УДК 681-3-06 (063)

Ю. В. Щербина[1], к.т.н., **Н. Ф. Казакова**[2], д.т.н., **Р. Г. Братченко**[3], **Г. В. Грабовский**[1]

[1]*Одесская государственная академия технического регулирования и качества, г. Одесса*
[2]*Одесский национальный экономический университет, г. Одесса*
[3]*ТОО «Софтвеа деливери менеджмент», г. Одесса*

ВЫЧИСЛЕНИЕ НА ЯЗЫКЕ C# СПЕЦИАЛЬНЫХ ФУНКЦИЙ *ERF*(*X*) И *ERFC*(*X*) ДЛЯ НУЖД КРИПТОГРАФИЧЕСКОГО АНАЛИЗА

Предложен вариант вычисления на языке C# *таких специальных функций как функция ошибки, неполная функция ошибки и функция нормального распределения, широко применяемые при выполнении задач, связанных с тестированием симметричных криптографических систем. Данный программный модуль является наиболее компактным из созданных в среде Visual Studio. NET и может быть использован разработчиками шифров в составе своего программного обеспечения.*

Ключевые слова*: криптографические системы, криптографический анализ, специальные функции, вычисления.*

Постановка проблемы в общем виде и ее связь с важными научными или практическими заданиями. В настоящее время в области защиты информации наращиваются усилия по созданию эффективных симметричных криптографических систем. В начале века в странах ЕС был открыт проект eSTREAM по выявлению новых поточных шифров, пригодных для широкого применения (Официальный сайт проекта http://www.ecrypt.eu.org/stream/index.html). Этот проект положил начало работе не только по созданию новых перспективных шифров, но и дал толчок дальнейшему развитию научно-методологической базы построения поточных систем шифрования, которые были объявлены одной из целей проекта. Частью этой работы являются средства тестирования предлагаемых способов формирования гаммирующих последовательностей, применяемых в таких шифрах. Основополагающий принцип тестирования впервые был сформулирован Дональдом Кнутом [1]. Он гласит, что псевдослучайная последовательность (ПСП) должна быть испытана как можно большим числом тестов, и удачное завершение каждого из них подтверждает возможность ее применения для нужд шифрования. К настоящему времени создано множество специализированных пакетов, которые призваны решать эту задачу. Наиболее известным из них является национальный стандарт NIST STS (NIST Statistical Test Suite), разработанный в 1999 году [2]. Он включает пакет из 16 независимых тестов и методику обобщения результатов испытаний. На сайте NIST (http://csrc.nist.gov/groups/ST/toolkit/rng/index.html) можно бесплатно получить и руководство к этому пакету, и программный код, написанный на языке C++. Известны и другие пакеты, созданные для этих целей. Например, пакет Diehard [3] разработанный Джорджем Марсальей (George Marsaglia) и включающий 12 тестов доступен по адресу http://stat.fsu.edu/pub/diehard/. Пакет Crypt-X [4] можно получить, связавшись с разработчиками по адресу http://www.isi.qut.edu.au/resources/cryptx/. Подходы и способы тестирования были сформулированы и разработаны достаточно давно. Эта работа активно проводилась всю вторую половину прошлого века и, к 2000 году, было создано и стандартизировано программное обеспечение, позволяющее решать задачи анализа криптографической стойкости симметричных шифров.

Постановка задачи. Идея тестирования предполагает, построение всех тестов, включенных в программу тестирования на основе различных математических вероятностных задач. Это обеспечивает «независимость» результатов тестирования каждым отдельным тестом. При этом приходится вычислять значения специальных трансцендентных функций, значение которых не может быть рассчитано аналитическим путем. В частности, к числу таких функций относят функцию ошибки $erf(x)$ и дополнительную функцию ошибки $erfc(x)$. К счастью, при их вычислении не приходится прибегать к вычислению иных специальных функций, как это, например, имеет место в случае вычисления гамма-функции.

