文章编号:1671-4598(2024)11-0153-08 DOI:10.16526/j.cnki.11-4762/tp.2024.11.021 中图分类号:TP319 文献标识码:A

基于 DLL 和多线程的随机序列测试软件设计

廖 熹,刘 强,郭元兴,赵 鹏,李建国,何志伟
(中国电子科技集团公司 第 30 研究所,成都 610041)

摘要:随机序列在基于密码学的网络安全算法中被大量应用,在网络安全应用中扮演着重要的角色,随机序列的测试方法也在持续发展,为了提高随机序列的随机性测试效率,开展了随机序列测试软件设计;通过对美国 NIST 公布的 SP 800-22 随机序列测试标准的梳理,在原配套测试集工具包的基础上,随机序列测试软件采用动态链接库和多线程技术,对软件设计架构和测试模式上进行了创新,提高了测试执行效率和人机界面友好性,并通过真随机数序列和伪随机数序列两种方式对其随机性检验效果进行了验证;经实际应用验证,该测试软件设计架构开放、灵活、可配置,测试模式支持多种随机性检验项目的并发执行,测试结果支持量化呈现,提升了测试评估人员的工作效率,在工程实践中具有一定的参考价值。

关键词:信息安全;随机性;显著性水平;伪随机序列;随机性检,验 ;动态链接库;多线程

Design of Random Sequence Testing Software Based on DLL and Multi-threading

LIAO Xi, LIU Qiang, GUO Yuanxing, ZHAO Peng, LI Jianguo, HE Zhiwei
(The 30th Research Institute of CETC, Chengdu 610041, China)

Abstract: Random sequences are widely used in cryptographic-based network security algorithms, it plays an important role in network security applications, the testing methods of random sequences are also developing continuously, in order to improve the testing efficiency of random sequences, a random sequence testing software design is carried out; Through the combing of the SP 800-22 random sequence test standard published by NIST in the United States, based on the original supporting test toolkit, the random sequence test software adopts the dynamic link library and multi-threading technology to innovative the software design architecture and test mode, improves the test execution efficiency and human-machine interface friendliness, the randomness test effect is verified by using true random number sequence and pseudo random number sequence; Through practical application and verification, the test software design architecture has the characteristics of open, flexible and configurable, the test mode supports the concurrent execution of a variety of random testing items, the test results support quantitative presentation, and improve the work efficiency of evaluators, with a certain reference value in engineering practice.

Keywords: information security; randomness; significance level; pseudo random sequence; randomness test; dynamic link library (DLL); multi-threading

0 引言

信息安全在当今社会的重要性是众所周知的。随着计算机和数据通信网络的日益快速发展和广泛应用,社会对其依赖愈发重要。一旦计算机与数据通信网络安全遭受损害,将危及国家安全,引发社会动荡,并造成重大损失[1]。因此,信息安全已成为信息科学技术领域的关注焦点,许多基于加密学的网络安全算法普遍采用随机序列。随机序列在多个网络安全应用中扮演关键角色,常被用作密钥、初始向量或密码协议中的时变参数,在密码学应用中发挥着重要作用[2],不管是密钥生成、数字签名、身份验证还是各种密码学协议,都需要使用随机序列[3]。随机序列的随机性直接影响系统的安全性,对随机序列的随机性进行检验是一项重要的工作,也是国内外研究机构关注的热点之一。

随机性体现了随机事件发生可能性的程度,可以统计其出现的概率[4]。通过对大量重复出现的随机事件进行统计分析,可得出其整体规律,支配着随机性系统的状态。在现实世界中,分为两种类型的随机数:真随机数和伪随机数。真随机数是指由物理噪声完全随机生成的理想随机数,通常利用离子辐射、闸流管或漏电容等实现。而伪随机数则是经由数学算法生成的随机数,由于其算法是确定的,因此所产生的序列并非统计随机,只是在算法设计恰当的情况下,才能够通过随机性检验。

若要在序列密码体制中应用加密序列,则这些序列必须是随机的,具有良好的随机特性。测试序列的随机性关键在于验证其是否真正随机,或者与真随机的差异,通常采用假设检验的方法[5]。随机性假设检验是指,如果已知真随机序列的某一方面符合特定分布,那么假设待检测序列也是随机的,那么在这一方面,待检测序列应该符合特

收稿日期:2024-04-28; 修回日期:2024-09-20。
作者简介:廖 熹(1979-),男,硕士,高级工程师。
引用格式:廖 熹,刘 强,郭元兴,等.基于 DLL 和多线程的随机序列测试软件设计[J].计算机测量与控制,2024,32(11):153-161.

