目　　录

1. 总体结构与实现目标

1.1 总体结构

QRNG实现框图如图1所示，包括熵源与采样部分和后处理与检测部分。本方案将设计并实现量子随机数发生器的后处理与检测部分。

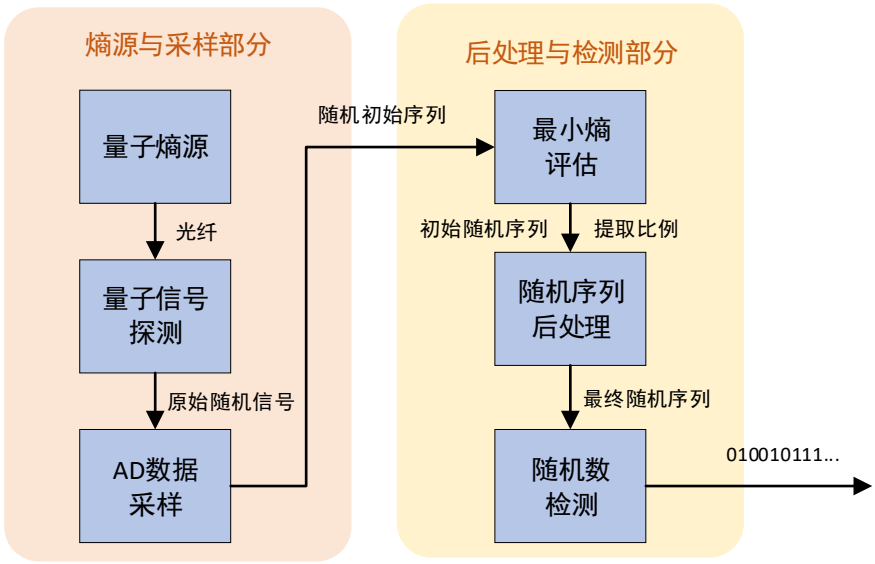

图1 QRNG实现框图

1.2 后处理与检测

理想情况下，QRNG可产生信息论可证明的安全随机数。但实际上，真随机信号的熵源——量子噪声，不可避免地会与经典噪声混合在一起，受到经典噪声的影响。从保密通信的安全性的角度来看，攻击者Eve可以控制经典噪声并获得有关原始随机数的部分信息。例如，假设QRNG系统装置使用的是外部电源，而可能存在的Eve可以控制外部电源波动以控制经典噪声，即使此时QRNG的设备是可信任的，Eve仍可以获得随机数里的部分信息。因此，有必要应用安全的后处理来防止Eve窃听，提取免受第三方攻击的安全随机序列。

后提取的作用是对初始随机序列进行数据后处理，保证满足统计均匀的随机序列的输出对原始随机序列进行数据后处理操作，即随机性提取过程，以获得独立、均匀、可证安全的最终随机序列，提取出量子真随机性。

随机性检验通常通过概率统计的方法考察被检测的序列是否满足随机序列

的某些特征以判定其是否随机。这些测试手段主要致力于判定可能存在于序列中的各种非随机性。其中检验统计量用于计算总结针对零假设的证据强度的 $P-value$ 值。

1.3 实现目标

本方案将接收AD采集卡传来的原始随机序列，设计并实现最小熵评估以获得随机序列提取比例；使用信息论安全的后处理算法进行随机序列的提取，去除经典噪声，获得完美的随机性；最后使用NIST-STS方案对提取后的随机序列进行随机性验证，最后输出可靠的最终随机数序列。

对于具体部分的实现方案将在下文详细介绍。

2. 最小熵评估

2.1 简介

熵是刻画不确定性的度量，反映了对实验结果预测和观察前后消除的不确定性的消除，即物理过程蕴含的不确定性——熵的量越大，蕴含的不确定性就越大。量子随机数产生过程中边信息是必然存在的，我们作边信息被窃听方完全掌控的最严苛假设，以量子最小熵给出系统的量子噪声熵含量下限，用以评估量子随机数发生器的随机性品质，给出后处理中量子随机数的提取比例[1]。

