ДК 004.054; 004.056.55; 004.421.5

Ю. В. Щербина[1], к.т.н., **С. Л. Волков**[2], к.т.н.

[1]*Одеський національний економічний університет*
[2]*Одеська державна академія технічного регулювання та якості*

ЕЛЕМЕНТИ ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ЧАСТОТНОГО ТЕСТУ ГЕНЕРАТОРІВ КРИПТОГРАФІЧНИХ ПЕРЕТВОРЕНЬ

В статті розглянуті теоретичні принципи частотного тестування псевдовипадкових криптографічних послідовностей та способи їх практичної реалізації.

Ключові слова: *криптографія, криптографічне перетворення, тестування, генератор псевдовипадкових послідовностей.*

Постановка проблеми в загальному вигляді та її зв'язок з науковими і практичними завданнями. Побудова криптостійких систем може бути здійснена шляхом багатократного застосування простих криптографічних перетворень (примітивів). У якості таких примітивів Клод Шеннон запропонував використовувати підстановки (*substitution*) і перестановки (*permutation*). Часто використовуваними криптографічними примітивами є перетворення типу *циклічне зрушення* або *гамування*, тобто метод шифрування, заснований на «накладенні» γ-послідовності на відкритий текст. Як γ-послідовності найчастіше використовують псевдовипадкові послідовності (ПВП), що формуються на основі поліномів.

В даний час, однією з проблем, пов'язаною з захистом даних, що передаються в інформаційно-телекомунікаційних системах, а також з конфіденційністю інформації, є потреба в генераторах криптографічних перетворень, які відповідають високим вимогам до рівномірності розподілу ймовірності формованих ними чисел. Ці вимоги сформовані фахівцями NIST, які в 1999 р. в рамках проекту AES (*Advanced Encryption Standard*) розробили набір статистичних тестів NIST STS (*NIST Statistical Test Suite*) для випробувань ПВП [1].

Також зазначимо, що генератори криптографічних перетворень знаходять застосування в криптографічних протоколах для формування ключів, при хешуванні паролів, а також в алгоритмах, закладених в основу потокових симетричних криптографічних систем, вживаних для захисту конфіденційності передаваної інформації. Побудова таких генераторів – абсолютно нетривіальне завдання і його рішення вимагає копіткої праці математиків та аналітиків. Генератор, слабкий з криптографічної точки зору, може значно ослабити захищеність інформаційної системи і, з цієї причини, для розробників криптографічних систем важливо мати в наявності засіб перевірки їх надійності.

Аналіз наукової і технічної літератури показує, що за останні десятиліття була розроблена і досліджена велика кількість «елементарних» генераторів криптографічних перетворень до яких відносять лінійні конгруентні генератори [2, 3], генератори Фібоначчі з запізнюванням [1], генератори, побудовані на лінійних регістрах зі зворотним зв'язком [3-6] і деякі їх різновиди. Багаторічні дослідження вчених привели до висновку про те, що всі вони не є криптографічно стійкими і можуть входити до складу формувачів криптографічних перетворень тільки як складові елементи.

Як показано в [3], ідея побудови складеного генератора базується на тому факті, що комбінація двух і більше вихідних послідовностей від генераторів різного типу за допомогою таких операцій як «+», «–», «×», «⊕», дозволяє розробити структуру генератора з «кращими властивостями випадковості». Найбільш вдалі складені генератори (з погляду криптографії) детально розглянуті в роботі Б. Шнаєра [7].

З часом в алгоритмах, що здавалися раніше надійними, знаходять нові «слабкі місця». З цієї причини в процесі розробки криптографічних протоколів стає питання про пошук нових інженерних рішень з метою побудови нових, ефективних з точки зору криптографії стійких генераторів, вільних від виявлених недоліків.

Найбільш вдалим на сьогоднішній день формувачем криптографічних перетворень є алгоритм, реалізований в потоковому шифрі RC4 [7]. Однак, все частіше з'являються повідомлення про те, що і в ньому вже знайдені уразливості. Стверджується, що вдалося встановити статистичну залежність характеру ПВП, що генерувалася в потоковому шифрі RC4, від перших символів ключа.