Доступность вычислительной техники в настоящее время несколько упрощает вычисление специальных функций и, более того позволяет это делать с достаточно высокой точностью. Для этой

цели могут с успехом быть использованы такие доступные программные продукты как Matlab, Mathematica, Mathcad, Maple, Statistica, Tabula и многие им подобные. Однако эти универсальные пакеты рассчитаны на чрезвычайно широкий круг математических задач и, поэтому, решения на их основе получаются достаточно громоздкими. Кроме того, разработчики пакетов, как правило, не приводят исходные тексты программ, а предоставляют доступ только к исполняемым модулям, что не всегда отвечает требованиям проблемных программистов.

В реальной жизни разработчики шифров создают свое программное обеспечение, предназначенное для нужд тестирования. При этом сами процедуры тестирования, как правило, предполагают многократное вычисление значений специальных функций. Наиболее рациональным в том случае является не обращение к сторонним пакетам, а наличие специального кода в составе своего программного обеспечения.

Сложность вычисления специальных функций заключается в том, что описывающие их аналитические выражения, как правило, не раскладываются на элементарные функции и, поэтому, их вычисляют, либо, разлагая в ряды, либо представляя их в виде бесконечных дробей. При этом выбор способа в каждом конкретном случае определяется требуемой скоростью и точностью вычислений. В прошлом, было написано достаточно много программ для вычисления этих функций таким способом. Доступными сайтами, где можно получить исходные коды программ для решения подобных задач являются сайты AlgoList (http://algolist.manual.ru/maths/) и AlgLib (http://www.alglib.net/). Эти коды написаны либо на языке C++, либо на Delphi, либо на более ранних версиях иных языков высокого уровня, которые применялись до появления платформы .NET. К настоящему моменту времени существенно обновились и системное обеспечение современных компьютеров, и языки программирования. Сегодня наиболее популярным является новый язык программирования C# от компании Microsoft. Он, как и языки Visual C++ и Visual Basic.NET, входит в пакет Visual Studio.NET, которые являются компонентно-ориентированными языками для новой платформы .NET и которые разрабатывались как альтернатива языку Java. С учетом этого *целью статьи является* описание класса, содержащего программный код, написанный на языке C# для вычисления функции ошибки и дополнительной функции ошибки.

Изложение основного материала исследования.

Обычно в процедурах тестирования функция ошибки $erf(x)$ напрямую не используется. Однако, например в пакете, предлагаемом NIST, в таких тестах как: частотный тест, блочный частотный тест, тест серий, тест на последовательность одинаковых битов, статистический тест Маурера, спектральный тест и дополнительный тест на произвольные отклонения, применяется дополнительная функция ошибки $erfc(x)$. Эту функцию при некоторых значениях аргумента приходится вычислять через значения функции $erf(x)$. Поэтому в программный код модуля они включены обе.

Функция ошибки $erf(x)$ – это неэлементарная функция. Иногда ее называют интегралом вероятности или функцией Крампа [5] и вычисляют как

$$erf\left(x\right)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt\,.$$

Она обладает следующими свойствами

$$erf\left(-\infty\right)=-1,\qquad erf\left(+\infty\right)=1,$$

$$erf\left(-x\right)=-erf\left(x\right),\quad erf\left(x^*\right)=\left[erf\left(x\right)\right]^*,$$

где звездочка означает комплексное сопряжение.