2.2 最小熵计算

最小熵是评价原始随机数随机性的常用指标，最小熵的物理意义是原始随机数中随机性的比例的下界，即每n比特的样本中包含$H_\infty(X)$个随机比特，所以计算最小熵就能够知道最多能从原始随机数中提取多少真随机数了，这为提取器提供了重要参数[1]。

$$H_\infty\left(X\right)=-log_2(max_{x\in\{0,1\}^n}P(X=x)) \tag{1}$$

除了直接用频率估算概率的方法，美国国家标准研究所发布的NIST SP 800-90B还给出了多种评估熵源最小熵的通用方案，分别为Most common Value检测、碰撞估熵、马尔可夫估熵、压缩估熵、t元组估熵、最长重复子串估熵、Multi Most Common in Window Prediction 估熵、滞后预测估熵、MultiMMC Prediction估熵、LZ78Y Prediction估熵。完成最小熵估计后，即可根据最小熵的估计结果计算提取比例[2]。

2.3 方案目标

本方案将采用频率估计概率的方法计算最小熵，这种方法的优势在于计算速度快，时间复杂度仅为$O(n)$，可以减少在熵评估部分花费的时间，以达到更高的随机数产生速率。

根据NIST提供的最小熵方案编写的开源代码（https://github.com/dj-on-github/SP800_90b_tests）将作为本方案的最小熵实现方案。

3. 熵源健康检测

3.1 简介

熵源健康检测通过判断量子熵源特性是否符合预期的统计特性，识别量子熵源是否处于异常状。熵源健康检测应检测量子熵源输出的原始随机序列，并在量子熵源运行过程中持续或周期性执行。执行熵源健康检测时不应导致量子熵源输出被抑制或输出速率被减低。若熵源健康检测结果为失败时，应告警并可以关闭量子随机数输出[3]。

3.2 方案目标

本方案使用的熵源健康检测的方法包括重复计数测试和适配比例测试（参考文献[1]所推荐的熵源健康检测方法）。这两种方法分别用于检测噪声源长时间地重复输出某一个数的极端异常状态和物理器件损坏或者环境变化导致的大量熵损失。

4. 随机数后处理

4.1 简介

在量子随机数产生过程中，真随机提取是必须的环节。一方面，提供量子随机数熵源的量子不确定性变量统计上并不服从均匀分布，如真空量子态分量起伏服从高斯分布，因此必须对原始随机数进行后处理。另一方面，量子探测过程中会尽量将经典噪声降至最低，但是光路及电路中的非理想因素还是会不可避免的引入经典噪声，造成边信息的必然存在。为了祛除边信息的影响，提取器需要依据熵源**量子噪声熵含量**的准确评估，采用安全性信息论可证的随机提取器从原始随机数中提取出源于量子熵的均匀分布的量子随机数。

因此在量子随机数的产生过程中，后处理程序是必不可少的。后处理的作用对象是直接采集得到的原始随机数，然后从中提取出量子真随机数。随机数后处理的过程一般分为两个部分：一是量子随机性的评估和随机性的提取。量子随机性的评估可以为原始随机数的提取提供合适的提取比例，二是量子随机性的评估方法必须是科学的，随机性提取的方法也要得到信息论证明的[4]。

4.2 Toeplitz-hash算法

后处理算法主要包括异或（XOR）处理方法[5]、m-LSB方法[6]、von Neumann方法[7]、基于Toeplitz矩阵的哈希方法[8]等。由于基于Toeplitz矩阵的哈希处理方法具有结构简单、易于构建、信息论可证安全性等特点，因此本方案采用将Toeplitz-hash方法作为后处理算法。

Toeplitz-hash函数目前广泛用于密码学领域，它的本质是一种压缩映射，又被称为单向散列函数。Toeplitz-hash是经过信息论证明的随机性提取算法，它可以将任意长度的输入序列，通过类似矩阵运算的方法变换成固定长度的序列进行输出，即$Y = H(X)$。

