УДК 004.421.5

МОДИФИКАЦИЯ АДДИТИВНОГО ГПСЧ И ЕГО РЕАЛИЗАЦИЯ НА ЯЗЫКЕ C#

Н.Е. Мыздриков, Е.А. Голоднов
Л.В Черкесова, Е.А. Ревякина
Донской государственный технический университет, Ростов–на–Дону, Российская Федерация
nmyzdrikov@mail.com,
golodnov@spark-mail.ru
chia2002@inbox.ru,
revyelena@yandex.ru

В статье представлен обзор аддитивного и инверсно-конгруэнтного ГПСЧ, а так же их модификация для возможного использования в криптосистемах. Приведен новый алгоритм генерации случайных последовательностей чисел и представлен код разработки на языке C#. Результат подкреплен тестами.

Ключевые слова: случайность, генератор случайных чисел, поля, алгоритм Евклида, аддитивный ГПСЧ, инверсно-конгруэнтный ГПСЧ

UDC 004.421.5

MODIFICATION OF THE ADDITIVE RNG AND ITS REALIZATION IN THE PROGRAMMING LANGUAGE C#

N.E. Myzdrikov, E.A. Golodnov
L.V. Cherkesova, E.A. Revyakina
Don State Technical University, Rostov-on-Don, Russian Federation
nmyzdrikov@mail.com,
golodnov@spark-mail.ru
chia2002@inbox.ru,
revyelena@yandex.ru

The article presents an overview of additive and inversely congruent RNG, as well as their modification for possible use in cryptosystems. A new algorithm for generating random sequences of numbers is presented and the development code in C# is given. The result is supported by tests.

Key words: randomness, random number generator, fields, Euklid algorithm, additive RNG, inverse-congruential RNG.

Введение. Для многих цифровых систем необходимым условием для их постоянной и бесперебойной работы является наличие в них генератора случайных чисел (далее ГСЧ). Но проблема идеальных ГСЧ заключается в том, что восстановить сформированную случайную последовательность (далее СП) невозможно, если ГСЧ основан на физических явлениях. Из этого следует, что в тех случаях, когда СП необходимо будет восстановить снова на основании каких либо начальных данных, используются другой способ формирования СП, а именно с помощью генераторов псевдослучайных последовательностей (далее ГПСЧ). Такие генераторы способны не просто создать псевдослучайную последовательность (далее ПСП), которая по всем характеристикам будет неотличима от СП [1], но и восстановить ее, если это потребуется.

В наши дни потребность в ГПСЧ очень высока [2], так как они используются в сфере разработки программного обеспечения и в разработке игр. Эти генераторы должны обладать большой продуктивной способностью и не затруднять работу программ своими вычислениями цифровой последовательности.

Некоторые ГПСЧ используются и в криптографии, для защиты данных от потерь. Например, может быть использован метод гаммирования между ПСП и данными, которые требуется зашифровать, в таком случае криптостойкость такого шифра будет зависеть только от сложности восстановления ПСП.

Аддитивный ГПСЧ применяется в составе криптосистем, которые используются крупневшими банками, занимающимися электронной коммерцией.

Из написанного выше следует, что разработки по теме ГПСЧ являются актуальными и востребованными в наше время, на что и нацелена эта работа.

В данной статье будет рассмотрен Линейный конгруэнтный метод и Инверсный конгруэнтный метод, а так же их возможные модификации. Главным достоинством этих методов является скорость генерации ПСП.

Целью данной работы является исследование конгруэнтного метода, реализация его алгоритма на языке C# и анализ его производительности в криптосистемах, использующих в своем составе ГПСЧ.

В процессе достижения поставленной цели работы авторами были сформулированы и успешно решены следующие задачи: Линейный конгруэнтный метод и Инверсный конгруэнтный метод, выявление возможных модификаций этих методов, анализ времени работы методов по отношению к другим известным ГПСЧ.

Основная часть

Для задач в разных сферах науки нередко требуются случайные числа или последовательности чисел, например, для моделирования некоторых физических явлений.

Различают генераторы случайных последовательностей и генераторы псевдослучайных последовательностей.