Дополнительная функция ошибки определяется как

$$erfc\left(x\right)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt=1-erf\left(x\right).$$

Кроме того, справедливо соотношение

$$\frac{2}{\sqrt{\pi}}\int_{-\infty}^x e^{-t^2}dt=1+erf\left(x\right).$$

Иногда при выполнении вычислений полезно пользоваться формулой

$$\int_0^x e^{-\frac{t^2}{2\sigma^2}}dt=\sqrt{\frac{\pi}{2}}\cdot\sigma\cdot erf\left(\frac{x}{\sqrt{2}\sigma}\right).$$

Поскольку функция ошибок не может быть представлена через элементарные функции, ее представляют в виде сходящегося ряда путем разложения в ряд Тейлора

$$erf\left(x\right)=\frac{2}{\sqrt{\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{n!(2n+1)}=\frac{2}{\sqrt{\pi}}\left(x-\frac{x^3}{3}+\frac{x^5}{10}-\frac{x^7}{42}...\right).$$

Предлагаемый программный модуль

включает класс EfEfc, состоящий из двух общедоступных процедур: вычисления функции ошибки *erf*(*x*) и дополнительной функции ошибки *erfc*(*x*). Ниже приводится тест самого программного модуля, который может быть включен в проект, созданный в среде Visual C#.

```
using System;
namespace EfEfc
{
 class ErrFnct
 {
 ************************************
*****************************************
**********/
 public double erf (double x)
 {
 double result = 0;
 double xsq = 0;
 double s = 0;
 double p = 0;
 double q = 0;

 s = Math.Sign(x);
 x = Math.Abs(x);
 if ((double)(x) < (double)(0.5))
 {
 xsq = x * x;
 p = 0.00754772803341863128 7834;
 p = 0.28880513720759408492 4010 + xsq * p;
 p = 14.3383842191748205576712 + xsq * p;
 p = 38.0140318123903008244444 + xsq * p;
 p = 3017.82788536507577809226 + xsq * p;
 p = 7404.07142710151470082064 + xsq * p;
 p = 80437.36309608401728321 62 + xsq * p;
 q = 0.0;
 q = 1.00000000000000000000 + xsq * q;
 q = 38.019071395193940375 3468 + xsq * q;
 q = 658.07015545924050632 6937 + xsq * q;
 q = 6379.60017324428279487120 + xsq * q;
 q = 34216.52579246285397690 06 + xsq * q;
 q = 80437.36309608401728262 66 + xsq * q;
 result = s * 1.12837916709551257389615890 31
* x * p / q;
 return result;
 }
 if ((double)(x) >= (double)(10))
 {
 result = s;
 return result;
 }
 result = s * (1 - erfc(x));
 return result;
 }
 ************************************
*****************************************
**********/
 public double erfc(double x)
 {
 double result = 0;
 double p = 0;
 double q = 0;

 if ((double)(x) < (double)(0))
 {
 result = 2 - erf(-x);
 return result;
 }
 if ((double)(x) < (double)(0.5))
 {
 result = 1.0 - erf(x);
 return result;
 }
 if ((double)(x) >= (double)(10))
 {
 result = 0;
 return result;
 }
 p = 0.0;
 p = 0.56418778255073974130870575 63 + x * p;
 p = 9.67580788298726540060420296 1 + x * p;
 p = 77.08161730368428609781633646 + x * p;
 p = 368.51961547100106371338757 46 + x * p;
 p = 1143.26207070388617360607333 8 + x * p;
 p = 2320.43959025163524738476871 1 + x * p;
 p = 2898.0293292167655611275846 + x * p;
 p = 1826.3348842295112592168999 + x * p;
 q = 1.0;
 q = 17.14980943627607849376131193 + x * q;
 q = 137.12559605006222028784435 78 + x * q;
 q = 661.73612071076534692119847 71 + x * q;
 q = 2094.38436778953959379028177 9 + x * q;
 q = 4429.61280388368272671152852 6 + x * q;
 q = 6089.54242327244355046330 68 + x * q;
 q = 4958.82756472114071495438422 + x * q;
 q = 1826.3348842295112595764 38 + x * q;
 result = Math.Exp(-(x * x)) * p / q;
 return result;
 }
 ************************************
*****************************************
**********/
 }
 }
```

При малых значениях *x* функция *erfc* (*x*) вычисляется по правилу *erfc* (*x*) = 1 – *erf* (*x*), при остальных значениях в соответствии с основной процедурой. Аналогично вычисляется и функция *erf* (*x*) [6]. Если этого не делать, программа может возвращать значение *erfc* (*x*) равное нулю, а это не всегда допустимо. Результат

вычисления этой функции должен быть всегда хотя бы и небольшим, но положительным.