哈希函数具有下面三个特性。第一个是随机性，输出Y统计上服从均匀分布；二是单向性，只能进行单向的计算，反过来计算是不可行的；三是唯一性，很难找到不同的X_1和X_2，使得他们的运算结果$H(X_1)$和$H(X_2)$完全相同。

接下来简要概述Toeplitz-hash方法的原理。下式为一个$m \times n$的Toeplitz矩阵：

$$T_{m\times n}=\begin{bmatrix} V_m & V_{m+1} & \cdots & V_{m+n-1} \\ V_{m-1} & V_m & \cdots & V_{m+n-2} \\ \vdots & \vdots & \cdots & \vdots \\ V_1 & V_2 & \cdots & V_n \end{bmatrix} \tag{2}$$

可以看到Toeplitz矩阵的对角元素相等，首行和首列的元素由真随机序列产生，其余元素都可以由首行和首列元素移位循环产生。Toeplitz矩阵的构成和循环矩阵类似，首行元素向右平移一位得到第二行元素，但首个元素是由新的随机数补充得到，并舍弃最后一个元素，以此类推。一个$m \times n$阶的Toeplitz矩阵的生成只需要$m+n-1$个随机比特即其首行和首列元素，从而极大减少计算过程中所需的计算及空间资源。

假设初始随机序列长度为N，其最小熵为H_{min}，则得到的最终随机序列长度为$M = N * H_{min}$，需要构建的Toeplitz矩阵大小为$N \times M$，则用Toeplitz矩阵实现QRNG后处理的原理框图如图2所示。

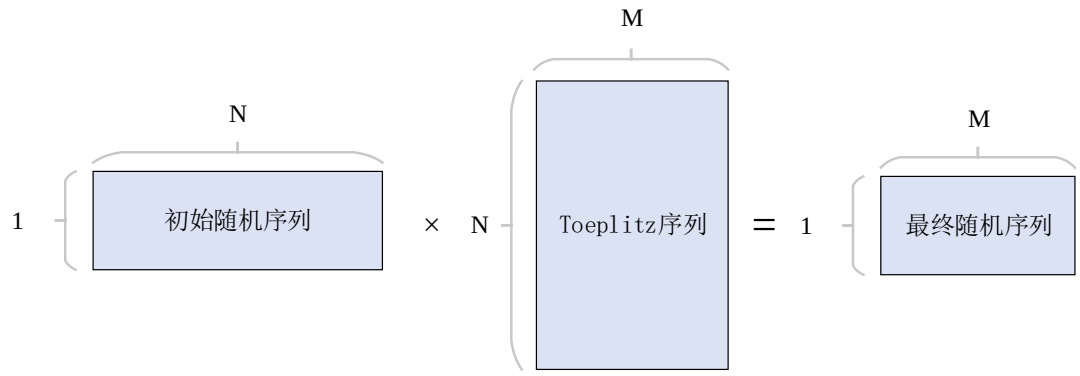

图2 Toeplitz实现原理图

图2描述的实现方式是初始随机序列取行向量，得到的最终随机序列也是行向量。当初始随机序列取列向量时，为$N \times M$的Toeplitz矩阵与$1 \times N$的初始随机序列相乘，得到$1 \times M$的最终随机序列。不论是哪种实现方式，构建Toeplitz矩阵时都仅需$N+M-1$个随机数作为种子，相比$N \times M$，大幅度减少了后处理过程中需要使用的随机数的数量，同时可以节约计算和存储资源，提高后处理的实现效率[9]。

4.3 使用FFT进行算法加速

基于Toeplitz矩阵的哈希实现本质是矩阵相乘，大小为$1 \times N$的原始数据经过大小为$N \times M$的Toeplitz矩阵哈希后，得到大小为$1 \times M$的最终数据。当N、M较大时，直接进行矩阵相乘需要很大的存储空间，同时对计算能力要求也比较高，且计算复杂度高，影响计算性能。

鉴于Toeplitz矩阵结构的特殊性，即可以用第一行和第一列元素表示整个矩阵，将矩阵相乘转为多项式相乘，并采用FFT加速算法，将计算复杂度从$O(n^2)$降为$O(nlog_2 n)$[10]。