定分布[6]。具体而言，随机性检验是为了确定被测序列中是否存在特定模式，这种特定模式在较长序列中出现的概率应该很小，如果这种模式多次出现，就可以认为被测序列的随机性不够良好。

实际上，尽管密码体制中的序列具有相当长的周期，但更关键的是对序列进行分段统计检验，以验证其随机性。有很多方法可以进行验证，如频数检验、序列检验、游程检验等。通过对序列进行这些统计检验，可以得到验证数据，还需要确定其置信水平，以判断序列是否通过检验。通常将通过标准设定为（1－α）＊100％，其中 α 被称为检验的显著性水平，通常 α 的取值范围为［0.001，0.01］。可以利用 *P-value* 方法来评估待检测序列的随机性。如果 *P-value* 为 1，表示被检测序列是完全随机的；如果 *P-value* 为 0，表示被检测序列是完全非随机的。根据事先设定的显著性水平 α，当 *P-value*$\geqslant\alpha$ 时，被检测序列被视为随机序列；当 P 值$<\alpha$ 时，则被检测序列被视为非随机序列。

1 NIST SP800-22 标准

根据初步统计，目前已公开的随机性检验方法超过 200 种。其中值得一提的有美国国家标准技术协会（NIST）发布的 FIPS140[7-8]、SP800-22[9]、SP800-9B[10] 标准，以及德国联邦信息安全办公室（BSI）发布的 AIS31[11] 标准。国内的一些科研机构也相继公布了一些随机性检验方法和结论[12]。中国的密码行业已经发布了国家标准 GB/T 32915-2016《信息安全技术二元序列随机性检测方法》和 GM/T 0062-2018《密码产品随机数检测要求》。其中 NIST SP800-22 标准是一款代表性的序列随机性检测套件，广泛覆盖了随机序列的统计特性[13]。该标准通过改进检验项和测试结果评判方法[14-19]，可以确保检测套件对被测序列的真实随机性水平进行准确可靠的反映；经过研究检验项之间关系，分析检测套件对序列统计性质的覆盖程度和检验项的相关性[20-22]，导致测试结果存在重叠。

NIST 发布的 SP800-22 标准含 16 种检测序列随机性项目，是最全面和通用的随机数测试标准。16 种检测项目包括频数检验、块内频数检验、游程检验、块内最长游程检验、二值矩阵秩检验、离散傅立叶变换检验、非重叠模板匹配检验、重叠模板匹配检验、Maurer 通用统计检验、Lempel-Ziv 压缩检验、线性复杂度检验、序列检验、近似熵检验、累加和检验、随机偏移检验、随机偏移变化检验[23]，如表 1 所示。

表 1 NIST SP800-22 标准检测项目表

序号	检测项目名称	测试方法描述	备注
1	频数检验	频数检验(又称单比特频数检验)用于检验 0 和 1 符号的分布是否均匀且个数大致相同。	频数检验要求检验序列的比特长度不低于 100。频数检验是随机性检验的基础，也是其他检验项目的前提条件之一。
2	块内频数检验	将待测序列按照 M 比特长度分割成子序列，然后检测每个子序列中符号 1 的比例。当每个子序列中符号 1 的比例达到 $M/2$ 时，则通过了该项随机性检验。	块内频数检验要求被测序列的比特长度不少于 100，并且分块数目不少于 20。
3	游程检验	游程是指一段连续的相同二进制序列，其前后的元素与自身不同。检验被测序列中 0 和 1 的不同长度游程是否符合随机性要求	游程检验要求被测序列的比特长度不少于 100。
4	块内最长游程检验	将待测序列划分成 M 比特长度的子序列，计算每个子序列中最长 1 游程长度，并对其进行统计，若发现大于显著水平 α 的情况，则判定为通过检验。	块内最长游程检验要求待测序列的比特长度不少于 128。
5	二值矩阵秩检验	将被检验的序列分成长为 M 比特的子序列，检测这些子序列之间的线性相关性。在 NIST 中，子矩阵的大小被设定为 32 行 32 列。	二值矩阵秩检验要求被检验序列的比特长度需不少于 38 912，即 38＊子矩阵行数＊子矩阵行数。
6	离散傅立叶变换检验	将待检测序列逐个划分成 M 比特长度的子序列，通过计算子序列的离散傅立叶变换得到的峰值高度，来检测序列的周期特性。	离散傅立叶变换检验要求待检测序列的比特长度不少于 1 000。
7	非重叠模板匹配检验	在被测试序列中验证预先选定的目标串的数量是否符合随机性要求，其计数方法为非重叠计数。在发现目标串时，非重叠模版匹配检验移动目标串长度的比特位。	非重叠模板匹配检验要求被测序列比特长度不低于 10^6。
8	重叠模板匹配项检验	确定被测序列中目标串的个数是否符合随机性要求，计数方法采用重叠计数。与非重叠模板测试的不同之处在于，找到目标串后，重叠模板匹配检验会向后移动一比特位。	重叠模板匹配项检验要求被测序列的比特长度不得少于 10^6。
9	Maurer 通用统计检验	Maurer 通用统计检验发现目标串出现的次数，检测被测序列是否能被显著压缩，同时其信息不损失。被测序列若能被显著压缩，则被认为是不随机的。	Maurer 通用统计检验要求被测序列比特长度不低于 387 840。
10	Lempel-Ziv 压缩检验	Lempel-Ziv 压缩检验检测被测序列中可组成不同单词的子序列数目，以确定被测序列可以被压缩程度。若被测序列可显著压缩，则被视为非随机的。	Lempel-Ziv 压缩检验要求被测序列的比特长度不少于 10^6。

