А.А. Радостев, М.С. Цыганова

Тюменский государственный университет

УДК 519.688

ГЕНЕРАЦИЯ ИСТИННО СЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА ОСНОВЕ КВАНТОВЫХ ВЫЧИСЛЕНИЙ

Аннотация. В статье анализируется возможность использования квантовых вычислений для генерации истинно случайных последовательностей. Представлены описание и реализация алгоритма, разработанного в рамках этого подхода, а также алгоритм шифрования данных, использующий результаты генерации. Обозначены перспективы использования квантовых вычислений.

Ключевые слова: истинно случайные последовательности, квантовые вычисления, шифрование данных, алгоритм Blowfish.

Введение

Последовательности случайных (либо псевдослучайных) чисел находят широкое применение в различных областях, связанных с использованием информационных технологий. Примерами таких областей являются криптография, сети и коммуникации, статистическое и имитационное моделирование, математическая статистика. Наиболее остро вопрос об использовании последовательностей истинно случайных чисел встает при обеспечении информационной безопасности (генерация ключей шифрования) или получении более реалистичных прогнозов в математическом моделировании.

Генераторы случайных последовательностей чисел делятся на две категории:

- аппаратные,

- программные.

Программные методы позволяют сгенерировать псевдослучайные (воспроизводимые) последовательности чисел; для получения истинно случайных чисел необходимо применять аппаратные методы. Аппаратная генерация случайных последовательностей подразумевает применение каких-либо дополнительных физических устройств. Для получения последовательностей такие устройства используют следующие источники энтропии:

- тепловой и электрический шум;
- квантовые процессы;
- радиоактивный распад;
- космическое излучение;
- различные механические, оптические и фотоэлектрические явления.

Подобные методы генерации часто являются довольно затратными как в плане финансов, так и в плане времени. В то же время можно предположить, что разумное применение современных технологий позволит снизить указанные затраты и повысить эффективность генерации истинно случайных последовательностей. В частности, квантовые компьютеры позволяют решить проблемы хранения в непосредственной близости аппаратного генератора, а также проблему времени генерации.

Постановка задачи

В данной работе была поставлена цель рассмотреть возможность генерации истинно случайных последовательностей с помощью квантовых вычислений, а также проанализировать состоятельность данного метода генерации. Для достижения поставленной цели необходимо решить ряд задач:

- разработать алгоритм генерации истинно случайных последовательностей, имеющих распределение, близкое к равномерному, на языке квантовых вычислений;
- определить состоятельность данного алгоритма, используя различные методы проверки качества полученных последовательностей (на соответствие предполагаемому распределению и на независимость);
- реализовать аналогичные процедуры проверки качества последовательностей, получаемых программным методом (для примера рассмотреть встроенные генераторы python-библиотек);
- выполнить сравнительный анализ показателей качества последовательностей, полученных с помощью программных генераторов, и генератора, использующего квантовые вычисления.

Алгоритм генерации

Алгоритм генерации истинно случайных последовательностей, имеющих распределение, близкое к равномерному, разрабатывался на базе сервиса IBM Quantum Experience [1]. На рис. 1 показано блочное представление алгоритма генерации в виде квантовых вентилей [2], характерное для выбранного сервиса.