该方法支持实现对长输入比特序列的处理，其实现方式可以归纳为：

（1）将大小为$j \times k$的Toeplitz矩阵拓展为大小为$(j+k-1) \times (j+k-1)$的循环矩阵；同时，通过在原始随机比特序列后补充$k-1$比特的0元素，从而实现将长度为j的原始随机比特序列向量拓展为长度为$j+k-1$的输入原始随机比特序列。图3以3×4的Toeplitz矩阵为例，介绍了循环矩阵的构造方法。

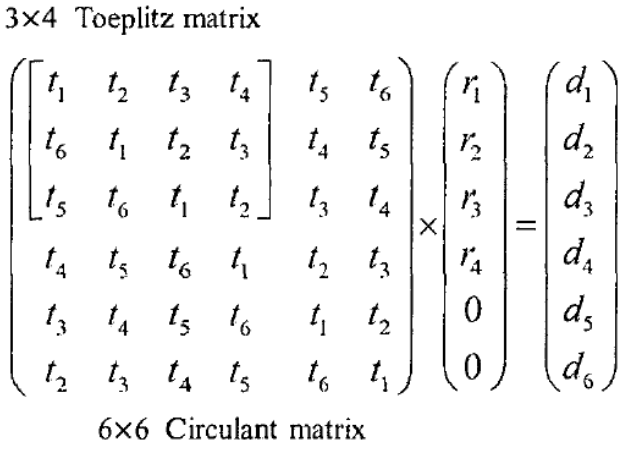

图3 3×4的Toeplitz循环矩阵构造方法

（2）利用快速傅里叶变换加速循环矩阵与拓展后的序列乘法的运算，实现将计算复杂度降低为$O(nlog_2 n)$。循环矩阵与拓展后的向量相乘输出的前j比特即为该随机提取器的输出结果。

基于Toeplitz矩阵的哈希实现过程中，可将矩阵相乘转为多项式相乘，同时采用FFT算法进行加速，具有线性计算复杂度，减少计算存储资源的同时，提高整个操作的处理效率，进而得到随机性提取操作的高性能实现。

4.4 流水线结构加速

在Toeplitz算法中，当N和M的值较大时，无论是矩阵相乘还是FFT加速算法都将耗费极大的计算资源，同时降低计算效率，影响性能。考虑到资源空间有限，在输入的初始随机序列长度较大时，将其按顺序拆分为长度较小的子序列，同时将Toeplitz矩阵按规律划分为与之对应大小的子矩阵，子序列与子矩阵之间独立执行后处理操作。如图4所示，为了充分利用计算资源，在各个子模块间采用流水线结构，具体的流程可描述如下：

（1） 将初始随机序列和Toeplitz矩阵按规律对应划分为较小的子序列和子矩阵，分别得到x个batch；

（2） 每个子序列和子矩阵对应的batch独立执行后处理操作，并采用FFT算法进行加速；

（3）若每个batch间采取顺序执行，即上一个子模块执行完FFT、复数相乘、IFFT后第二个子模块再执行相应操作，将造成大量延时及资源的浪费；很明显内存资源不足以支撑子模块间的并行执行。因此采用流水线模式，在第一个batch执行完FFT并继续执行复数相乘的同时，第二个batch开始执行FFT操作，以此类推，保证从第三个batch开始每一时刻在同时执行FFT、复数相乘及IFFT三种操作，直到结束。