Через функции $erf(x)$ и $erfc(x)$ значение функции нормального распределения

$$ndtr(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt = (1 + erf(z)) / 2 erfc\ (z)$$

где $z = x / \sqrt{2}$.

Оно возвращает значение площади под кривой плотности нормального распределения вероятности то $-\infty$ до x. В программном коде это будет иметь вид

$$ndtr(x) = 0.5 * (erf(x / 1.4142135623730950488)) .$$

Описанный модуль может быть включен либо в консольный проект, либо в проект, содержащий форму для вызова результатов выполнения содержащихся в нем процедур. Например, в случае консольного приложения, это будет выглядеть так:

```
static void Main()
{
Console.Write(" x = "); var sx =
Console.ReadLine(); double q = Double.Parse(sx);
ErrFnct fer = new ErrFnct(); double c =
fer.erf(q); // Error function
Console.WriteLine(" Erf(x) = {0}", c);
ErrFnct fec = new ErrFnct(); c = fec.erfc(q); //
Complemented error function
Console.WriteLine(" Erfc(x) = {0}", c);
ErrFnct gsr = new ErrFnct(); // Normal
distribution function
c = 0.5 * (gsr.erf(q / 1.4142135623730950488) + 1);
Console.WriteLine(" Ndtr(x) = {0}", c);
}
```

Выводы

Задача, которую ставили перед собой авторы, сводилась к созданию инструментального средства для расчета некоторых, часто вычисляемых в процессе криптографического анализа, специальных функций. В настоящее время известно множество способов их расчета и программных реализаций этих способов. Здесь приводится наиболее компактный, на наш взгляд, вариант программного кода, созданный в среде Visual Studio.NET. Испытания предлагаемого модуля показали, что результаты вычислений значений функции ошибки и дополнительной функции ошибки не отличаются от результатов, которые дают вычисления с помощью таких пакетов как Matlab, Mathematica, Mathcad, Statistica и Excel.

Список использованных источников

1. Кнут Д. Искусство программирования для ЭВМ. / Д. Кнут. – Т. 2. – М.: Мир, 1977. – 727 с.

2. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. NIST Special Publication 800-22. May 15, 2001.

3. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness//http://www.stat.fsu.edu/pub/diehard/**.**

4. Statistical test suite Crypt-X //http://www.isi.qut.edu.au/resources/cryptx/**.**

5. Попов Б. А., Теслер Г.С. Приближение функций для технических приложений [справочник] / Б. А. Попов, Г. С. Теслер. – Киев: Издательство: Наукова думка, 1984. – 600 с.

6. Абрамовиц М. Справочник по специальным функциям / М. Абрамовиц, И. Стиган. – М.: Наука, 1979. – 486 с.

Поступила в редакцию 25.04.2016

Рецензент: д.т.н., профессор Скачков В. В., Одесская государственная академия технического регулирования и качества, г. Одесса.

Ю. В. Щербина, к.т.н., **Н. Ф. Казакова,** д.т.н., **Р. Г. Братченко, Г. В. Грабовський**

ОБЧИСЛЕННЯ МОВОЮ C # СПЕЦІАЛЬНИХ ФУНКЦІЙ *ERF(X)* ТА *ERFC(X)* ДЛЯ ПОТРЕБ КРИПТОГРАФІЧНОГО АНАЛІЗУ

Запропоновано варіант обчислення на мові C # таких спеціальних функцій як функція помилки, неповна функція помилки і функція нормального розподілу, які широко застосовуються при виконанні завдань, пов'язаних з тестуванням симетричних криптографічних систем. Даний програмний модуль є найбільш компактним з створених в середовищі Visual Stu-dio.NET і може бути використаний розробниками шифрів в складі свого програмного забезпечення.

