

# Rdrand——基于硬件随机数发生器生成真随机数

白博臣、何骐多、夏营

(四川大学 物理学拔尖计划)



Figure 1: 白博臣



Figure 2: 何骐多



Figure 3: 夏营

## 摘 要

RDRAND 是一个基于 Intel 芯片实现的利用电阻热噪声实现真随机数生成的一个函数。本文在 C/C++, Python, Java 等语言环境下, 基于随机行, 均匀性, 效率等方向, 采用《GM/T 0005-2021 随机性检测规范》, 对该函数进行了测试。讨论了该函数在密码学中的应用, 并对其未来进行了展望。

**关键词:** RdRand, 电阻热噪声, GM/T 0005-2021 随机性检测规范, 密码学

---

## 1 引言

在蒙特卡洛方法的学习过程中，我们认识到随机数在数值积分中具有极为重要的作用。随机数的随机性质与均匀性质往往会决定计算结果的好坏。目前主流用于计算使用的随机数多是伪随机数与准随机数，这类随机数往往是通过固定的算法或公式产生的序列。其虽然具有较好的随机性质与均匀性质，但这类随机数生成的结果是可预测的，是有规律可循的，故而称为伪随机数或准随机数。而真随机数生成的结果则是完全不可预测的，是真正意义上的“随机”。

出于兴趣，我们小组对于真随机数进行了一定的探索与研究。在这一过程中，我们发现基于算法直接实现真随机生成比较困难，而一些特定的物理现象本身便具有真随机性，如电阻热噪声、大气噪声等。而 Intel 正好在其近几年发行的 IVB 架构的 CPU 中提供了电阻热噪声的相关数据，于是我们便基于这一硬件提供的电阻热噪声的真随机性实现了真随机数生成器，并基于《GM/T 0005-2021 随机性检测规范》对其随机性质，均匀性质进行了检验，确保我们所得到的随机数是具有良好应用可能性的。

## 2 生成器原理简介

电阻的热噪声是其自身产生的噪声，是由导体中电子的振动所引起的，其存在于所有的电子器件与传输介质中。它是温度变化的结果，但不受频率变化的影响。热噪声属于电阻的本征噪声，与外界变量的联系极其微弱。

噪声产生的源头在于导体中的电子热运动，电子的运动是一种无规则运动，同时因为电子热运动的自由程很短，所以单个电子的运动可以看作产生了一个电流脉冲。而整个导体由热运动所产生的电流，便是导体内所有电子热运动产生电流的叠加，在某一时刻，沿某一方向的热运动可能略占优势，进而产生沿这一方向的电流。但由于电子运动的无规则性，占优势的热运动方向是无法预测的，因而其所带来的噪声具有真随机性。

而基于电阻热噪声的真随机数生成器便是通过将这个噪声放大，并通过比较器产生二进制随机数序列，是基本没有算法与公式参与的，完全依赖于硬件本身物理性质得到的，产生随机二进制序列的随机数生成器。

## 3 随机性检验

因为所得序列为二进制序列，所以可以采用《GM/T 0005-2021 随机性检测规范》给出的 15 种随机性检验，包括对其随机性、均匀性、自相关性等的检验对所得随机数

---

进行检验，检验方式简介如下：

### 3.1 单比特频数检测方法

单比特频数检测是最基本的检测，用来检测一个二元序列中 0 和 1 的个数是否相近。随机序列应具有较好的 0,1 平衡性。在一个足够长的随机序列内，其统计值应服从标准正态分布。

### 3.2 块内频数检测方法

块内频数检测用来检测待检序列的  $m$  位子序列中的 1 的个数是否接近于  $\frac{1}{m}$ 。对于随机数列而言，其任意长度的子序列中 1 的个数都应该接近于  $\frac{1}{m}$ 。