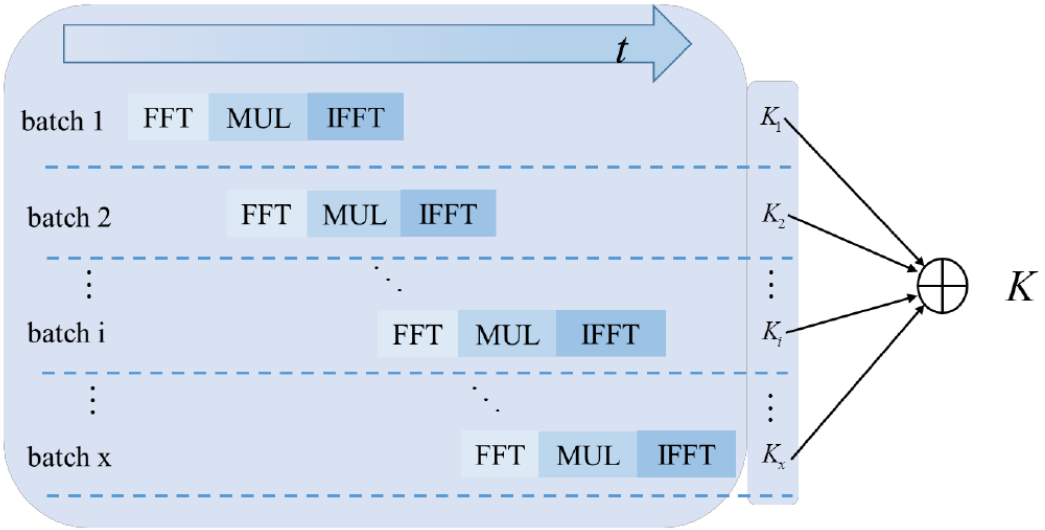

图4 流水线结构加速FFT示意图

流水线工作模式可以充分利用内存资源，提高内存的利用率，进而提高运行效率。其关键点在于子序列及子矩阵大小的选择，一方面要保证有合适的内

存来满足同时执行FFT、复数相乘及IFFT三种操作，内存过大同样将造成资源的浪费；另一方面要选择合适大小的流水线，流水线数量过小时难以发挥其优势，过大时实际运行中存在的延时总和会造成一笔不小的时间开支，不利于高速实现后处理的初衷。二者的选取缺一不可，需要同时考虑初始随机序列的长度大小、使用的计算机的内存资源以及FFT、复数相乘、IFFT三种操作所需要的资源等各种因素，进而选取合适的参数大小来实现后处理的高效实现[11]。

4.5 基于GPU的随机性提取

基于Toeplitz矩阵的哈希过程的实现目前主要可以通过硬件（如FPGA）和软件(如CPU、GPU)两种方式。相比较而言，硬件易于实现，具有较高时效性，用线性反馈移位寄存器来构造Toeplitz矩阵简单方便，发展也比较迅速，但受硬件资源限制，可处理的码长相对较短；且FPGA更适合定点数运算，而基于Toeplitz矩阵的哈希中有复数相乘等多种浮点数计算。