На відміну від інших інженерних завдань,

розробка нового алгоритму формування відрізняється тим, що достовірна відповідь на питання про ефективність знайденого рішення задачі може з'явитися тільки через деякий час, коли для нього буде розроблений індивідуальний метод криптоаналізу. Розробникові залишається задовольнятися тільки результатами попереднього тестування. Про це прямо сказано в керівництві до пакету тестів, розроблених NIST [1]. Будь-який із запропонованих тестів або навіть цілий пакет тестів не замінює криптоаналізу. При цьому попереднє тестування є обов'язковим. Генератор, що не задовольняє умовам тестування непридатний. Кожен з вхідних в пакет тестів орієнтується на пошук певного виду аномалій в потоці формованих символів.

До тестових пакетів, які найбільш рекомендуються до використання, відноситься вже згадуваний пакет NIST STS. Він включає набір з 16-ти тестів і методику їх використання. Успішний результат випробувань проектованого генератора із застосуванням всього набору цих тестів дає підстави сподіватися на те, що формована генератором послідовність невідмітна від «справжньої» випадкової послідовності.

Відомі й інші пакети тестів, створені для потреб криптографії. До них відноситься набір статистичних тестів під назвою Diehard [6], призначений для визначення якості послідовності випадкових чисел. Ці тести були розроблені Дж. Марсальей (George Marsaglia). Він включає 12 тестів і доступний в Інтернеті за адресою: http://stat.fsu.edu/ pub/diehard/.

За адресою http://www.isi.qut.edu.au/resources/cryptx/ можна зв'язатися з розробниками пакету тестів CRYPT-X [2] і отримати програмне забезпечення та керівництво по їх застосуванню.

Проте використання відмічених пакетів прикладних програм натрапляє на ряд серйозних перешкод. Перша з них полягає в тому, що вони призначені для оцінки вже готових генераторів. У практичній роботі такі пристрої розробники конструюють поетапно, поступово доводячи їх до рівня відповідності вимогам, що пред'являються.

Друга проблема полягає в тому, що в основі кожного з вхідних в пропонований пакет тестів лежить достатньо складне теоретичне обґрунтування, що вимагає від розробників серйозної математичної підготовки та знань різних несуміжних розділів математики. На жаль, в керівництві, що додається до тестів, розробники такого обґрунтування, як правило, не приводять.

Нарешті, третя проблема полягає в тому, що, хоча до програмного забезпечення і наданий вільний безкоштовний доступ, скористатися ним складно. Більшість тестів припускають попереднє створення файлу, в який записується випробовувана псевдовипадкова послідовність у вигляді 32-бітових слів, а потім запускається процедура тестування. Це не завжди зручно і підходить не для всіх тестів, оскільки вимагає значних програмно-апаратних ресурсів. До того ж, пропоновані тести розраховані на певну програмно-апаратну платформу.

Перераховані проблеми примушують розробників якщо не розробляти власні тести, то, принаймні, створювати власне програмне забезпечення, яке їх реалізує, є зручним в роботі та може ефективно використовуватись в процесі пошуку конструктивного вирішення генератора, що розробляється.

Всякий пакет тестів має свою внутрішню логіку. Передбачається, що випробування нового генератора повинне починатися з частотного тестування. Як вказується в [1], якщо генератор не проходить частотний тест, то проведення всіх інших тестів вже не має сенсу. Тому, враховуючи все вищесказане, **метою статті** є аналіз виконання частотного тестування і методики її проведення.

Виклад основного матеріалу. Тестування генератора, зокрема частотне, засноване на порівнянні цього генератора з ідеалом. Передбачається, що такий ідеальний генератор формує криптографічну послідовність з рівномірним розподілом ймовірності одиниць і нулів, причому таку, що наступний вихідний біт неможливо передбачити за наслідками спостереження деякого відрізка цієї послідовності з ймовірністю, що відрізняється від 0,5.

Насправді, реальний генератор криптографічних перетворень видає «несправжню» випадкову послідовність, а повністю визначувану значенням секретного ключа. Ступінь його схожості з реальним формувачем випадкової гами може бути встановлена на підставі вибраного еталону та критерію, який дозволяє визначити ступінь відмінності отриманого результату від очікуваного рівномірного розподілу ймовірності.

Формальне визначення критерію припускає завдання нульової гіпотези H_0, відповідно до якої тестована послідовність є випадковою. З нею безпосередньо пов'язана альтернативна гіпотеза H_A відповідно до якої ця послідовність не може бути визнана випадковою. Приймаючи нульову гіпотезу, експериментатор з ймовірністю α, ризикує помилитися – зробити так звану «помилку першого роду». Відповідно, з ймовірністю $1-\alpha$, він буде правим. Зазвичай, величину α вибирають в межах $0,01 < \alpha < 0,001$.