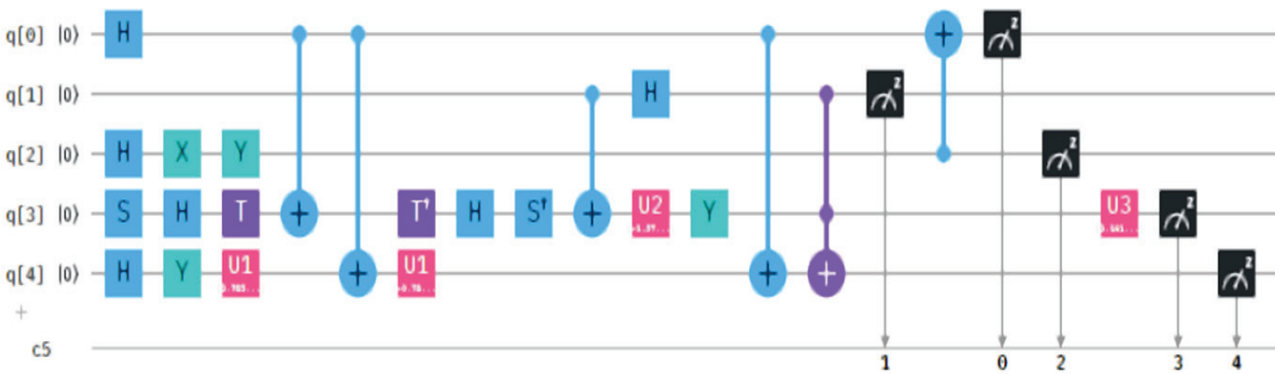

Рис. 1. Схема алгоритма генерации.

Квантовый вентиль с символом «H» называется преобразованием Адамара и представляет собой унитарный оператор, действующий на кубит по правилу:

$$\hat{H}|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$\hat{H}|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

где H – элемент Адамара $\frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, $|0\rangle$ – состояние, описываемое матрицей $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle$ – состояние, описываемое матрицей $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Ещё одним базовым вентилем является отрицание – «NOT». Так как в квантовой механике состояние кубита описывается волновой функцией

$$\psi = \alpha|0\rangle + \beta|1\rangle ,$$

где α, β – комплексные числа, то использование вышеупомянутого вентиля приведет к следующему:

$$NOT\ \psi = \alpha|1\rangle + \beta|0\rangle$$

Немаловажным в представленном алгоритме является вентиль CNOT или контролируемое отрицание. Данный вентиль представлен на рис. 2.

Рис 2. Квантовый вентиль CNOT.

Операции с использованием данного вентиля выглядят следующим образом:

$$\hat{X}|00\rangle = |00\rangle$$

$$\hat{X}|01\rangle = |01\rangle$$

$$\hat{X}|10\rangle = |11\rangle$$

$$\hat{X}|11\rangle = |10\rangle$$

Квантовый оператор CNOT является естественным обобщением классического оператора XOR. Если первый кубит находится в состоянии $|0\rangle$, а второй кубит находится в одном из базовых состояний $|0\rangle$ или $|1\rangle$, то CNOT не изменяет состояния системы, если же первый кубит находится в состоянии $|1\rangle$, а второй кубит в одном из базовых состояний, то под действием оператора CNOT второй кубит перейдет в другое базовое состояние. Матрица оператора CNOT имеет вид:

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Первоначально реализация алгоритма включала в себя лишь 5 кубитов, что на выходе давало лишь 32 различных последовательности бит. Количество возможных генерируемых чисел увеличилось за счет прерывания и повторного наблюдения результатов. Подобный метод возможно реализовать на уровне алгоритма, но не каждый квантовый компьютер поддерживает используемые инструкции. На данный момент единственным ограничением работы данного алгоритма является нехватка кубит (максимальное количество доступных кубитов составляет 15). Данный алгоритм легко модифицируется, что в свою очередь дает возможность реализовать 64-битные последовательности или даже больше.

Проверка состоятельности алгоритма и сравнение с программными генераторами

В рамках данной работы проверка состоятельности алгоритма генерации ограничивалась проверкой распределения получаемой последовательности на соответствие предполагаемому закону распределения.

Реализация оценки качества последовательностей, полученных с помощью квантовых вычислений, выполнена на языке python [3] в среде

разработки spyder. Это решение обосновано наличием у сервиса IBM Quantum Experience открытого API [1], что позволило получать необходимые данные в виде числовых последовательностей и автоматически приводить их к необходимому для проверки виду.

Проверка гипотезы о равномерном на (0, 1) распределении сгенерированных последовательностей выполнялась с помощью критерия Пирсона. Для предварительного оценивания выполнялась визуализация полученных последовательностей (например, рис. 3). Аналогичному исследованию были подвергнуты псевдослучайные последовательности, полученные с помощью встроенного генератора python.

В ходе анализа последовательностей с различным количеством бит было обнаружено, что уже при 16 битах последовательность в 9 из 10 случаев имеет распределение, близкое к равномерному. Исследование проводилось при различной битности чисел и различном количестве чисел в последовательности. Эти результаты обусловлены количеством возможных значений, которые могут принимать генерируемые числа.