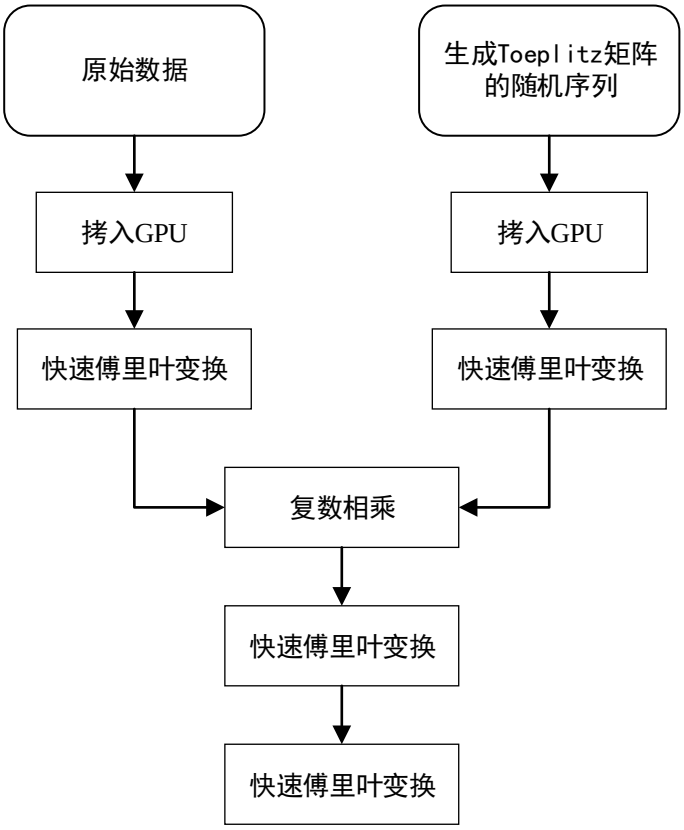

图5 GPU平台上使用FFT加速的基于Toeplitz矩阵后提取流程图

对于高速实时的QRNG系统而言，可以选择GPU实现基于Toeplitz矩阵的哈希，主要有两个原因：一方面可以采用FFT算法加速，另一方面由于GPU自身

结构的特殊性，可以并行处理大量数据，极大简化了计算过程，从而实现数据的高速处理。NVIDIA公司生产的GPU提供了并行计算架构CUDA，可以并行进行大量数据的操作，带有FFT的库函数，因此可在GPU上采用FFT进行加速，实现大量数据的高性能计算[11]。

使用GPU平台，FFT加速的基于Toeplitz矩阵后提取流程图如图5所示，可总结如下：

（1）将原始数据和用来生成Toeplitz矩阵的随机序列分别从主机拷贝到GPU；

（2）将拷入的两组数据进行处理后分别作正向FFT；

（3）两组数据分别进行快速傅里叶变换后的结果进行复数相乘；

（4）对上一步复数相乘的结果作逆向FFT；

（5）IFFT的结果进行处理后拷回到CPU。

4.6 方案目标

本方案计划在GPU（或CPU）平台上，采用FFT加速的基于Toeplitz矩阵后提取方法，并使用流水线结构进行多线程运行优化，预计提取速率上限可达到2Gbps。

5. 随机数检测

5.1 简介

随机数发生器产生的随机数在应用之前，需要有一定的方法进行检验其随机性。由于真随机性的基础是具有无限长的序列，无法用数学上的统计检测完成，所以真正意义上的真随机性的检测方法目前并不存在。随机性检验通常通过概率统计的方法考察被检测的序列是否满足随机序列的某些特征以判定其是否随机。目前国际上的随机性测试方法有多种，包括：ENT[12]、Diehard、NIST-STS[13]等。下面给出ENT和NIST-STS的随机性检测方法的说明。

5.2 ENT检测

ENT检测主要包括以下四个统计量的测量：

（1）香农熵：如果一个n位随机序列X满足 $H(x)=n$，说明该序列具有最大的不确定性；

（2）π值(Monte Carlo模拟中的π)：将随机序列用于Monte Carlo模拟中计算圆周率π，结果越接近真实值则随机序列的随机性越好；

（3）平均值：计算随机序列的算术平均值，无偏置的随机序列的均值应该等于0.5，满足均匀分布；

（4）序列相关系数：真随机序列任意k皆相关系数皆为零。

5.3 NIST-STS检测

美国国家标准与技术研究院(National Institute of Standards and Technology，NIST)发布了一篇名为"A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications"的文章[13]，对用于检验加密应用的随机数发生器的质量NIST SP 800-22随机数测试进行了全面的介绍。该标准用于对密码应用中的随机数或伪随机数发生器产生的二进制随机数序列进行统计测试。

5.4 NIST-STS随机数测试集

NIST SP 800-22随机数测试集包含15个测试项，用于测试基于硬件或软件的随机数或为随机数生成器生成的(任意长)二进制序列的随机性。每一项测试分别侧重于该序列中可能存在的不同类型的非随机性，其中部分测试可以分解为不同的子测试。

（1）频数检验(The Frequency Test)

该检验方法测试的重点在序列中0和1的比例。这个测试的目的是确定一个序列中1和0的数量是否与一个真正的随机序列的期望大致相同，也就是说，在整个序列中1和0的数量应该大致相同。所有后续的检验手段都在该检验成立的基础上进行。

（2）块内频数检验(Frequency Test within a Block)