### 3.3 扑克检测

对任意长度  $m$  的子序列，其共有  $2^m$  种不同的组成。对于长度为  $N$  的待检随机序列而言，将其划分为  $\lfloor \frac{N}{m} \rfloor$ （向下取整）个非重叠的长度为  $m$  的这  $2^m$  种子序列出现的个数应当接近。

### 3.4 重叠子序列检测

扑克检测类似，但是区间划分不同。将长度为  $n$  的待检序列划分为  $n$  个可以重叠的长度为  $m$  的子序列，在这些子序列中，这  $2^m$  种出现的个数应当接近。

### 3.5 游程总数检测

游程是二元序列的一个子序列，由连续的 0 或 1 组成，并且其前导和后继元素都与其本身不同。游程总数检测主要检测待检序列中游程的总数是否服从随机性要求。

### 3.6 游程分布检测

连续 1(或 0) 的一个游程称为一个块 (或一个间断)。根据 Golomb 共设，如果待检二元序列是随机的，则相同长度的数目接近一致，且长度为  $i$  的游程个数约占总游程个数的  $2^{-i}$ 。

---

### 3.7 块内最大游程检测

将待检序列划分成  $N$  个等长的子序列，根据各个子序列中最大“1”或“0”游程的分布来评价待检序列的随机性。

### 3.8 二元推导检测

二元推导序列是由初始序列生成的一个新的序列。对于长度为  $n$  的一个二元初始序列，依次将初始序列中两个相邻比特做异或操作，即可得到该序列的一次二元推导序列，长度为  $n-1$ 。依次执行上述操作  $k$  次，即可得到  $k$  次二元推导序列，长度为  $n-k$ 。二元推导检测的目的是判定第  $k$  次二元推导序列中 0 和 1 的个数是否接近一致。

### 3.9 自相关检测

自相关检测用来检测待检序列与将其左移 (逻辑左移) $d$  位后所得新序列的关联程度。一个随机序列应该和将其左移任意位所得的新序列都是独立的，故其关联程度也应该很低。

### 3.10 矩阵秩检测

矩阵秩检测用来检测待检序列中给定长度的子序列之间的线性独立性。由待检序列构造矩阵，然后检测矩阵的行或列之间的线性独立性，矩阵秩的偏移程度可以给出关于线性独立性的量的认识，从而影响对序列随机性好坏的评价。

### 3.11 累加和检测

累加和检测将待检序列的各个子序列中最大的偏移 (与 0 之间)，也就是最大累加和与一个随机序列应具有的最大偏移相比较，以判断待检序列的最大偏移是过大还是过小。实际上，随机序列的最大偏移应该接近 0，所以累加和不能过大，也不能过小 (累加和可以是负数)。根据最大偏移值来判断待检序列的随机程度。

### 3.12 近似熵检测

近似熵检测通过比较  $m$  位可重叠子序列模式的频数和  $m+1$  位可重叠子序列模式的频数来评价其随机性。

---

### 3.13 线性复杂度检测

将待检序列划分为  $N$  个长度为  $m$  的子序列, 此时  $n = N * m$ , 然后利用 Berlekamp-Massey 算法 (具体内容可网上查询), 计算每个子序列的线性复杂度  $L_i$ , 然后利用其给出的统计值来进行随机性判断。

### 3.14 Maurer 检测

Maurer 通用统计 (简称通用统计) 检测主要检测待检序列能否被无损压缩。如果待检序列能被显著地压缩, 则认为该序列是不随机的, 因为随机序列是不能被显著压缩的。

通用统计检测可以用来检测待检序列多方面的特性, 但这并不意味着通用统计检测是前面几个检测的拼装, 通用统计检测完全采取了和其他检测所不同的方法, 可以检测待检序列某些统计上的缺陷。一个序列可以通过通用统计检测当且仅当这个序列是不可压缩的。

### 3.15 离散傅里叶检测

离散傅立叶变换检测使用频谱的方法来检测序列的随机性。对待检序列进行傅立叶变换后可以得到尖峰高度, 根据随机性的假设, 这个尖峰高度不能超过某个门限值 (与序列长度  $n$  有关), 否则将其归入不正常的范围; 如果不正常的尖峰个数超过了允许值, 即可认为待检序列是不随机的。