Генераторы случайных последовательностей – генераторы, основанные на некоторых физических явлениях. Они фиксируют состояние окружающей среды в текущий момент времени и выдают на этом основании следующий член последовательности. Генераторы псевдослучайных последовательностей – это детерминированные алгоритмы, которые генерируют числа по строго определенным правилам. Именно ГПСЧ имеют наибольшее распространение, потому что их легко восстановить по исходным данным [3].

Теория псевдослучайных генераторов – раздел математики, занимающийся изучением методов и алгоритмов генерации псевдослучайных последовательностей. В основе большинства алгоритмов лежат вычисления в полях.

Простые поля

Множество F с введенными на нем алгебраическими операциями сложения «+» и умножения «×» ($+:F\times F\to F, *:F\times F\to F$,т.е.$\forall a,b\in F\ (a+b)\in F, a*b\in F$) называется полем $\langle F,+,*\rangle$, если выполнены следующие аксиомы:

1. Коммутативность сложения: $\forall a,b\in F$: $a+b=b+a$;
2. Ассоциативность сложения: $\forall a,b,c\in F$: $(a+b)+c=a+(b+c)$;
3. Существование нулевого элемента: $\exists 0\in F:\forall a\in F\ a+0=0+a=a$;
4. Существование противоположного элемента: $\forall a\in F\ \exists\ (-a)\in F: a+(-a)=0$;
5. Коммутативность умножения: $\forall a,b\in F: a*b=b*a$;
6. Ассоциативность умножения: $\forall a,b,c\in F:(a*b)*c=a*(b*c)$;
7. Существование единичного элемента: $\exists e\in F\{0\}:\forall a\in F\ a*e=a$;
8. Существование обратного элемента для ненулевых элементов: $(\forall a\in F: a\neq 0)$ $\exists a-1\in F: a*a-1=e$;
9. Дистрибутивность умножения относительно сложения: $\forall a,b,c\in F\ (a+b)*c=(a*c)+(b*c)$.

Генератор псевдослучайных чисел

Генератор псевдослучайных чисел – алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются равномерному распределению [2], [3].

К идеальной модели ГПЧ предъявляют требования, которые гарантируют его корректную работу:

- Длинный период, гарантирующий отсутствие зацикливания последовательности;

- Быстрота работы алгоритма и малые затраты памяти;
- Возможность заново воспроизвести ранее сгенерированную последовательность;
- Одинаковое функционирование на различном оборудовании и ОС.

Применение случайности

Псевдослучайные последовательности имеют широкое применение в различных отраслях науки и программирования.

В науке ПСГ нашли применение в методе статистического анализа, позволяющем определить подлинность того или иного зафиксированного сигнала. Также в моделировании при проведении экспериментов, связанных с симуляцией реальных явлений, ПСГ тоже используются [4].

Немалое значение ПСГ имеют в криптографии. В ней используется непредсказуемость случайных чисел, с помощью которой можно обеспечить безопасность в современных коммуникациях, например, в области электронной коммерции.

Линейный конгруэнтный метод

Линейный конгруэнтный метод – метод генерации псевдослучайных чисел, используемый в стандартных библиотеках компиляторов [1].

Суть метода заключается в вычислении последовательности случайных чисел X_n, полагая, что $X_{n+1} = (aX_n + c)\ mod\ m$, где m – размерность поля, относительно которого вычисляют остаток от деления (m – простое число), a – множитель ($0 \le a < m$), c – приращение ($0 \le c < m$), X_0 – начальное значение ($0 \le X_0 < m$). Сгенерированная последовательность называется линейной конгруэнтной последовательностью [6].

Инверсный конгруэнтный метод

Инверсный конгруэнтный метод является модификацией линейного конгруэнтного метода. Основным отличием от линейного метода является использование при генерации последовательности числа, обратного к предыдущему элементу, вместо предыдущего элемента [7].

Параметрами генератора являются X_0 – исходный элемент, a – множитель ($0 \le a < n$), b – приращение ($0 \le b < n$), где n – размерность поля.

В случае простого числа n члены последовательности задаются в виде $x_{n+1} = a * x_n^{-1} + b \bmod m$, если $x_n \neq 0$, или $x_n + 1 \equiv b$ в противном случае.