检验主要是看M位的子块中"1”的比例。该检验的目的是判定M位的子块内“1”的频率是否像随机假设下预测的那样，近似于M/2。对于块大小M=1的情况，此测试退化为则频数检验。

（3）游程检验(The Runs Test)

检验主要是看游程的总数，游程J旨的是一个无间断的相同数序列。一个长度为k的游程则包含k个相同的位，并且前后都有一个相反的值。运行测试的目的是确定不同长度的"1”游程的数目以及“0”游程的数目是否跟理想的随机序列的期望值一致，判断在这样的“0”“1”子块之间的振荡是否太快或太慢。

（4）块内最长游程检验(Test for the Longest-Run-of-Ones in a Block)

该检验主要是看M-bits的子块中最长的“1”游程，目的是判定待测序列中最长“1”游程的长度是否与随机序列相同。

（5）二元矩阵秩检验(The Binary Matrix Rank Test)

该测试的重点是整个序列的不相交子矩阵的等级。此测试的目的是检查原始序列的固定长度子串之间的线性相关性。

（6）离散傅里叶变换检验(The Discrete Fourier Transform Test)

该检验主要是看对序列进行分布傅里叶变换后的峰值高度，目的是探测待检验信号的周期性，以此揭示其与相应的随机信号之间的偏差程度。

（7）非重叠模块匹配检验(The Non-overlapping Template Matching Test)

该检验主要是看提前设置好的目标数据串发生的次数，为了探测那些产生太多给出的非周期性模式的发生器。

（8）重叠模块匹配检验(The Overlapping Template Matching Test)

该检验主要是看提前设定的目标模块发生的数目。

（9）Maurer的通用统计检验(Maurer's "Universal Statistical" Test)

该检验主要是看匹配模块间的bit数，为了检验序列能否在没有信息损耗的条件下被大大的压缩。一个能被大大压缩的序列被认为是一个非随机序列。

（10）线性复杂度检验(The Linear Complexity Test)

该检验手段主要是看线性反馈移位寄存器的长度。检验的目的是判定序列的复杂程度是否达到可视 为是随机序列的程度。

（11）序列检验(The Serial Test)

该检验主要是看整个序列中所有可能的重叠m-bit模式的频率，目的是判定2m个m-bit重叠模式的数目是否跟随机情况下预期的值相近似。随机序列具有均匀性也就是说对于每个m-bit模式其出现的概率应该是一样的。当m=1时等价于频数检验。

（12）近似熵检验(The Approximate Entropy Test)

该检验目的是看是整个序列中所有可能的重叠m-bit模式的频率。目的是将两相邻长度(m和m+1)的重叠子块的频数与随机情况下预期的频数相比较。

（13）累加和检验(The Cumulative Sums (Cusums) Test)

该检验主要是看随机游动的最大偏移。随机游动被定义为序列中调整后的1，+1的累加和。检验的目的是判定序列的累加和相对于预期的累加和过大还是过小。这个累加和可被看做随机游动。对于随机序列，随机游动的偏离应该在0附近。对于非随机序列，这个随机游动偏离将会比0大很多。

（14）随机游动检验(The Random Excursions Test)

该检验主要是看一个累加和随机游动中具有K个节点的循环的个数，目的是确定在一个循环内的特殊状态对应的节点数与在随机序列中预计达到的节点数相背离。

（15）随机游动状态频数检验(The Random Excursions Variant Test)