续表

序号	检测项目名称	测试方法描述	备注
11	线性复杂度检验	线性复杂度检验对被检测序列进行线性反馈移位寄存器(LFSR)长度的调整，以确认被检测序列的线性复杂度是否足够。	线性复杂度检验要求被检测序列的比特长度介于 500 至 5 000 之间。
12	序列检验	序列检验检测整个被测序列中出现长度为 m 比特的重叠子序列的次数，确定其是否符合随机序列的要求。	序列检验要求被检测的比特长度不得少于 10^6。
13	近似熵检验	近似熵检验检测长度为 m 比特的重叠子序列在整个被测序列中的出现次数，以及检测被测序列中相邻长度为 m 比特和 $m+1$ 比特的重叠子序列的发现频率。	近似熵检验要求被测序列的比特长度不得少于 10^6。
14	累加和检验	累积和检验采用累积和的方法来探测序列的最大偏移量。它确定被测序列中部分序列的正向和反向累积和的偏移量，相对于随机序列的要求有无过大或过小。	累积和检验要求被测序列的比特长度不少于 100。
15	随机偏移检验	随机偏移检验在每轮测试中，检测被测序列累加和的 8 个特定状态(−4，−3，−2，−1，+1，+2，+3，+4)出现次数，以判断是否符合随机性要求。	随机偏移检验要求被测序列的比特长度不得少于 10^6。
16	随机偏移变化检验	随机偏移变化检验检测被测序列的累加和的 18 个特定状态(−9，−8，…，−1，+1，+2，…，+9)出现总次数，以判断是否符合随机性要求。	随机偏移变化检验要求被测序列的比特长度不少于 10^6。

2010 年，NIST 对 SP800-22 标准版本进行了更新，版本变更为 speccial Publication 800-22 Revision 1a[24]。和上一版相比，新版本减少了 Lempel-Ziv 压缩检验的测试项[25]，同时配套下发了新的测试集工具包 sts-2 _ 1 _ 2.zip，该工具包运行在 linux 操作系统下，提供了可执行程序 assess，用于 SP800-22 Revision 1a 标准规定的 15 项随机序列随机性测试，并将最终的测试结果输出到对应的文件中。其测试项设置运行效果如图 1 所示。

```
                    S T A T I S T I C A L    T E S T S

[01]Frequency                                   [02]Block Frequency
[03]Cumulative Sums                             [04]Runs
[05]Longest Run of Ones                         [06]Rank
[07]Discrete Fourier Transform                  [08]Nonperiodic Template Matchings
[09]Overlapping Template Matchings              [10]Universal Statistical
[11]Approximate Entropy                         [12]Random Excursions
[13]Random Excursions Variant                   [14]Serial
[15]Linear Complexity

        INSTRUCTIONS
           Enter 0 if you DO NOT want to apply all of the
        statistical tests to each sequence and 1 if you DO.

Enter Choice:1
```

图 1　NIST 配套测试集工具包运行效果图

该测试集工具包通过提供命令行操作的方式为用户提供随机序列随机性测试，当执行随机性检验项时，是通过串行方式逐项进行的检测，导致测试集工具包执行效率较低，当检测数据达到 GB 级时，在标准计算机上完成一次完整的随机性检测可能需要数小时[26]，而且测试过程中的人机界面交互也较简单。故需在原测试集工具包的基础上，设计随机序列测试软件，改善随机序列随机性测试的测试模式和人机界面，提升随机性测试的工作效率。

2　随机序列测试软件的设计

2.1　软件总体设计

根据 NIST SP800-22 Revision 1a 标准的随机性测试需求，在 NIST 测试集工具包 sts-2 _ 1 _ 2 的基础上，从软件交互界面的友好性、测试接口调用的规范性、测试项目的并发测试效率、测试结果管理的便捷性等方面，进行了随机序列测试软件的设计。该测试软件主要由测试控制模块、数据解析模块、测试结果显示模块、测试报表管理模块等组成，其组成原理框图如图 2 所示。

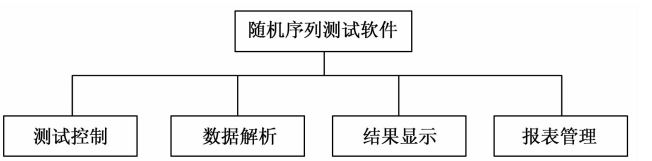

图 2　软件组成原理框图

测试控制模块完成被测随机序列文件、测试参数及测试项目的设置；数据解析模块完成 15 项随机性检验的具体实现；测试结果显示模块将检验结果直观的呈现给用户；测试报表管理模块根据用户自定义要求，完成报表的相关操作。随机序列测试软件运行效果如图 3 所示。

图 3　随机序列测试软件运行效果图

用户运行该测试软件，对测试项目、测试数据、测试参数完成设置后，可进行对应的测试项目执行，完成测试结果的收集，测试结果包括测试项目内容及结果、测试时间等，同时可为用户提供报表生成、报表存储、报表导出、

报表打印等服务。下面对各软件模块设计进行阐述，因数据解析模块是该软件的核心部件，故对数据解析模块的设计进行了详细说明。

2.2 测试控制模块设计

随机序列测试软件的测试控制模块，主要采用 BCB6 提供的界面框架组件、输入输出控件、对话框控件来完成测试控制界面人机交互的设计，确保使用者通过测试控制界面输入测试指令，从而控制随机序列测试软件完成对应测试任务。确保用户能够识别界面信息，理解交互含义，让使用者通过简易的界面操作，保证各测试指令的正确执行。测试参数输入便捷，测试方式支持单项测试和组合测试，提升了测试配置的灵活性和方便性，其运行效果如图 3 所示。