## 4 随机数检测判定

### 4.1 样本通过率判定

对于每一个随机性检测项目, 统计二元序列样本集中  $P_{value}$  值大于或等于  $\alpha$  的样本个数, 本文确定的用于样本通过率检测的显著性水平  $\alpha = 0.01$ 。

记样本数量为  $s$ , 当通过某检测项目的样本个数大于或等于  $s(1 - \alpha - 3\sqrt{\frac{\alpha(1-\alpha)}{s}})$  时, 认为该样本集通过该项检测, 否则未通过此项检测。例如, 样本数量为 1000 个, 则通过该检测项目的样本个数应大于或等于 981。

---

## 4.2 样本分布均匀性判定

对于每一个随机性检测项目，二元序列样本集中各样本的  $Q_{value}$  值应该在区间  $[0, 1]$  均匀分布。记样本数量为  $s$ ，将区间  $[0, 1]$  分为  $k$  个均匀的子区间，统计二元序列样本集中各样本的  $Q_{value}$  在  $k$  个子区间的实际数量  $F_i$ ，并利用  $\chi^2$  分布与理论值  $s/k$  进行比较，计算统计值  $V = \sum_{i=1}^k \frac{(F_i - s/k)^2}{s/k}$ ，并计算  $P_T = igamc(\frac{k-1}{2}, \frac{V}{2})$ 。当  $P_T$  大于或等于  $\alpha_T$  时，认为该二元序列样本集通过此项目检测；否则，未通过此项目检测。本文件确定的用于样本分布均匀性检测的显著性水平  $\alpha_T = 0.0001$ ，子区间数量  $k = 10$ 。

## 5 随机性检测

我们不仅对 RdRand 随机数进行了随机性检测，也对部分主流随机数进行了检测，包括 Python 中的 random 库与 numpy 库的伪随机数，参数为  $seed = 123, a = 1664525, c = 1013904223, m = 2^{32}$  的 LCG 以及 Matlab 中的伪随机数。

经过随机性检验我们可以发现，Rdrand 通过随机性检验的次数要多于另外四种随机数，LCG，Matlab 伪随机数和 random 库伪随机数均出现了不能通过随机性检验的情况，LCG 虽然能通过样本通过率判定，但是在分布均匀性上十分糟糕，使用简单的 LCG 很难得到很好的随机数；Random 库伪随机数在块内最大“1”游程检测 ( $m=10000$ ) 中没有通过样本通过率判定；Matlab 伪随机数没有通过单比特频数检测的样本通过率检测。

这也反映了 Rdrand 随机数的优越性，我们可以确保在 1000 个样本数，每个样本内 1000000bit 数的情况下，Rdrand 随机数的随机性要更加出色，配合原理层面的安全性使其尤其在密码学方向具有自己的独有优势。