该检验主要是看累计和随机游动中经历的特殊状态的总数，目的是判定随机游动中实际经历多个状 态的值与预期值之间的偏离程度。

5.5 NIST参数建议

设 n 为样本序列的比特长度，NIST 并未定义，但倾向于取 1000000。

检测项	NIST 参数建议
单比特频数检验	$n>100$
块内频数检验	$n>100$，块长 $M>0.1n$
游程检验	$n>100$
块内最长游程检验	当 128<6272 时，M=9 当 $6272<n<750000$，M=128 当 $n\geq 750000$，M=10000
二元矩阵秩检验	$n\geq 38MQ$，行列 $M=Q=32$
离散傅里叶变换检验	$n\geq 1000$
非重叠模块匹配检验	模版长度通常取 9 或 10（更有意义） 块数 $N\leq 100$ 块大小 $M>0.01n$
重叠模块匹配检验	$n>1000000$，建议 $m=9,10$
Maurer 的通用统计检验	$n\geq (Q+K)L \quad 6\leq L\leq 16$ $Q=10*2L$
线性复杂度检验	$n\geq 1000000$ 块长 $500\leq M\leq 5000$ 块数 $N>200$
序列检验	$m<\lfloor \log_2 n \rfloor -2$
近似熵检验	$m<\lfloor \log_2 n \rfloor -2$
累加和检验	$n>100$
随机游动检验	$n\geq 1000000$
随机游动状态频数检验	$n\geq 1000000$

5.6 方案目标

本方案的随机数检测将以NIST-STS测试为主，ENT、Diehard测试辅助，代码使用NIST测试套件[14]，技术指标参考5.5节。NIST测试套件是由NIST提供的15个测试组成的统计软件包，用于对密码应用中的随机数或伪随机数发生器产生的二进制随机数序列进行统计测试，需要在Linux系统下运行。当对输入的随机数序列或为随机数序列完成测试之后，NIST随机数测试集工具包会生成相应的测试报告文件，包括相关的中间值，如每个统计测试的P-value值。基于这些值，该测试套件会自动生成关于该序列的随机数质量结论。

本部分规定的随机性检测采用的显著性水平均为$\alpha = 0.01$。则当$P-value \geq \alpha$时，测试序列通过该项随机数统计检验，否则测试序列不通过随机数统计检验。

6. 参考文献

[1]成琛. 连续变量量子随机数发生器熵含量及产生速率提升的研究[D].太原理工大学,2020.DOI:10.27352/d.cnki.gylgu.2020.000920.

[2]NIST. Information Technology Laboratory [EB/OL]. 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

[3]YD/T 3907.3—2021，基于BB84协议的量子密钥分发（QKD）用关键器件和模块第3部分：量子随机数发生器（QRNG）[S].

[4]Impagliazzo R, Levin L A, Luby M. Pseudo-random generation from one-way functions[C]//Proceedings of the twenty-first annual ACM symposium on Theory of computing. 1989: 12-24.

[5]Qi B, Chi Y M, Lo H K, et al. High-speed quantum random number generation by measuring phase noise of a single-mode laser[J]. Optics letters, 2010, 35(3): 312-314.

[6]Uchida A, Amano K, Inoue M, et al. Fast physical random bit generation with chaotic semiconductor lasers[J]. Nature Photonics, 2008, 2(12): 728-732.

[7]Von Neumann J. 13. various techniques used in connection with random digits[J]. Appl. Math Ser, 1951, 12(36-38): 3.

[8]Zheng Z, Zhang Y, Huang W, et al. 6 Gbps real-time optical quantum random number generator based on vacuum fluctuation[J]. Review of Scientific Instruments, 2019, 90(4): 043105.

[9]郑子勇. 连续变量源无关量子随机数发生器研究[D].北京邮电大学,2020.DOI:10.26969/d.cnki.gbydu.2020.000073.

[10]Ma X, Xu F, Xu H, et al. Postprocessing for quantum random-number generators: Entropy evaluation and randomness extraction[J]. Physical Review A, 2013, 87(6): 062327.

[11]罗钰杰. 量子保密通信中随机性提取的实现技术研究及应用[D].中国电子科技集团公司电子科学研究院,2021.DOI:10.27728/d.cnki.gdzkx.2021.000131.

[12]ENT. A Pseudorandom Number Sequence Test Program [EB/OL]. 2022. https://www.fourmilab.ch/random/

[13]Rukhin A, Soto J, Nechvatal J, et al. A statistical test suite for random and

pseudorandom number generators for cryptographic applications[R]. Booz-allen and hamilton inc mclean va, 2001.

[14] NIST information Technology Laboratory [EB/OL]. 2018. https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software