Аддитивный ГПСЧ

Идею создать ГПСЧ на основе рекуррентного соотношения, где каждое последующее значение зависит более чем от одного предыдущего элемента, вывели еще давно. Такой простейшей последовательностью является последовательность Фибоначчи, которую можно описать в виде формульного соотношения

$$X_{n+1} = X_n + X_{n-1} \bmod m$$

Максимальная длина последовательности, сгенерированная таким способом, не может быть больше m^2, и зачастую последовательность, описанная формулой выше, выдает длину последовательности больше чем m [2], [5]. Но такая последовательность не обладает достаточной случайностью.

Возможные модификации

Целью данного исследования является модификация существующего аддитивного алгоритма ГПСЧ. Модифицированный алгоритм должен иметь больший период ЧП, иметь лучшее распределение элементов и проходить статистические тесты. В ходе работы авторами был разработан модифицированный аддитивный ГПСЧ, в котором так же присутствуют принципы инверсно-конгруэнтного метода. Каждый последующий член последовательности формируется по следующей рекуррентной формуле.

$$X_n = a * X_{n-4}^{-1} + X_{n-3} + b * X_{n-2}^{-1} + X_{n-1}$$

Таким образом, для генерации последовательности требуется задать начальные значения последовательности и два коэффициента, значения которых остаются неизменными в течение генерации всей последовательности.

В получившимся многочлене первый и третий элемент являются обратными элементами от значений на X_{n-i} позиции.

Для построения последующего элемента был сформулирован алгоритм:

1. **Ввод:** значение поля *P* (целое, положительное и простое число).
2. **Ввод:** константы a *и* b.
3. **Ввод:** начальные значения последовательности X_1, X_2, X_3, X_4.
4. **Создать** массив **List** который будет в себе содержать все *X*.
5. Добавить в **List** значения X_1, X_2, X_3, X_4.
6. $i =$ *Длина* **List**.
7. **Цикл ПОКА** $X_1 \neq X_{i-3}$ **И** $X_2 \neq X_{i-2}$ **И** $X_3 \neq X_{i-1}$ **И** $X_4 \neq X_i$
   Добавить в **List** новое значение = $X_{i-4}^{-1} + a * X_{i-3} + X_{i-2}^{-1} + b * X_{i-1}$
   i=i+1;

Конец цикла

Вывести содержимое **List.**

Реализация приведенного выше алгоритма на языке C#:

```
private List<int> Method(int P, int a, int b, int X1, int X2, int X3, int X4)
{
   List<int> list = new List<int>();
   list.AddRange(new[] { X1, X2, X3, X4 });
   int i = list.Count();
   while (X1 != list[i - 4] && X2 != list[i - 3] && X3 != list[i - 2] && X4 != list[i - 1])
   {
      list.Add(Reverse(list[i - 4],P)+a* list[i - 3]+Reverse(list[i - 2],P) +b* list[i - 1]);
      i++;
   }
   return list;
}
private int Reverse(int a, int pole)
{
   if (a == 0)
      return a;

   int g = GCD(a, pole, out int x, out int y);
   if (g != 1)
      throw new ArgumentException();
   return (x % pole + pole) % pole;
}
private  int GCD(int a, int b, out int x, out int y)
{
   if (a == 0)
```

```
{
   x = 0;
   y = 1;
   return b;
}
int x1, y1;
int d = GCD(b % a, a, out x1, out y1);
x = y1 - (b / a) * x1;
y = x1;
return d;
 }
```

В получившимся алгоритме так же используется метод поиска наименьшего общего делителя, для вычисления обратного числа в поле.

Результаты тестирования

Модификация аддитивного метода при тестировании авторами дала следующие результаты.

Сгенерированная модифицированным методом последовательность при одинаковых параметрах дает большую длину уникальной последовательности. Как было выяснено, длина уникальной последовательности увеличивается в среднем на 37% в сравнении со стандартным аддитивным методом. Результаты сравнения приведены в таблице №1.

Таблица №1

Сравнение длин последовательностей двух алгоритмов.