При частотному тестуванні передбачається, що поява символів на виході генератора повністю відповідає розподілу Бернуллі. При цьому

ймовірність одиничного символу p дорівнює ймовірності появи нульового символу – $q = 1 - p$. В цьому випадку різниця між числом одиниць n_1 і числом нулів $n_0, S_n = n_1 - n_0$ в n-розрядній послідовності складає $\varepsilon = \varepsilon_1, \varepsilon_2, ..., \varepsilon_n$, де ε_i – двійковий символ, що приймає значення $\{0,1\}$. Різниця буде підпорядкована біноміальному закону розподілу ймовірності, який відповідно до теореми Муавра-Лапласа, при достатньо великому значенні n, добре апроксимірується стандартним нормальним законом $N_{0,1}$ з нульовим математичним очікуванням і одиничною дисперсією. Як показано в [1, 10], ця різниця, відповідно до центральної граничної теореми, задовольняє умові

$$\lim_{n\to\infty} P\left(\frac{S_n}{\sqrt{n}} < z\right) = \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{u^2}{2}} du,$$

де $\Phi(z)$ – функція Лапласа, яка чисельно дорівнює площі фігури, обмеженої зверху кривою Гауса, знизу віссю абсцис, а справа – прямою $y = z$.

У [1] показано, що для додатних значень z, буде справедливий вираз

$$P\left(\frac{S_n}{\sqrt{n}} \le z\right) = 2\Phi(z) - 1.$$

За даними випробування n-розрядної двійкової послідовності необхідно обчислити статистику

$$s_{obs} = \frac{|n_1 - n_0|}{\sqrt{n}} = \frac{|S_n|}{\sqrt{n}},$$

Значення статистики дозволяє розрахувати ймовірність того, що параметр z не вийде за межі допустимого значення α, визначуваного вибраним критерієм. Ця ймовірність може бути представлена у вигляді

$$2\left[1 - \Phi\left(\frac{|S_{obs}|}{\sqrt{n}}\right)\right] = erfc\left(\frac{|S_{obs}|}{\sqrt{n}}\right). \qquad (1)$$

Враховуючи, що додаткова функція помилки визначається як

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_{z}^{\infty} e^{-u^2} du, \qquad (2)$$

тоді, з урахуванням границь інтегрування, вираз (1) можна переписати у вигляді

$$2\left[1 - \Phi\left(\frac{|S_{obs}|}{\sqrt{2n}}\right)\right] = erfc\left(\frac{|S_{obs}|}{\sqrt{2n}}\right),$$

Якщо, в результаті випробувань обчислена величина $P = erfc\left(\frac{|S_{obs}|}{\sqrt{2n}}\right) \ge \alpha$, то тестована послідовність визнається випадковою.

Якщо величина α вибрана рівною 0,01, то це означає, що зі ста тестованих послідовностей не більше ніж одна з них може бути забракована як «невипадкова».

Програмна реалізація такого тесту має дві складності. Перша з них полягає в тому, що достатньо складно реалізувати ефективний підрахунок числа одиничних n_1 і, відповідно, нульових n_0 символів в тестованій послідовності. Це пояснюється тим, що в більшості сучасних обчислювальних архітектур команд для роботи з окремими бітами немає.

Друга складність полягає в необхідності обчислення в кожному тесті значення додаткової функції помилки $erfc$. В принципі, її значення можна обчислити і за допомогою прикладного програмного пакету MatLab, але в процесі тестування великого числа послідовностей звертатися до окремого програмного пакету не зручно.

Оскільки розмір тестованої послідовності n невеликий (у [1] рекомендується вибирати n не більше, ніж 100 символів), їх кількість в кожному байті може бути підрахована так, як це описано далі.

Зважаючи, що при двійковому численні вага кожного двійкового розряду машинного слова більше суми вагів всіх розрядів, які стоять зліва від нього (молодших по відношенню до цього розряду), значення двійкових символів, що входять до складу цього слова і їх кількість визначається по наступній методиці.