Визуализация последовательностей, сгенерированных с помощью квантовых вычислений, и псевдослучайных последовательностей, полученных с помощью встроенного генератора python, показана на рис. 3 и 4 соответственно. Все измерения проводились при генерации 10000 элементов последовательности и разрядности чисел 32 бита.

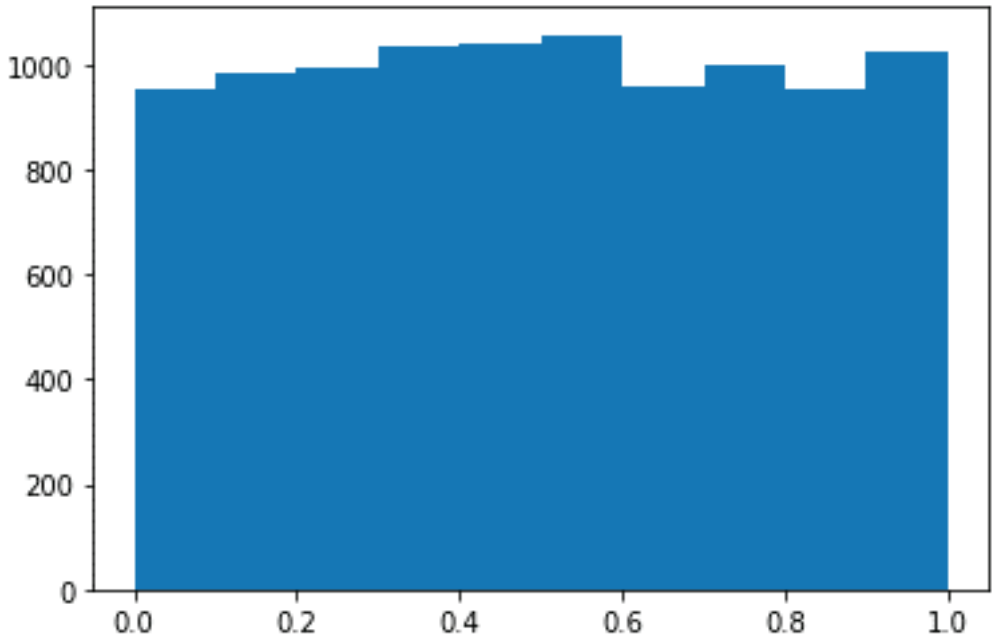

Рис. 3. Гистограмма распределения сгенерированной последовательности (квантовый алгоритм, 32 бита, симуляция).

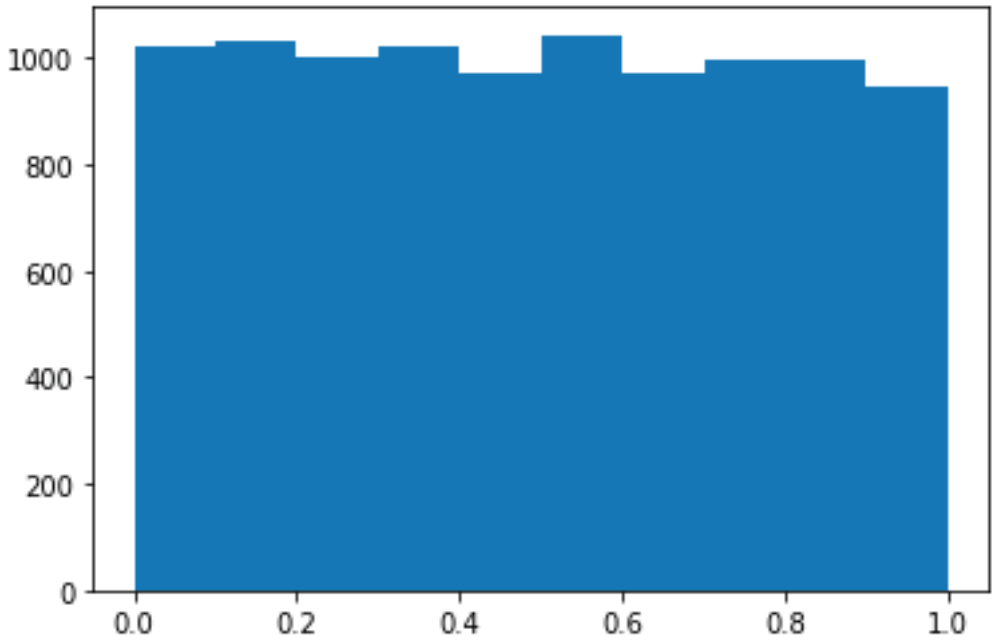

Рис. 4. Гистограмма распределения псевдослучайной последовательности (встроенный генератор python).