Размерность поля	19	31	257	997	2017	4001	65537
Аддитивный	1856	3892	9600	28942	138964	298520	988788
Модифицированный	6540	5678	13600	44890	192880	378576	1344124

Большая длина последовательности позволяет выбрать величину исходных параметров модифицированного алгоритма так, чтобы размерность поля и коэффициенты были намного меньше. Это позволяет использовать этот алгоритм на устройствах с низкой разрядностью процессора.

Также преимуществом модифицированного метода является то, что он исключает аномалии, возникающие в инверсном и аддитивном методах. Нередки ситуации, когда сгенерированная последовательность содержит в себе “шаблон”, который не нарушает уникальность последовательности в целом, но портит ее случайность на коротких промежутках.

Для подтверждения качества сгенерированной последовательности авторы взяли короткий отрезок последовательности над полем 31 и проверили частоты появления чисел, сгенерированных аддитивным ГПСЧ и модифицированным ГПСЧ. Последовательности чисел были взяты одинаковой длины. Как видно на рисунке 1, частоты появления символов модифицированного ГПСЧ распределены более равномерно, чем у аддитивного ГПСЧ. Это значит, что каждый последующий член модифицированной последовательности является более непредсказуемым.

```
Введите е сумматоры через запятую.
12,30,7,19


Размерность поля: 31


Для Аддитивного ГПСЧ: 397 повторений числа 0  ***  Для модифицированного ГПСЧ: 395 повторений числа 0
Для Аддитивного ГПСЧ: 402 повторений числа 1  ***  Для модифицированного ГПСЧ: 384 повторений числа 1
Для Аддитивного ГПСЧ: 404 повторений числа 2  ***  Для модифицированного ГПСЧ: 407 повторений числа 2
Для Аддитивного ГПСЧ: 390 повторений числа 3  ***  Для модифицированного ГПСЧ: 394 повторений числа 3
Для Аддитивного ГПСЧ: 408 повторений числа 4  ***  Для модифицированного ГПСЧ: 403 повторений числа 4
Для Аддитивного ГПСЧ: 391 повторений числа 5  ***  Для модифицированного ГПСЧ: 394 повторений числа 5
Для Аддитивного ГПСЧ: 400 повторений числа 6  ***  Для модифицированного ГПСЧ: 390 повторений числа 6
Для Аддитивного ГПСЧ: 388 повторений числа 7  ***  Для модифицированного ГПСЧ: 384 повторений числа 7
Для Аддитивного ГПСЧ: 406 повторений числа 8  ***  Для модифицированного ГПСЧ: 399 повторений числа 8
Для Аддитивного ГПСЧ: 388 повторений числа 9  ***  Для модифицированного ГПСЧ: 400 повторений числа 9
Для Аддитивного ГПСЧ: 396 повторений числа 10 ***  Для модифицированного ГПСЧ: 392 повторений числа 10
Для Аддитивного ГПСЧ: 411 повторений числа 11 ***  Для модифицированного ГПСЧ: 400 повторений числа 11
Для Аддитивного ГПСЧ: 386 повторений числа 12 ***  Для модифицированного ГПСЧ: 392 повторений числа 12
Для Аддитивного ГПСЧ: 390 повторений числа 13 ***  Для модифицированного ГПСЧ: 385 повторений числа 13
Для Аддитивного ГПСЧ: 406 повторений числа 14 ***  Для модифицированного ГПСЧ: 402 повторений числа 14
Для Аддитивного ГПСЧ: 386 повторений числа 15 ***  Для модифицированного ГПСЧ: 394 повторений числа 15
Для Аддитивного ГПСЧ: 386 повторений числа 16 ***  Для модифицированного ГПСЧ: 378 повторений числа 16
Для Аддитивного ГПСЧ: 406 повторений числа 17 ***  Для модифицированного ГПСЧ: 394 повторений числа 17
Для Аддитивного ГПСЧ: 390 повторений числа 18 ***  Для модифицированного ГПСЧ: 388 повторений числа 18
Для Аддитивного ГПСЧ: 386 повторений числа 19 ***  Для модифицированного ГПСЧ: 413 повторений числа 19
Для Аддитивного ГПСЧ: 411 повторений числа 20 ***  Для модифицированного ГПСЧ: 393 повторений числа 20
Для Аддитивного ГПСЧ: 396 повторений числа 21 ***  Для модифицированного ГПСЧ: 402 повторений числа 21
Для Аддитивного ГПСЧ: 388 повторений числа 22 ***  Для модифицированного ГПСЧ: 418 повторений числа 22
Для Аддитивного ГПСЧ: 406 повторений числа 23 ***  Для модифицированного ГПСЧ: 403 повторений числа 23
Для Аддитивного ГПСЧ: 388 повторений числа 24 ***  Для модифицированного ГПСЧ: 397 повторений числа 24
Для Аддитивного ГПСЧ: 400 повторений числа 25 ***  Для модифицированного ГПСЧ: 410 повторений числа 25
Для Аддитивного ГПСЧ: 391 повторений числа 26 ***  Для модифицированного ГПСЧ: 400 повторений числа 26
Для Аддитивного ГПСЧ: 408 повторений числа 27 ***  Для модифицированного ГПСЧ: 398 повторений числа 27
Для Аддитивного ГПСЧ: 390 повторений числа 28 ***  Для модифицированного ГПСЧ: 394 повторений числа 28
Для Аддитивного ГПСЧ: 404 повторений числа 29 ***  Для модифицированного ГПСЧ: 393 повторений числа 29
Для Аддитивного ГПСЧ: 402 повторений числа 30 ***  Для модифицированного ГПСЧ: 398 повторений числа 30


Размер Аддитивной ГПСЧ = 61568
Размер модифицированной ГПСЧ = 72294
```