具体结果如下：

RdRand				
用例	方法/参数	结果		
		P 样本通过率	Q 分布均匀性	结论
1	单比特频数检测	990	0.623587	通过
2	块内频数检测 ( m = 10000 )	990	0.492436	通过
3	扑克检测 ( m = 4 )	987	0.839507	通过
4	扑克检测 ( m = 8 )	989	0.271619	通过
5	重叠子序列检测 1 ( m = 3 )	988	0.823725	通过
6	重叠子序列检测 2 ( m = 3 )	989	0.144915	通过
7	重叠子序列检测 1 ( m = 5 )	986	0.877852	通过
8	重叠子序列检测 2 ( m = 5 )	994	0.509162	通过
9	游程总数检测	990	0.875539	通过
10	游程分布检测	994	0.447490	通过
11	块内最大“1”游程检测 ( m = 10000 )	985	0.496351	通过
12	块内最大“0”游程检测 ( m = 10000 )	982	0.672470	通过
13	二元推导检测 ( k = 3 )	995	0.478839	通过
14	二元推导检测 ( k = 7 )	988	0.607993	通过
15	自相关检测 ( d = 1 )	990	0.880145	通过
16	自相关检测 ( d = 2 )	989	0.963094	通过
17	自相关检测 ( d = 8 )	990	0.840367	通过
18	自相关检测 ( d = 16 )	988	0.922241	通过
19	自矩阵秩检测	989	0.993090	通过
20	累加和检测 ( 前向 )	988	0.580051	通过
21	累加和检测 ( 后向 )	988	0.149073	通过
22	近似熵检测 ( m = 2 )	988	0.749884	通过
23	近似熵检测 ( m = 5 )	990	0.871642	通过
24	线性复杂度检测 ( m = 500 )	983	0.602803	通过
25	线性复杂度检测 ( m = 1000 )	989	0.454995	通过
26	通用统计检测	990	0.429008	通过
27	离散傅立叶检测	986	0.052275	通过
结论	通过			

Figure 4: 对 RdRand 随机数的随机性检验结果

LCG				
用例	方法/ 参数	结果		
		P 样本通过率	Q 分布均匀性	结论
1	单比特频数检测	1000	0.000000	不通过
2	块内频数检测 ( m = 10000 )	1000	0.000000	不通过
3	扑克检测 ( m = 4 )	1000	0.000000	不通过
4	扑克检测 ( m = 8 )	1000	0.000000	不通过
5	重叠子序列检测 1 ( m = 3 )	999	0.000000	不通过
6	重叠子序列检测 2 ( m = 3 )	1000	0.000000	不通过
7	重叠子序列检测 1 ( m = 5 )	1000	0.000000	不通过
8	重叠子序列检测 2 ( m = 5 )	1000	0.000000	不通过
9	游程总数检测	999	0.000000	不通过
10	游程分布检测	1000	0.000000	不通过
11	块内最大“1”游程检测 ( m = 10000 )	996	0.082513	通过
12	块内最大“0”游程检测 ( m = 10000 )	997	0.733899	通过
13	二元推导检测 ( k = 3 )	999	0.000000	不通过
14	二元推导检测 ( k = 7 )	998	0.005742	通过
15	自相关检测 ( d = 1 )	999	0.000000	不通过
16	自相关检测 ( d = 2 )	999	0.000000	不通过
17	自相关检测 ( d = 8 )	994	0.073872	通过
18	自相关检测 ( d = 16 )	994	0.489508	通过
19	自矩阵秩检测	990	0.443761	通过
20	累加和检测 ( 前向 )	1000	0.000000	不通过
21	累加和检测 ( 后向 )	1000	0.000000	不通过
22	近似熵检测 ( m = 2 )	999	0.000000	不通过
23	近似熵检测 ( m = 5 )	1000	0.000000	不通过
24	线性复杂度检测 ( m = 500 )	998	0.044652	通过
25	线性复杂度检测 ( m = 1000 )	991	0.182053	通过
26	通用统计检测	967	0.000000	不通过
27	离散傅立叶检测	0	0.000000	不通过
结论	不通过			