2.3 数据解析模块设计

2.3.1 数据解析模块的测试解析组件设计

数据解析模块在 NIST 官网提供的随机数测试软件工具包的基础上，采用动态链接库 DLL 的方式，对 15 种随机性检验方法的 C 语言源文件进行了接口函数封装，形成测试解析组件，增强测试接口调用的规范性，可供 windows 平台的 C＋＋开发工具进行调用。使用 BCB6 的 IDE 开发工具，可以方便的进行 DLL 的编写，DLL 的内容编写和一般的程序并无本质的区别，开发人员完全可以在 DLL 中编写各种函数、使用变量和对象。数据解析模块主要包含了 15 个测试解析组件，负责完成对应随机性检验项目的执行，其定义如表 2 所示。

表 2 随机性检验项目测试解析组件列表

序号	测试解析组件名称	功能描述	函数接口定义
1	Frequency. dll	频数检验测试项目执行	Frequency (char * filename, int filetype, int block, int bitofblock, double * p_value)
2	BlockFrequency. dll	块内频数检验测试项目执行	BlockFrequency (char * filename,int filetype,int block, int bitofblock, int N, double * p_value)
3	Runs. dll	游程检验测试项目执行	Runs (char * filename, int filetype,int block, int bitofblock,double * p_value)
4	LongestRunOfOnes. dll	块内最长游程检验测试项目执行	LRO(char * filename,,int filetype,int block, int bitofblock, double * p_value)
5	Rank. dll	二值矩阵秩检验测试项目执行	Rank (char * filename, int filetype, int block, int bitofblock,double * p_value)
6	DFT. dll	离散傅立叶变换检验测试项目执行	DFT (char * filename, int filetype, int block, int bitofblock,double * p_value)
7	NOTM. dll	非重叠模板匹配检验测试项目执行	NOTM (char * filename, int filetype, int block, int bitofblock, int lengthoftemplate, double p_value[300][256])
8	OTM. dll	重叠模板匹配项检验测试项目执行	OTM (char * filename, int filetype, int block,int bitofblock,int lentp, double * p_value)
9	Universal. dll	Maurer 通用统计检验测试项目执行	Universal (char * filename, int filetype, int block, int bitofblock, int L, int Q, double * p_value)
10	LinearComplexity. dll	线性复杂度检验测试项目执行	LinearComplexity (char * filename, int filetype, int block, int bitofblock, int M, double * p_value)
11	Serial. dll	序列检验测试项目执行	Serial (char * filename, int filetype, int block, int bitofblock,int m, double * p_value1,double * p_value2)
12	ApproximateEntropy. dll	近似熵检验测试项目执行	ApproximateEntropy (char * filename, int filetype, int block, int bitofblock,int m,double * p_value)
13	CumulativeSums. dll	累加和检验测试项目执行	Csums(char * filename, int filetype, int block, int bitofblock,double * p_value)
14	RandomExcursion. dll	随机偏移检验测试项目执行	RandomExcursion (char * filename, int filetype, int block, int bitofblock,double p_value[256][256])
15	RandomExcursions-Variant. dll	随机偏移变化检验测试项目执行	RandomExcursionsV(char * filename, int filetype, int block, int bitofblock,double p_value[256][256])

DLL 编写的关键在于 DLL 接口部分，即提供给其他程序使用的函数、类和数据说明。在 DLL 接口部分，推荐使用 _ _ declspec 关键字来进行修饰，它使用较为灵活，可以出现在函数、类和数据声明中的任意位置，推荐采用 _ _ stdcall 关键字来明确调用的参数是固定的，调用者严格按照定义传递传输。在 DLL 程序的 DLL 接口中，使用 _ _ declspec（dllexport）来输出声明；在调用 DLL 的程序，则使用 _ _ declspec（dllimport）来输入声明。

以频数检验 DLL 为例，其接口函数的定义如下：

extern " C" _declspec（dllexport） _ _stdcall int Frequency（char * filename，int filetype，int block，int bitofblock，double * p _ value）；

前 4 个参数为输入参数，filename 是待测试的随机序列数据的路径及文件名，filetype 是该文件的格式，block 是序列分段数目，bitofblock 是每段被测数据长度，最后一个参数为每段被测数据返回的该项测试的显著性水平值结果数组。完成频数检验 DLL 编写后，进行编译即可获得 Frequency. dll。调用该 DLL 时应在主程序中使用相符的函数名称，这也是需要使用 _ _stdcall 关键字的原因。DLL 的调用方式有两种，包括明确连接和隐含连接。

确定连接是通过 LoadLibrary 函数加载 DLL，并使用 FreeLibrary 函数释放 DLL。这种方法是设计人员主动加载该 DLL，然后使用 GetProcAddress 获取函数地址，再调用该函数，并在 DLL 不再使用时，使用 FreeLibrary 释放它。使用明确连接的好处在于，可以充分掌握该 DLL 的载入与释放，最大程度地利用系统资源，只是在使用时稍显繁琐。调用 DLL 时使用明确连接，需要调用 WIN32API 函数 LoadLibrary、GetProcAddress、FreeLibrary。

隐含连接利用输出函数库对应的 DLL 函数库来实现调用函数，因此载入和释放 DLL 的过程是隐形的。一旦程序加载该输出函数库，系统会自动加载相应的 DLL；程序结束后，系统会立即释放该 DLL。采用隐含连接能够使开发人员在不必考虑 DLL 加载和释放问题的情况下调用 DLL，使用起来就像使用静态库连接那样便捷。使用隐含连接时，需将 DLL 对应的 .lib 文件添加到 BCB6 的项目中，通常也会包括 DLL 的 .h 头文件来声明函数，之后就能像调用常规函数那样使用 DLL 中的函数。