Графики показывают, что распределение чисел в обоих случаях близко к равномерному. В таблице 1 представлены числовые характеристики обоих последовательностей.

Таблица 1. Числовые характеристики сгенерированных последовательностей

	Квантовый генератор	Стандартный генератор python (random)
Выборочная средняя	0.5006959574420471	0.4949123960194062
Относительная погрешность выборочной средней по сравнению с теоретическим значением (в долях)	0.0013919148840941808	0.010175207961187649
Теоретическое значение дисперсии на отрезке	0,083(3)	
Дисперсия	0.08198749056954333	0.08319969365546212
Наблюдаемое значение хи-квадрат	13.254	8.192

Полученные результаты показывают, что разработанный алгоритм генерации имеет достаточно высокое качество получаемых последовательностей и сравним с генераторами псевдослучайных последовательностей.

Метод генерации, основанный на использовании квантового компьютера, позволяет избавиться от многих «минусов» других способов генерации. При этом он обладает следующими достоинствами:

- удаленное подключение к квантовому компьютеру и полноценное использование его мощностей;
- сервис IBM (а значит, и процесс генерации) является бесплатным;
- предсказать генерируемые значения невозможно, так как сам квантовый компьютер является «черным ящиком», а кубиты принимают непредсказуемые значения.

Время генерации является довольно спорным аспектом. Следствием

свободного предоставления доступа к квантовым компьютерам является очередь работ для выполнения. Время ожидания заданием своей очереди может быть значительным (по имеющемуся опыту – до 53.5 сек.). При этом сама генерация 10000 элементов требует не больше полсекунды.

Разработанное приложение

В рамках данной работы было разработано приложение для удобного взаимодействия пользователя с API от IBM, а также шифрования файлов. На рис. 5 представлен графический интерфейс приложения, созданный с помощью библиотеки PySimpleGUI [4]. Основные элементы интерфейса:

- Count – количество чисел в генерируемой последовательности;
- Bits – разрядность чисел;
- File – отображение пути к зашифрованному файлу или же к файлу для шифрования;
- Key – отображение пути к файлу-ключу для расшифровки;
- журнал событий (крупное текстовое поле);
- поле для вывода сгенерированной последовательности (текстовое поле в правой части формы);
- кнопка “Submit” – вызов функции генерации последовательности с заданными параметрами.