Вважатимемо, що символи a_i, k-розрядного слова $a = \{a_k, a_{k-1}, ..., a_1\}$, що виражає число b, пронумеровані справа наліво (від молодшого розряду до старшого). Тоді значення кожного з них можна розрахувати за правилом

$$a_i = \begin{cases} 1, & \text{при } b - 2^{k-1} \ge 0, \\ 0, & \text{при } b - 2^{k-1} < 0. \end{cases}$$

Обчислення цієї процедури безпосередньо недоцільно, оскільки піднесення до ступеня – операція, трудомістка для обчислювальної системи. Якщо, наприклад, прочитування тестованої послідовності здійснюється побайтно, процедура підрахунку кількості одиничних n_1 і нульових n_0 бітів, написана на мові Delphi, може мати наступний вигляд:

```
b:=a-128;
if b>=0   then   Begin
                   a:=a-128; n1:=n1+1 End
                 else n0:=n0+1;
b:=a-64;
if b>=0   then   Begin
                   a:=a-64; n1:=n1+1 End
                 else n0:=n0+1;
b:=a-32;
if b>=0   then   Begin
                   a:=a-32; n1:=n1+1 End
                 else n0:=n0+1;
b:=a-16;
if b>=0   then    Begin
                   a:=a-16; n1:=n1+1 End
                 else n0:=n0+1;
b:=a-8;
if b>=0   then   Begin
                   a:=a-8; n1:=n1+1 End
                 else n0:=n0+1;
b:=a-4;
if b>=0   then   Begin
                   a:=a-4; n1:=n1+1 End
                 else n0:=n0+1;
b:=a-2;
if b>=0   then   Begin
                   a:=a-2; n1:=n1+1 End
                 else n0:=n0+1;
b:=a-1;
if b>=0   then   Begin
                   a:=a-1; n1:=n1+1 End
                 else n0:=n0+1;
```

Тут лічильники **n1** і **n0** обнуляються на початку тестування і потім накопичують інформацію про кількість відповідних символів до закінчення тестування всієї послідовності. Така процедура виглядає декілька громіздко, проте працює швидко. Вона легко може бути розширена і на випадок блоку більшого розміру.

Алгоритм розглянутої процедури приведено на рис. 1.

Що стосується обчислення показника P, що є додатковою функцією помилки $erfc(x)$ вигляду (2), то її можна представити так:

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

Брати такий інтеграл у вказаних межах незручно, тому краще обчислити функцію $erf\,(x) = \frac{2}{\sqrt{\pi}} \int_0^{x} e^{-t^2} dt$, а потім перейти до функції $erfc(x)$.

Функція $erf\,(x)$ не може бути представлена через елементарні функції. Проте її можна представити у вигляді ряду

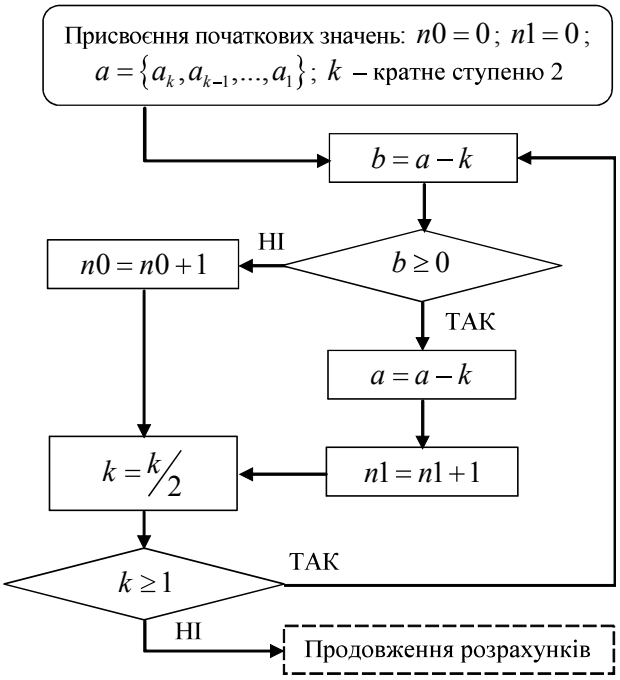

Рисунок 1 – Алгоритм підрахунку кількості одиничних і нульових бітів

$$erf\,(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} =$$
$$\frac{2}{\sqrt{\pi}} \left(x - \frac{x^3}{3} + \frac{x^5}{10} - \frac{x^7}{42} + \frac{x^9}{216} - ... \right).$$