以频数检验 DLL 为例，随机序列测试软件以明确连接方式调用 Frequency. dll。主要步骤包括通过 LoadLibrary 函数加载 Frequency. dll，随后使用 GetProcAddress 获取频数检验函数 Frequency 的地址，然后调用 Frequency 函数对随机序列进行频数检验，获取频数检验结果，最后使用 FreeLibrary 函数完成 Frequency. dll 的资源释放。

2.3.2　数据解析模块的并行测试流程设计

数据解析服务的处理流程：数据解析模块根据用户的测试需求，启动测试线程，创建被测随机序列的数据副本，加载检验所需的测试解析组件，解析组件对被测数据格式进行符合性检查，判断其是否满足该组件执行随机性检验的测试要求，如满足，则解析组件对被测数据进行对应的随机性检验，完成检验后，返回检验结果，并释放资源，关闭测试线程，完成对应的随机性检验执行。数据解析服务提供测试项目随机性检验执行的控制流程图如图 4 所示。

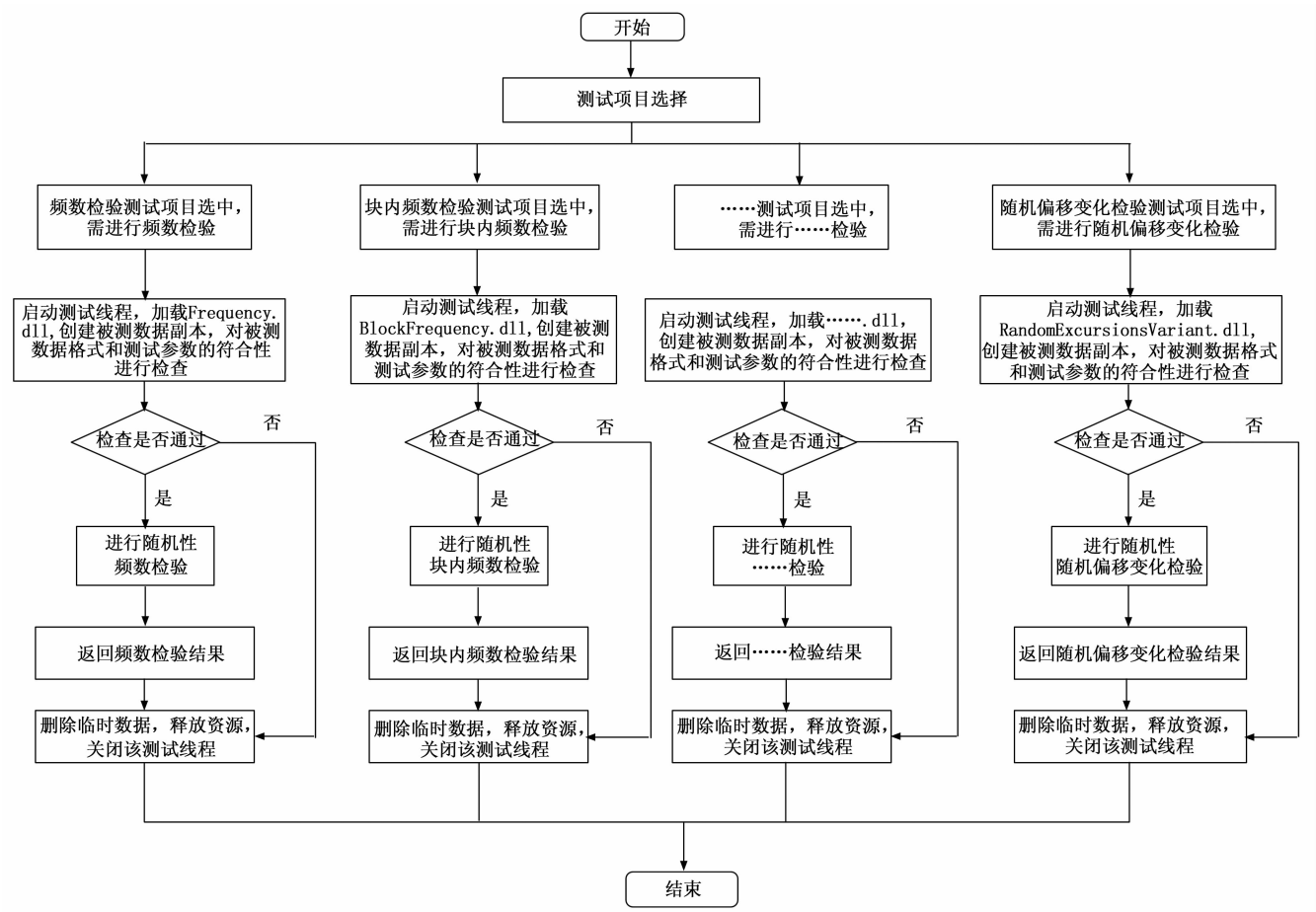

图 4　数据解析服务的控制流程图

为提高数据解析服务的检验效率，随机序列测试软件采用多线程方式[27]来进行实现不同随机性检验测试项目的并行测试。BCB6 提供了线程类（TThread）来封装和线程相关的函数调用，它是作为所有线程类的基类来使用，本身并不能直接使用。开发人员必须定义一个从 TThread 类派生出的线程类，然后创建该线程类的对象实例，才能使用多线程技术。

