некоторых битов (по обратной связи).

7. \*\*Синтез РСЛОС (а) 3210, б) 420, в) 5410, г) 520, д) 84320)\*\*:

- \*\*а) 3210\*\*: Регистр 4 бита, обратная связь \(x\_3 \oplus x\_2 \oplus x\_1 \oplus x\_0\). Период 15 (максимальный, полином \(x^4 + x^3 + x^2 + x + 1\)).

Таблица: 0001 → 1000 → 0100 → ...

- \*\*б) 420\*\*: 5 бит, \(x\_4 \oplus x\_2 \oplus x\_0\). Период 31.

- \*\*в) 5410\*\*: 5 бит, \(x\_5 \oplus x\_4 \oplus x\_1 \oplus x\_0\). Период 31.

- \*\*г) 520\*\*: 5 бит, \(x\_5 \oplus x\_2 \oplus x\_0\). Период 31.

- \*\*д) 84320\*\*: 8 бит, \(x\_8 \oplus x\_4 \oplus x\_3 \oplus x\_2 \oplus x\_0\). Период 255.

8. \*\*Первые 12 бит ПСП для 5410, начальное состояние 10101\*\*:

Регистр 5 бит, обратная связь \(x\_5 \oplus x\_4 \oplus x\_1 \oplus x\_0\):

10101 → 01010 → 00101 → 10010 → 01001 → 10100 → 01010 → 10101 → 11010 → 01101 → 10110 → 01011.

Первые 12 бит: \(1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0\).

9. \*\*Генератор ПСП на основе RSA\*\*:

Устройство: выбираются два простых числа \(p\), \(q\), \(n = p \cdot q\), начальное \(x\_0\) взаимно простое с \(n\), \(x\_{i+1} = x\_i^e \mod n\).

Криптостойкость: основана на сложности факторизации \(n\) (проблема RSA).

10. \*\*\(x\_1, x\_5, x\_9, x\_{11}\) по BBS (\(p = 11, q = 19, x = 3\))\*\*:

\(n = 11 \cdot 19 = 209\), \(x\_0 = 3\), \(x\_{i+1} = x\_i^2 \mod 209\):

\(x\_1 = 9\), \(x\_2 = 81\), \(x\_3 = 82\), \(x\_4 = 67\), \(x\_5 = 144\), \(x\_6 = 197\), \(x\_7 = 90\), \(x\_8 = 192\), \(x\_9 = 126\), \(x\_{10} = 208\), \(x\_{11} = 16\).

Ответ: \(x\_1 = 9\), \(x\_5 = 144\), \(x\_9 = 126\), \(x\_{11} = 16\).

11. \*\*Базовый алгоритм RC4\*\*:

RC4 генерирует ПСП:

- KSA: инициализирует таблицу \(S\) (0–255) и перемешивает её с ключом.

- PRGA: генерирует ПСП, меняя местами элементы \(S\) и выдавая байты.

- Шифрование: текст XOR с ПСП.

12. \*\*Принципы формирования истинных случайных последовательностей\*\*:

Основаны на "естественных случайностях":

- Физические процессы (шум в электронике, радиоактивный распад).

- Внешние события (движение мыши, интервалы нажатий клавиш).

- Природные явления (атмосферный шум).

Такие числа непредсказуемы, так как зависят от случайных физических факторов.