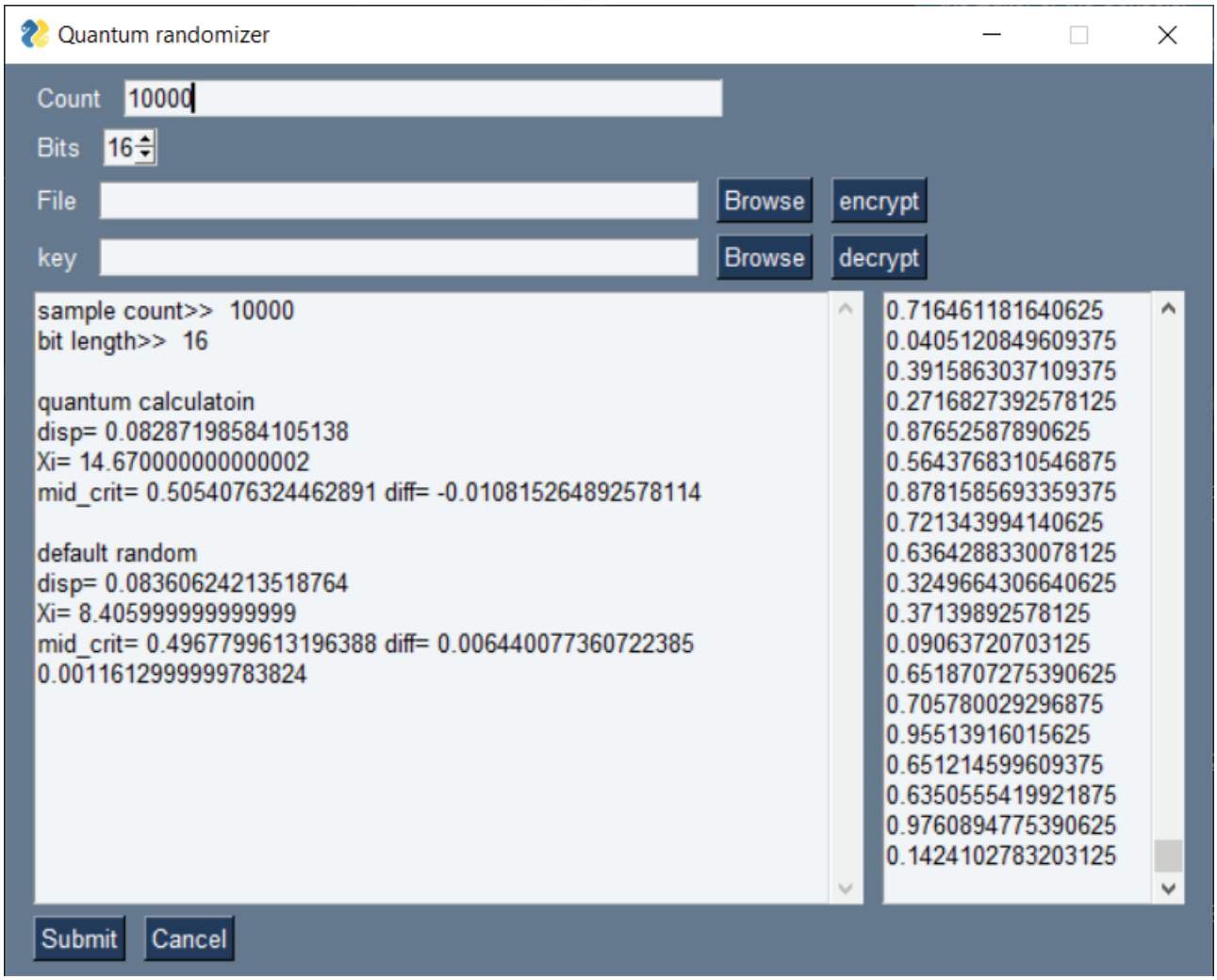

Рис. 4. Интерфейс приложения.

Основной возможностью программы является шифрование данных с помощью алгоритма BlowFish [5], использующего генерируемую последовательность случайных чисел для создания ключа. Данный алгоритм позволяет шифровать данные блоками по 8 байт. Алгоритм способен шифровать файлы разного размера. При шифровании программа создает файлы с расширениями «*.enc» (зашифрованные данные) и «*.key» (ключ). Каждый ключ уникален для каждого зашифрованного файла.

Заключение

В рамках настоящего исследования

- реализован алгоритм генерации последовательностей истинно-случайных чисел;
- выполнен сравнительный анализ результатов работы созданного алгоритма со стандартными генераторами псевдослучайных чисел, показывающий состоятельность рассматриваемого метода генерации;
- реализовано шифрование данных с использованием разработанного алгоритма.

Направления дальнейших исследований:

- доработка пользовательского интерфейса;
- реализация более надежного алгоритма шифрования данных;
- создание web-приложения, доступного пользователям Интернета.

СПИСОК ЛИТЕРАТУРЫ

1. Документация сервиса IBM Quantum Experience: [Электронный ресурс. https://quantum-computing.ibm.com/docs/] (Дата обращения: 26.05.2020)
2. Квантовые вычисления. Учебное пособие / Гайнутдинова А.Ф. – Казань: КГУ, 2009
3. Документация языка python: [Электронный ресурс. https://www.python.org] (Дата обращения: 24.05.2020)
4. Документация PySimpleGUI: [Электронный ресурс. https://pysimplegui.readthedocs.io/en/latest/] (Дата обращения: 26.05.2020)
5. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Шнайер Б. — М.: Триумф, 2002.