多线程的应用一般分为以下步骤：1）从 TThread 类派生出一个新的线程类；2）创建线程对象；3）设置线程对象的属性；4）根据需要，立即执行、挂起或唤醒线程；5）结束线程。在随机序列测试软件中，首先通过 IDE 提供的线程向导生成新的测试线程类 TestThread，然后根据项目需求，在其 .h 头文件中对声明进行修改，主要代码示例如表 3 所示。

表 3 测试线程类 TestThread 说明表

序号	TestThread 定义代码
1	class TestThread : public TThread
2	{
3	private:
4	protected:
5	void __fastcall Execute ();
6	…….
7	public:
8	__fastcall TestThread (int TestID);
9	int RandomTestFun (int TestID, char * inFileName, char * outFileName);
10	int CurrentTestID;
11	char RNDFileName [256];
12	char ResultFileName [256];
13	…….
14	};

然后在其 CPP 文件中对构造函数及 Execute 方法中添加代码，主要代码示例如表 4、表 5 所示。

表 4 测试线程类 TestThread 构造函数说明表

序号	TestThread 构造函数定义代码
1	__fastcall TestThread:: TestThread (int TestID) : TThread (true)
2	{
3	CurrentTestID = TestID;
4	……
5	};

表 5 测试线程类 TestThread 的 Execute 方法说明表

序号	TestThread 的 Execute 方法代码
1	void __fastcall TestThread:: Execute ()
2	{
3	RandomTestFun (CurrentTestID, RNDFileName, ResultFileName);
4	};

接着在使用线程的 CPP 文件中创建该测试线程对象，并执行线程，主要代码示例如表 6 所示。

表 6 测试线程对象创建及执行说明表

序号	测试线程对象创建及执行代码
1	for (int i=0; i<15; i++)
2	{
3	if (true == TestState [i])
4	{
5	MyCurrentThread [i] = new TestThread (Test _ ID); //启动对应项目的测试线程
6	MyCurrentThread [i] ->FreeOnTerminate = true; //线程执行完成后，自动释放
7	MyCurrentThread [i] ->Resume (); //唤醒线程
8	}
9	}

通过上述方法，随机序列测试软件运用多线程技术，实现多个随机性检验项的并发测试，很好地提升了测试效率。

2.4 结果显示模块设计

随机序列测试软件的结果显示模块，通过两种方式呈现随机性检验结果，方便测试评估人员从不同维度对随机性检验结果进行观察。方式一是对本次执行的全部随机性检验项目的检验结果进行展示，如图 5 所示。方式二是针对具体的随机性检验项目的检验结果进行展示，如图 6 所示。

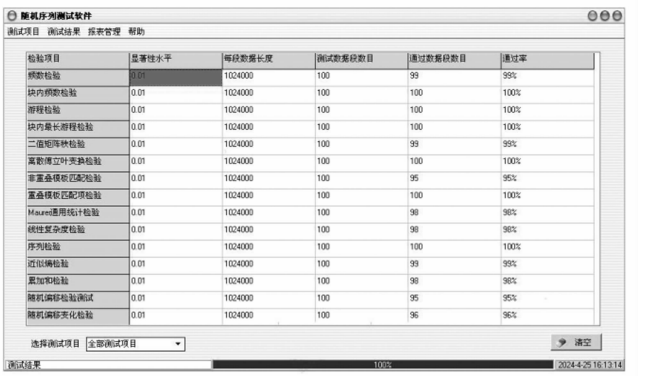

图 5 全部测试项目检验结果展示效果图

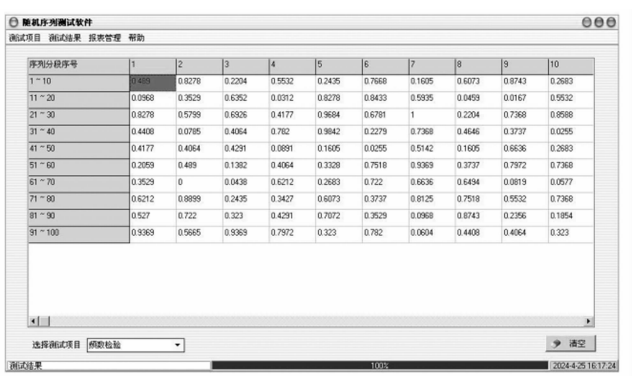

图 6 单个测试项目检验结果展示效果图

2.5　报表管理模块设计

随机序列测试软件的报表管理模块，通过 OLE 自动化技术同 word 软件进行交互，调用预先制定的测试报表 Word 文档模板，采用书签定位的方式，建立随机序列测试软件测试结果数据与测试报表 Word 文档的参数映射关系，实现文字、表格等数据信息在测试报表 Word 中的生成，支持测试评估人员快捷生成本次随机性检验的测试报表。

3　随机序列测试软件的验证

使用的测试平台为 i5 酷睿处理器，频率为 3.3 GHZ，内存为 4 GB DDR3，频率 1 600 MHz，因 NIST 测试集工具包不能在 windows 平台直接运行，通过在 windows 平台上运行 Cygwin 终端，在 Cygwin 终端完成 NIST 测试集工具包的编译，生成 NIST 测试集工具包的 assess. exe 可执行程序。分别运行随机序列测试软件和 NIST 测试集工具包 assess. exe，通过真随机数和伪随机数两种方式对 NIST SP800-22 Revision 1a 标准的 15 项随机性检测方法进行对比验证。