Цим розкладанням можна користуватися, якщо значення аргументу функції $erf\,(x)$ не перевищує значення $x = 3$. При більшому значенні аргументу можна скористатися асимптотичним розкладанням вигляду

$$erfc(x) = \frac{e^{-x^2}}{x\sqrt{\pi}} \left[1 + \sum_{n=1}^{\infty} (-1)^n \frac{1 \times 3 \times 5 \times ... \times (2n-1)}{(2x^2)^n} \right] =$$
$$= \frac{e^{-x^2}}{x\sqrt{\pi}} \sum_{n=0}^{\infty} (-1)^n \frac{(2n)!}{n!(2x)^{2n}}$$

і обчислити значення функції $erfc(x)$ безносередньо. Якщо ряд для визначення значення функції $erf\,(x)$ вимагає обчислення до 30-ти членів, то останній ряд, для обчислення функції $erfc(x)$, дає хороше наближення вже при обчисленні чотирьох членів.

Представлені розкладання для функцій $erf\,(x)$ і $erfc(x)$ можна знайти за адресою http://functions.wolfram.com/GammaBetaErf/.

Висновки

Таким чином, для тестування послідовності, що формується проектованим генератором,

необхідний програмний продукт, в який цей генератор входитиме як складова частина, і який дозволить формувати m n-розрядних послідовностей. Тут m – задане число тестів. При чому цей продукт повинен мати окремий вбудований генератор ключів, з тим, щоб забезпечити їх незалежність. У цьому ж продукті можна розмістити і програму для тестування, яка, з використанням викладених прийомів, дозволить визначати частку послідовностей, що не пройшли тест.

На закінчення відзначимо, що саме такий підхід поетапного тестування і підбору складових елементів генератора дозволяє зробити попередні висновки про прийнятність передбачуваної архітектури проектованого генератора або шифру. Авторами реалізований описуваний підхід до тестування у вигляді двох окремих складових – модуля генератора і модуля тесту. Це дозволяє застосовувати модуль тестування для інших типів генераторів тієї ж розрядності.

Список використаних джерел

1. 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.

2. Кнут, Д. Искусство программирования для ЭВМ [Текст] : монография / Д. Кнут. – М. : Мир, 1977. – 727 с.

3. Харин, Ю. С. Математические и компьютерные основы криптологии [Текст] : учебное пособие / Ю. С. Харин, В. И. Берник, Г. В. Матвеев, С. В. Агиевич. – М. : Новое издание, 2003. – 272 с.

4. Земор, Ж. Курс криптографии [Текст] : монография / Ж. Земор. – Ижевск: НИЦ «Регулярная и хаотическая динамика»; Институт компьютерных исследований, 2006. – 256 с.

5. Рябко, Б.Я. Криптографические методы защиты информации [Текст] : учебное пособие / Б. Я. Рябко, А. Н. Фионов. – М. : МГУ, 2005. – 115 с.

6. Фомичев, В. М. Дискретная математика и криптология [Текст] : курс лекций / В. М. Фомичев // под общ. ред. Н. Д. Подуфалова. – М. : ДИАЛОГ-МИФИ, 2003. – 400 с.

7. Шнайер Б., Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си [Текст] : монография / Б. Шнайер. – М. : Триумф, 2002. – 816 с.

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

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

10. Кац М. Статистическая независимость в теории вероятностей, анализе и теории чисел [Текст] : монография / М. Кац. – М. : Издательство иностранной литературы, 1963. – 156 с.

Надійшла до редакції 18.11.2013

Рецензент: д.т.н., доцент Скопа О. О., Одеський національний економічний університет, м. Одеса.

Ю. В. Щербина, к.т.н., **С. Л. Волков,** к.т.н.

ЭЛЕМЕНТЫ ПРАКТИЧЕСКОЙ РЕАЛИЗАЦИИ ЧАСТОТНОГО ТЕСТА ГЕНЕРАТОРОВ КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

В статье рассмотрены теоретические принципы частотного тестирования псевдослучайных криптографических последовательностей и способы их практической реализации.

Ключевые слова: *криптография, криптографическое преобразование, тестирование, генератор псевдослучайных последовательностей.*

Y. V. Shcherbyna, PhD, **S. L. Volkov,** PhD

ELEMENTS OF PRACTICAL IMPLEMENTATION OF TEST FREQUENCY GENERATORS CRYPTOGRAPHIC TRANSFORMATIONS

In this article the theoretical principles of the frequency of testing of cryptographic pseudo-random sequences and methods of their implementation.

Keywords: *cryptography, cryptographic transformation, testing, generator of pseudocasual sequences.*