

# Реверсивные генераторы Reversible-RNG

- Анатолий М. Георгиевский, ИТМО

Для исследования нормальности могут применяться различные варианты генератора псевдослучайных чисел. Генераторы с измененным порядком следования бит, мы рассматриваем в разделе Скрамблеры. В этом разделе мы предлагаем расширить представление о генераторе, как об обратимой функции внутреннего состояния. Если функция допускает существование обратной, то генерацию можно обратить по шагам. Многие генераторы можно представить не только в виде функции `next()`, но и как функцию `prev()` которая дает предыдущее состояние. В общем случае тестирование генератора может включать функцию `jump()` - пропуск сегмента, и `prev()` - реверс генератора. Генераторы, для которых можно составить функцию обратного хода backward-RNG будем называть reversible-RNG. Обратный генератор в некоторых случаях получается не менее эффективным, чем прямой и может рассматриваться, как самостоятельный метод ГПСЧ.

- см. [пример обратимого генератора - reversible Xoroshiro-64/128](#)
- см. [пример обратимого генератора - reversible MWC64/128](#)

Операция `prev` - обратная для `next`, вместе с `jump` они образуют арифметику над счетчиком итераций. Если для генераторов типа MWC `next` аналогична операции умножения, а `jump` - возведению в степень, то `prev` - умножение на обратное число (деление).

Можно предположить, что генераторы прямого и обратного хода будут обладать аналогичными показателями в тестах нормальности, т.к. TestU01.

Идея обратимого (reversible) псевдослучайного генератора — полезная концепция, находит применение в нескольких областях:

- adjoint Monte Carlo / обратные симуляции (физика, молекулярная динамика, теплообмен)
- рандомизация с undo/redo (игры, симуляции)
- рандомизация в обратимых (квантовых) вычислениях
- криптоанализ (когда нужно восстановить предыдущее состояние)
- тестирование и отладка RNG-последовательностей

В контексте обратимых (квантовых) вычислений, мы отсылаем к методу *Stochastic Rounding* с использованием reversible-PRNG. Этот метод крайне важен в задачах обучения нейросетей на весах малой разрядности. Именно внедрение метода SR позволяет не терять точность при использовании вычислений низкой разрядностью Float, т.к. FP4.

Метод реверсивной генерации ГПСЧ может найти место в методах пост-квантовой криптографии основанной на квантовом шуме. Пост-квантовые алгоритмы защиты данных основанные на обучении с ошибками, с добавлением квантового шума, Ring-LWE.

## Метод обращения генератора

Метод мы разберем на примере простых функций. Возможность вывода обратных преобразования для генераторов типа LFSR, LCG, PCG, MWC может быть показана аналитически, через существование обратных чисел в модульной арифметике и полиномиальной арифметики с использованием нередуцируемых полиномов. В тех случаях, когда преобразование выполняется по модулю простого числа - обратное число существует и единственно для каждого числа принадлежащего множеству.

Для линейного конгруэнтного генератора (LCG)

$$x_k = a \cdot x_{k-1} + c \bmod p$$

Обратный ход выражается через обратное число

$$x_{k-1} = a^{-1} \cdot (x_k - c) \bmod p$$

[2302.02778] Reversible random number generation for adjoint Monte Carlo simulation of the heat equation

## Обращение раундов DRNG генераторов

Мы рассмотрим возможность обращения различных хэш-функций, в частности нас интересует возможность использования в качестве генератора DRNG `CTR-Chacha` или `Hash-DRNG` на основе `SHA256`, когда линейный счетчик продевается через криптографическую функцию. Для таких генераторов также возможно предложить обратную функцию.

В качестве примера рассмотрим обратимость преобразования `ROUND` в алгоритме `SHA256`

В алгоритмах используются операции  $x = \text{rotl}(x, n)$ , при обратном ходе они заменяются на обратные  $x = \text{rotr}(x, n)$ . Векторные и матричные операции пермутации, типа транспонирования и перестановки строк и столбцов, заменяются на обратные - это очевидно. В хэш функциях могут использоваться матричные перестановки, которые можно выразить через обратные матрицы. В тех случаях, когда применяется нелинейная биекция (bijection), зачастую можно обойти путем подстановки значения прямого преобразования, как в приведенном ниже примере.

Прямое `ROUND(a,b,c,d,e,f,g,h,i)`:

```

h += K[i] + W[i&15];
uint32_t x = Sum1(e) + Ch (e,f,g);
uint32_t y = Sum0(a) + Maj(a,b,c);
d += x + h;
h += x + y;

```

Некоторые выражения  $x = \dots$  не меняют состояние, такие выражения остаются без изменения. В тех случаях, когда  $x = f(x, \dots)$ , значение заменяется, надо использовать обратную операцию.

Обратное: поменять порядок операций, заменить операторы `+=` на обратные `-=`, получаем:

```

uint32_t x = Sum1(e) + Ch (e,f,g);
uint32_t y = Sum0(a) + Maj(a,b,c);
h -= x+y;
d -= x+h;
h -= K[i]+W[i&15];

```

- см. [исходный код SHA256-rev](#)

Циклы заменяются так, чтобы переменная цикла `i` шла в обратном порядке.

Другой пример, базовая операция ARX (Add-Rotate-Xor) в Salsa20 представлена как

```

x0 ^= ROTL(x1 + x2, 18);

```

При обратном ходе идет без изменений, потому что операция `^=` обратна самой себе, но порядок применения операций в цикле - обратный.

ARX структура в ChaCha20

```

b = ROTL(b^c, 7);

```

заменяется на

```

b = ROTR(b, 7)^c;

```

со сменой направления циклического сдвига

- см. [исходный код Salsa20-rev](#)

- см. [исходный код ChaCha20-rev](#)

*Пример шифра.* Алгоритм ГОСТ Магма построен на операции ARX и биекции (S-box) в цикле.

```
n2 ^= ROTL(S(k[i]+n1), 11);
```

Обратная функция не использует обратное вычисление биекции, обращение выполняется на прямых операциях благодаря тому, что в правой части выражения не используется значение `n2`. В качестве упражнения можно выполнить обращение функции `Encrypt` по предложенной методике и сравнить результат со стандартом.

Реализация обратного шага ( `prev()` ) для `xoroshiro64/128` (включая варианты скрамблеров `+`, `++`, `**`) показана без необходимости хранения промежуточных состояний — восстановление предыдущего состояния выполняется исключительно на основе текущего состояния за один шаг и не имеет ограничений по числу итераций.

Для криптографических примитивов (`ChaCha20/Salsa20 quarter-round`, `ARX-based RNG`) показана обратимость заданного фиксированного числа раундов при условии знания входного состояния и добавляемых констант/ключа. Полное обращение длинной последовательности без знания ключа остаётся вычислительно неосуществимым. Мы предлагаем рассматривать обращение раундов, как инструмент вероятностного криптоанализа функций.

**Вероятностный тест совместимости Forward/Backward состояний.** Описание теста: В качестве основы используется маска установленных бит на входе и выходе криптографической функции, для которой можно определить функцию обращения раунда. Тест возвращает вероятность, метрику совместимости двух состояний. Если все биты определены и тест проходит полное число раундов, тест возвращает единицу. Число раундов функции является параметром теста. Тест является вероятностным и допускает контролируемый уровень ложноположительных срабатываний.

см. также аддитивный дифференциальный криптоанализ ARX-структур. Выявить связь между малым изменением на входе и измерением на выходе при использовании небольшого числа раундов ARX-структур (использующих сложение, ротацию бит и побитовую операцию XOR и перестановки слов).