Figure 5: 对 LCG 随机数的随机性检验结果



Random				
用例	方法/参数	结果		
		P 样本通过率	Q 分布均匀性	结论
1	单比特频数检测	988	0.838645	通过
2	块内频数检测 ( m = 10000 )	994	0.727851	通过
3	扑克检测 ( m = 4 )	993	0.184549	通过
4	扑克检测 ( m = 8 )	987	0.281232	通过
5	重叠子序列检测 1 ( m = 3 )	991	0.739918	通过
6	重叠子序列检测 2 ( m = 3 )	990	0.674543	通过
7	重叠子序列检测 1 ( m = 5 )	986	0.844641	通过
8	重叠子序列检测 2 ( m = 5 )	990	0.546283	通过
9	游程总数检测	990	0.797204	通过
10	游程分布检测	994	0.434510	通过
11	块内最大“1”游程检测 ( m = 10000 )	980	0.763677	不通过
12	块内最大“0”游程检测 ( m = 10000 )	991	0.255057	通过
13	二元推导检测 ( k = 3 )	994	0.914672	通过
14	二元推导检测 ( k = 7 )	987	0.630872	通过
15	自相关检测 ( d = 1 )	990	0.758773	通过
16	自相关检测 ( d = 2 )	983	0.401199	通过
17	自相关检测 ( d = 8 )	986	0.707513	通过
18	自相关检测 ( d = 16 )	990	0.220159	通过
19	自矩阵秩检测	989	0.467322	通过
20	累加和检测 ( 前向 )	988	0.269590	通过
21	累加和检测 ( 后向 )	989	0.606954	通过
22	近似熵检测 ( m = 2 )	990	0.749884	通过
23	近似熵检测 ( m = 5 )	992	0.377007	通过
24	线性复杂度检测 ( m = 500 )	992	0.895633	通过
25	线性复杂度检测 ( m = 1000 )	992	0.178604	通过
26	通用统计检测	996	0.992952	通过
27	离散傅立叶检测	991	0.703417	通过
结论	不通过			

Figure 6: 对 random 随机数的随机性检验结果

Np.random				
用例	方法/参数	结果		
		P 样本通过率	Q 分布均匀性	结论
1	单比特频数检测	993	0.603841	通过
2	块内频数检测 ( m = 10000 )	994	0.308561	通过
3	扑克检测 ( m = 4 )	990	0.971354	通过
4	扑克检测 ( m = 8 )	991	0.852217	通过
5	重叠子序列检测 1 ( m = 3 )	995	0.041709	通过
6	重叠子序列检测 2 ( m = 3 )	991	0.332188	通过
7	重叠子序列检测 1 ( m = 5 )	989	0.593478	通过
8	重叠子序列检测 2 ( m = 5 )	993	0.948298	通过
9	游程总数检测	993	0.333754	通过
10	游程分布检测	993	0.809249	通过
11	块内最大“1”游程检测 ( m = 10000 )	990	0.403844	通过
12	块内最大“0”游程检测 ( m = 10000 )	984	0.962280	通过
13	二元推导检测 ( k = 3 )	994	0.379555	通过
14	二元推导检测 ( k = 7 )	986	0.112047	通过
15	自相关检测 ( d = 1 )	993	0.358641	通过
16	自相关检测 ( d = 2 )	988	0.473064	通过
17	自相关检测 ( d = 8 )	984	0.293235	通过
18	自相关检测 ( d = 16 )	990	0.723804	通过
19	自矩阵秩检测	988	0.507182	通过
20	累加和检测 ( 前向 )	991	0.966244	通过
21	累加和检测 ( 后向 )	989	0.175691	通过
22	近似熵检测 ( m = 2 )	995	0.033473	通过
23	近似熵检测 ( m = 5 )	992	0.894201	通过
24	线性复杂度检测 ( m = 500 )	989	0.325977	通过
25	线性复杂度检测 ( m = 1000 )	985	0.892036	通过
26	通用统计检测	982	0.154629	通过
27	离散傅立叶检测	987	0.892036	通过
结论	通过			