1）伪随机数序列测试：

首先要生成测试样本，伪随机数序列通过 31 级线性反馈移位寄存器 LFSR 产生被测文件，其本原多项式为：$f(x) = x^{31} + x^{17} + x^{10} + x^{5} + 1$，实现原理图如图 7 所示。

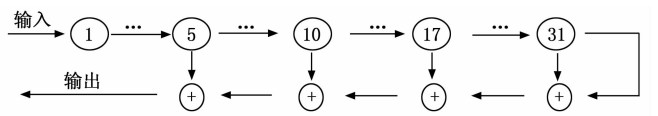

图 7　31 级线性反馈移位寄存器原理图

输入随机的 31 比特“1100 0001 0101 0010 1001 0010 0010 110”作为初态，进入 31 级线性反馈移位寄存器，产生 102 400 000 比特的数据，保存为 PRND. dat 文件。运行随机序列测试软件，选择 PRND. dat 文件，将被测序列设置为 100 段，每段长度为 1 024 000 比特，显著性水平为 0.01，进行全部 15 项随机性检验项，测试耗时为 112 633 ms，其测试结果如表 7 所示。

由表 7 可知，通过 31 级线性反馈移位寄存器产生的随机数序列，采用随机序列测试软件通过了 13 项随机性检验，但是二值矩阵秩检验和线性复杂度检验的两项随机性检验，未能达到 NIST SP800-22 Revision 1a 标准的要求。

运行 NIST 测试集工具包进行该伪随机数序列测试样本的 15 项随机性检验，得到的结果和表 7 是一致的，测试耗时为 542 463 ms。

2）真随机数序列测试：

采集某噪声源芯片的输出数据作为被测随机数序列，采集的噪声源数据长度也为 102 400 000 比特，保存为 RND. dat 文件，运行随机序列测试软件，选择 RND. dat 文件，将被测序列设置为 100 段，每段长度为 1 024 000 比特，显著性水平为 0.01，进行全部 15 项随机性检验项，测试耗时为 113 016 ms，其测试结果如表 8 所示。

表 7　伪随机数序列测试结果表

测试项	显著性水平	每段数据长度	测试数据段数目	通过数据段数目	通过率/%
频数检验	0.01	1 024 000	100	100	100
块内频数检验	0.01	1 024 000	100	100	100
游程检验	0.01	1 024 000	100	90	90
块内最长游程检验	0.01	1 024 000	100	100	100
二值矩阵秩检验	0.01	1 024 000	100	0	0
离散傅立叶变换检验	0.01	1 024 000	100	100	100
非重叠模板匹配检验	0.01	1 024 000	100	99	99
重叠模板匹配检验	0.01	1 024 000	100	99	99
Maurer 通用统计检验	0.01	1 024 000	100	98	98
线性复杂度检验	0.01	1 024 000	100	0	0
序列检验	0.01	1 024 000	100	100	100
近似熵检验	0.01	1 024 000	100	99	99
累加和检验	0.01	1 024 000	100	100	100
随机偏移检验	0.01	1 024 000	100	92	92
随机偏移变化检验	0.01	1 024 000	100	90	90

表 8　真随机数序列测试结果表

测试项	显著性水平	每段数据长度	测试数据段数目	通过数据段数目	通过率/%
频数检验	0.01	1 024 000	100	99	99
块内频数检验	0.01	1 024 000	100	100	100
游程检验	0.01	1 024 000	100	100	100
块内最长游程检验	0.01	1 024 000	100	100	100
二值矩阵秩检验	0.01	1 024 000	100	99	99
离散傅立叶变换检验	0.01	1 024 000	100	100	100
非重叠模板匹配检验	0.01	1 024 000	100	95	95
重叠模板匹配检验	0.01	1 024 000	100	100	100
Maurer 通用统计检验	0.01	1 024 000	100	98	98
线性复杂度检验	0.01	1 024 000	100	98	98
序列检验	0.01	1 024 000	100	100	100
近似熵检验	0.01	1 024 000	100	99	99
累加和检验	0.01	1 024 000	100	98	98
随机偏移检验	0.01	1 024 000	100	95	95
随机偏移变化检验	0.01	1 024 000	100	96	96

由表 8 可知，该随机数序列通过了全部 15 项随机性检验。因此，通过该噪声源芯片生成的随机数序列，其随机性达到 NIST SP800-22 Revision 1a 标准的要求。

运行 NIST 测试集工具包进行该真随机数序列测试样本的 15 项随机性检验，得到的结果和表 8 是一致的，测试耗时为 543 678 ms。

3）测试验证结果：

通过比较随机序列测试软件和 NIST 测试集工具包的测试结果和测试耗时可知，随机序列测试软件和 NIST 测试集工具包的随机性测试标准是一致的，同时随机序列测试软件测试效率提升明显，约为 NIST 测试集工具包的 5 倍。下

一步准备对每项随机性检验项的具体算法进行优化，进一步提升随机性检验项的测试效率。

4 结束语

本文通过对 NIST800-22 标准的研究，在原配套测试集工具包的基础上，采用动态链接库和多线程技术，设计了随机序列测试软件，并通过真随机数和伪随机数两种方式对其随机性检验效果进行了测试，该测试软件设计架构开放、灵活、可配置，可对随机数质量进行自动化量化计算，并以直观方式显示测试结果，增强了随机性检验的测试执行效率和人机界面友好性，提升了测试评估人员的工作效率，已在实际工作中得到应用，对于随机序列的随机性检验具有一定的参考价值。

参考文献：

[1] STALLINGS W. 密码编码学与网络安全：原理与实践 [M]. 孟庆树，王丽娜，傅建明，等译（4 版）. 北京：电子工业出版社，2010.