Ключові слова: *криптографічні системи, криптографічний аналіз, спеціальні функції, обчислення.*

Y. Scherbina, PhD, **N. Kazakova,** DSc, **R. Bratchenko, H. Grabovskii**

CALCULATION IN C # SPECIAL FUNCTIONS *ERF(X)* AND *ERFC(X)* FOR THE NEEDS OF CRYPTANALYSIS

A variant of computation in C # language such special features as a function of an error, an incomplete error function and the function of the normal distribution, commonly used in the performance of tasks related to the testing of symmetric cryptographic systems. This software module is the most compact of the created in Visual Stu-dio.NET environment and can be used by developers of ciphers as part of their software.

Keywords: *cryptographic systems, cryptographic analysis, special functions, calculation.*

УДК 629.05:621.317

О. О. Шелуха

Національний авіаційний університет, м. Київ

ОБРОБКА ІНФОРМАЦІЇ В СИСТЕМАХ СТЕЖЕННЯ ЗА ДИНАМІЧНИМИ ОБ'ЄКТАМИ

В статті розглянуто методику обробку інформації для системи візуального спостереження за динамічними об'єктами. Визначено необхідність створення підсистеми обробки даних рухомої системи спостереження та виділення математичного апарату для визначення положення об'єкта, що постережуться, за даними, що передаються з оптично-електронного модуля. Першим кроком було наведено методику для визначення координат об'єкта спостереження в кадрі переданому з оптично-електронного модуля. Другим кроком було наведено розрахунки координат об'єкта в системі прямокутних координат в горизонтальній площині за допомогою кутових координат. В подальшому пропонується створення комплексної системи обробки даних для побудови тривимірної моделі об'єкта спостереження.

Ключові слова : *система прямокутних координат, рухома система стеження, спостереження динамічних об'єктів.*

Вступ

Сучасні системи обробки і аналізу інформації широко застосовуються в різних областях людської діяльності. Найбільш широке поширення вони отримали в областях навігації, системах стеження, забезпечення безпеки різних об'єктів, передачі та зберігання відеоінформації [1 – 3]. Особливе місце в розкритті проблеми сприйняття, інтерпретації, ідентифікації та опису руху об'єктів займає завдання стеження. Необхідність стеження за динамічними об'єктами і визначення параметрів їх руху пояснюється великою кількістю практичних застосувань, наприклад, при визначенні параметрів руху автотранспорту, при проведенні випробувань для забезпечення безпеки руху повітряних і морських об'єктів, при обробці і реалізації взаємодії цих об'єктів між собою.

При цьому важливим завданням є автоматична реєстрація, відстежування відносного переміщення і визначення параметрів динамічних об'єктів, розташованих в полі зору мобільного пристрою. Рішення даного завдання істотно розрізняються за складністю залежно від виду об'єкта, фону і розташування мобільної оптико-електронної системи.

Сучасний етап розвитку техніки характеризується переважним використанням плоских стаціонарних систем візуалізації зображень [4, 5]. У той же час виникає безліч проблемних питань, пов'язаних з аналізом тривимірних зображень, отриманих від рухомих систем відеоспостереження, які не можуть бути вирішені стаціонарними системами. Плоска проекція не є реальним відображенням дійсності, тому частина інформації про первинне зображення, незважаючи на високу якість, як правило, втрачається. Звідси виникає об'єктивна необхідність у створенні технологій, що будуть виконувати обробку цієї інформації та відновлювати втрачені дані. Підвищення вимог до точності, надійності і мобільності автоматичного виявлення і подальшого супроводу об'єктів змушує шукати нові рішення в сфері розробки алгоритмічного забезпечення для аналізу та обробки відеопослідовностей та застосовувати інформаційні технології нових типів. У зв'язку з цим необхідно вирішити такі завдання у сфері стеження за об'єктами.