Figure 7: 对 np.random 随机数的随机性检验结果

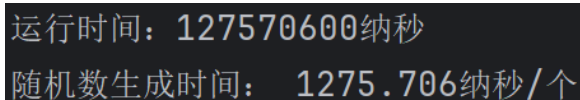
Matlab Random				
用例	方法/参数	结果		
		P 样本通过率	Q 分布均匀性	结论
1	单比特频数检测	979	0.347257	不通过
2	块内频数检测 ( m = 10000 )	987	0.293952	通过
3	扑克检测 ( m = 4 )	995	0.805569	通过
4	扑克检测 ( m = 8 )	996	0.012430	通过
5	重叠子序列检测 1 ( m = 3 )	989	0.963897	通过
6	重叠子序列检测 2 ( m = 3 )	987	0.616305	通过
7	重叠子序列检测 1 ( m = 5 )	995	0.423545	通过
8	重叠子序列检测 2 ( m = 5 )	993	0.727851	通过
9	游程总数检测	989	0.736912	通过
10	游程分布检测	986	0.538182	通过
11	块内最大“1”游程检测 ( m = 10000 )	989	0.741918	通过
12	块内最大“0”游程检测 ( m = 10000 )	989	0.695200	通过
13	二元推导检测 ( k = 3 )	989	0.289667	通过
14	二元推导检测 ( k = 7 )	984	0.980613	通过
15	自相关检测 ( d = 1 )	989	0.811993	通过
16	自相关检测 ( d = 2 )	993	0.371941	通过
17	自相关检测 ( d = 8 )	993	0.867692	通过
18	自相关检测 ( d = 16 )	991	0.930607	通过
19	自矩阵秩检测	987	0.034712	通过
20	累加和检测 ( 前向 )	986	0.886905	通过
21	累加和检测 ( 后向 )	986	0.918502	通过
22	近似熵检测 ( m = 2 )	989	0.955384	通过
23	近似熵检测 ( m = 5 )	987	0.792508	通过
24	线性复杂度检测 ( m = 500 )	991	0.335324	通过
25	线性复杂度检测 ( m = 1000 )	993	0.361112	通过
26	通用统计检测	992	0.197450	通过
27	离散傅立叶检测	988	0.766607	通过
结论	不通过			

Figure 8: 对 Matlab random 随机数的随机性检验结果

## 6 随机数应用

### 6.1 随机数效率

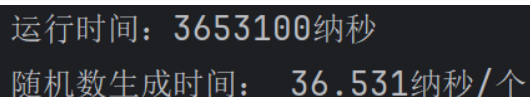
我们通过测量计算 100 000 个点的运行时间（纳秒值）如下



```
运行时间: 127570600纳秒
随机数生成时间: 1275.706纳秒/个
```

Figure 9: 真随机数发生器速度

对比机器自带伪随机



```
运行时间: 3653100纳秒
随机数生成时间: 36.531纳秒/个
```

Figure 10: 伪随机发生器速度

牺牲了一定的速度，但换来了更高的随机性，同时其相较于部分其他真随机数生成器在速度上有一定优势。这种随机性带来的难以预测，使得其在安全性上有着较大提升。

### 6.2 具体应用

正如前文提到的，这种随机数的诸多性质，决定了其在密码学中具有重要的地位。如今普遍使用的 RSA 体制中，便可使用该随机数进行加密。

RSA 体制大致阐释如下，用户的数字签名  $a(0 < a < N - 1)$ ，通过密钥  $a^e \equiv b \pmod{N}$  进行加密，收件方收到后，在通过该用户提供的解钥  $b^d \equiv a \pmod{N}$  进行解密。

使用常规的机器随机数，在大规模的并行计算时，完全有可能出现两条线程使用相同的种子（即系统时有相同的毫秒值），此时生成的随机数便会出现重复，在将其作为密钥加密时，便有可能出现两个不同的用户公用相同的密钥。此时，就会导致即使非该用户的目标对象，也能够得到解钥，从而破解密文。

在使用 RdRand 时，由于其是基于物理上的混沌系统产生的随机数，其的不可预测性和随机性，保证了不同用户无法截获到其他人的解钥，从而提高了网络传输的安全性。

---

## 7 总结

本文概述了 RdRand 的基本原理，并根据国标《GM/T 0005-2021 随机性检测规范》对该随机数发生其进行了检验。结果说明，RdRand 是具有良好的随机性和均匀性的随机数发生器，可以用于实际应用中，例如密码学中。基于其良好的性质，相信在未来，会有更好的发展。