[2] 陈 爽，曹素梅，左金印. 随机数发生器检测与设计 [J]. 信息安全与通信保密，2012 (12)：103.

[3] 刘 康. 伪随机序列随机性测试系统的设计与实现 [D]. 石家庄：石家庄铁道大学，2016.

[4] 赵海英. 随机数发生器随机性检测系统的设计与实现 [D]. 成都：电子科技大学，2014.

[5] 师国栋，康 绯，顾海文. 随机性测试的研究与实现 [J]. 计算机工程，2009，35 (20)：145－147.

[6] 张永强，李顺波，屈 帅，等. NIST 随机性检测方法及应用 [J]. 电脑知识与技术，2014，10 (26)：6064－6066.

[7] National Institute of Standards and Technology. Federal information processing standards publication fips pub 140-1 [S]. Gaithersburg：National Institute of Standards and Technology，1994.

[8] National Institute of Standards and Technology. Federal information processing standards publication fips pub 140-2 [S]. Gaithersburg：National Institute of Standards and Technology，2001.

[9] RUKHIN A，SOTO J，NECHVATAL J. A statistical test suite for random and pseudorandom number generators for cryptographic applications [S]. NIST Special Publication 800－22，2013.

[10] BARKER E B，KELSEY J M. Recommendation for Random Number Generation Using Deterministic Random Bit Generators [M]. NIST Special Publication 800-90A，2012.

[11] KILLMANN W，SCHINDLER W. A proposal for：Functionality classes for random number generators [S]. Bonn：Bundesamt fur Sicherheit in der Informationstechnik，2011.

[12] 亓民勇，董金新，潘全科. 信息安全中序列随机性测试系统的研究与设计 [J]，计算机工程与设计，2008，29 (6)：1454.

[13] 马 原，陈天宇，吴鑫莹，等. 随机数发生器的设计与检测 [J]. 信息安全研究. 2019，5 (1)：39－49.

[14] KIM S J，UMENO K，HASEGAWA A. Corrections of the NIST statistical test suite for randomness [OL]. 2004 [2015-09-16]. https://arxiv.org/abs/nlin/0401040.

[15] HAMANO K. The distribution of the spectrum for the discrete Fourier transform test included in SP800-22 [J]. IEICE Transactions on Fundamentals of Electronics，Communications and Computer Sciences，2005，1 (88)：67－73.

[16] H AMANO K，KANEKO T. Correction of overlapping template matching test included in NIST randomness test suite [J]. IEICE Transactions on Fundamentals of Electronics，communications and Computer Sciences，2007，90 (9)：1788－1792.

[17] SULAK F，DOGANAKSOY A，EGE B，et al. Evaluation of randomness test results for short sequences [C] //In：Sequences and Their Applications SETA 2010. Springer Berlin Heidelberg，2010：309－319.

[18] PARESCHI F，ROVATTI R，SETTI G. Second-level NIST randomness tests for improving test reliability [C] //In：IEEE International Symposium on Circuits and Systems-ISCAS 2007：1437－1440.

[19] PARESCHI F，ROVATTI R，SETTI G. On statistical Tests for Randomness included in the NIST SP800-22 test suite and based on the Binomial Distribution [J]. IEEE Transactions on Information Forensics and Security，2012，7 (2)：491－505.

[20] TURAN M S，DOGANAKSOY A，BOZTAS S. On independence and Sensitivity of statistical randomness tests [C] //In：Sequences and Their Applications SETA 2008. Springer Berlin Heidelberg，2008：18－29.

[21] 范丽敏，冯登国，陈 华. 基于熵的随机性检测相关性研究 [J]. 软件学报，2009，20 (7)：1967－1976.

[22] SULAK F，UGUZ M，KOCAK O，et al. On the independence of statistical randomness tests included in the NIST test suite [J]. Turkish Journal of Electrical Engineering & Computer Sciences，2017，25 (5)：3673－3683.

[23] NIST. Special publication 800-22，a statistical test suite for random and pseudorandom number generators for cryptographic applications [Z]. 2001.

[24] NIST. Special publication 800-22 revision 1a，a statistical test suite for random and pseudorandom number generators for cryptographic applications [Z]. 2010.

[25] 段俊红，韩炼冰，房利国，等. 基于 SP 800-22 标准的随机性测试方法研究与快速实现 [J]. 通信技术，2018，51 (8)：1940－1944.

[26] PARESCHI F，ROVATTI R，SETTI G，et al. On statistical tests for randomness included in the NIST SP800-22 test suite and based on the binomial Distribution [J]. IEEE Transactions on Information Forensics and Security，2012，7 (2)：491－505.

[27] 王天澳，于 洵，朱 镭，等. 微小型无人机载光电吊舱显控软件的设计与实现 [J]. 计算机测量与控制，2022，30 (3)：145－150.