Рис. 1 – Распределения чисел аддитивного и модифицированного ГПСЧ

Заключение. Генерация хороших псевдослучайных последовательностей остается важной задачей на сегодняшний день, так как многие программные продукты используют случайные числа. Реализованный в ходе работы программы алгоритм эффективно справляется с этой задачей.

Основной инновацией в проделанной работе является алгоритм генерации случайной последовательности, позволяющий получить более длинную уникальную последовательность в сравнении с аддитивным алгоритмом при равных размерах полей. Вторым достоинством разработанного алгоритма можно считать исключение аномалий последовательности на коротких промежутках, и это было подтверждено тестированием. Так же алгоритм может быть реализован в бинарном представлении, что так же позволит сократить затраты машинных ресурсов.

По итогам проделанной работы можно сделать вывод о пригодности и возможности эффективного использования модифицированного аддитивного метода в криптосистемах, используемых в банковских системах электронной коммерции.

Библиографический список

1) М. А. Иванов, И. В. Чугунков. Глава 4. Методика оценки качества генераторов ПСП // Теория, применение и оценка качества генераторов псевдослучайных последовательностей. — М.: КУДИЦ-ОБРАЗ, 2013. — 240 с. — ISBN 5-93378-056-1.

2) В. А. Успенский. Четыре алгоритмических лица случайности. — МЦНМО, 2015. — 48 с. — ISBN 978-5-94057-485-9.

3) L'Ecuyer, Pierre. Random Number Generation // Springer Handbooks of Computational Statistics: Глава. — 2012. — С. 93 - 137.

4) Intel® Digital Random Number Digital Random Number Generator Generator(DRNG). Software Implementation Guide. Revision 1.1. August 7, 2012. [Электронный ресурс]. Режим доступа:https://software.intel.com/sites/default/files/m/d/4/1/d/8/441_Intel_R__DRNG_Software_Implementation_Guide_final_Aug7.pdf

5) Бабаш А.В., Шанкин Г.П. Криптография. Под редакцией В.П. Шерстюка, ЭЛ. Применко / А.В. Бабаш, Г.П. Шанкин. - М.: СОЛОН-ПРЕСС, 2007. - 512 с. - (Серия книг «Аспекты защиты»). ISBN 5-93455-135-3

6) Dunkelman Orr Related-Key Attacks / Department of Computer Science, University of Haifa, Faculty of Mathematics and Computer Science, Weizmann Institute of Science: журнал. — 2014 – No.1.

7) Moon, D. Impossible differential cryptanalysis of reduced round XTEA and TEA/ Hwang K., Lee W., Lee S., Lim J., Lecture Notes in Computer Science 2365: журнал. — 2018. — 49–60. DOI:10.1007/3-540-45661-9_4