基于Intel RNG的真随机数生成器研究

1 引言

随机数已经广泛地应用于仿真、抽样、数值分析、计算机程序设计、决策、美学和娱乐之中[1]。在生物医学中，随机数的使用也非常广泛，例如生物医学工程中随机数的使用；生物信息学中进行基因序列分析；组织学和病理学研究中随机视野的采样；流行病调查、医学检验和生物体视学中的随机抽样等。

一个数字，例如8，不能称之为是随机数。随机数是指具有确定分布和概率的、位于一定阈值内的一组数字。除了传统的均匀分布随机数之外，还有非均匀分布随机数，例如二项分布、Poisson分布和指数分布等。本研究中所指的随机数是均匀分布的随机数，其序列中的数字满足独立性和分布均匀性的特点。

常见的随机数生成器有两种：使用数学算法的伪随机数发生器(pseudo random number generator)和以物理随机量作为发生源的真随机数生成器(true ran-dom number generator)。前者的研究较为深入，除了常见的线性同余发生器(linear congruence generator，LCG)和反馈位寄存器法(feedback shift register meth-ods，FSR)等之外，还有很多基于混沌、数论和密码学等的随机数生成器算法。他们的共同特点是使用方便，但是由于通过算法不能实现真正的随机数[1][2]，常常存在一些缺陷，例如稀疏网格，周期性依赖于字长；不居中现象等。要获取真正随机的真随机数，常使用硬件随机数生成器的方法来获取真随机数[3][4]，这些真随机数都是使基于特定的真随机数发生源，例如热噪声、电流噪声等，每次获取的随机数都是不可测的，具有很好的随机性。目前市场上也有不少商业化的真随机数生成器，例如ComScire QNG和Protego SG100 TRNG等。但是利用硬件随机数生成器增加了额外的成本，由于硬件的通用性导致应用范围有限，升级困难；另外如果用于商业和通讯，还存在版权和保密问题。

构建一种基于硬件真随机数发生源，具有良好随机性，同时又具有伪随机数生成器方便易用的特点，而且具有广泛应用价值。作者采用了Intel RNG作为硬件随机数发生源。

Intel最早在i810系列芯片组中就已经集成了基于硬件的随机数生成器(random number generator, RNG)的芯片。硬件随机数生成器和从Pentium III 开始的CPU中添加的序列号功能一起构成Intel在1996年提出的公共数据安全体系结构(common data security architecture，CDSA)的两个重要组成部分。随机数发生器位于芯片组的FWH(firmware hub)中，常见的FWH为82802AB/82802AC。主要机理是利用Johnson热噪声[5]放大后，影响一个由电压控制的振荡器，通过另外一个频率比电压控制的振荡器高100倍的高速振荡器信号收集数据[6]，并经过基于John von Neumann的原理进行数字的后期处理(除偏)，最后将随机数输出。其原理如图1所示。

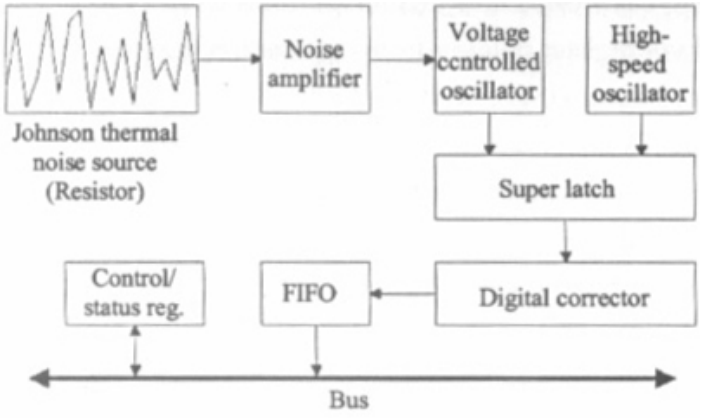

图 1 Intel RNG 工作原理图

Fig.1 Principles for the working of Intel RNG

图1 Intel RNG工作原理图

Fig.1 Principles for the working of Intel RNG

2 方法

[7]

Intel的RNG与开发平台的接口主要依靠硬件状态寄存器(hardware status register)。生成随机数的主要寄存器如下：

(1)物理地址：0xFFBC015F(可读写)。地址0x40开始的6个字节表明RNG 的状态(RNG Present)。如果为1，说明RNG在FHW中，如果为0，则说明RNG不存在。地址0x41开始的1个字节表明RNG的可读状态(RNG enable)。如果为1，说明RNG目前可用，为0则表示RNG不可用。

(2)物理地址：0xFFBC0160(只读)。地址0x10表示数据状态(data available)。如果为1，说明RNG的数据寄存器中包含有效的随机数数据。如果为0，说明RNG数据寄存器中的数据为无效数据。

(3)物理地址：0xFFBC0161(只读)。此位即为随机数数据。如果该位数值为0，说明目前RNG不可用。

在使用RNG以前，需要按照下面的顺序对设备进行初始化：(1)检测FWH设备；(2)检测RND设备；(3)初始化RND设备。生成随机数的步骤：(1)检测RNG状态；(2)轮询时间(polling time)；(3)读取数据。

设计伪代码表示如下：

```
// 检测和初始化RNG设备的伪语言
uint 8* pFeatureSpace = 0xFFBC0000;
uint 8* pHardwareStatus = pFeatureSpace + 0x015F;
uint 8* pRNGStatus = pFeatureSpace + 0x0160;
uint 8* pRNGData = pFeatureSpace + 0x0161;
//检查82802 FHW代码(略)
//检查RNG设备是否存在
if (( *pHardwareStatus & 0x40) = = 0 )
then (FAIL);  // 说明RNG不存在
//将RNG设备设置为enable
*pHardwareStatus = ( *pHardwareStatus | 0x01 ) ;
//首次读取RNG数据，以清除以前的数据
if ( GetRandomData() = = FAIL )
then (FAIL);                          //不能初始化随机数
//测试RNG
if ( RunFipsTest() = = FAIL)
then (FAIL)                          //初始测试失败
//下面是生成随机数的代码
//获取32位的随机数数据
```

```
uint32 dwRandomData = 0;
//寄存器位置
uint8 *pFeatureSpace = 0xFFBC0000;
uint8 *pRNGStatus = pFeatureSpace + 0x0160;
uint8 *pRNGData = pFeatureSpace + 0x0161;
//定义随机数的一个指针
uint8 *pRandomDataPointer = &dwRandomData;
//读取数据
for (int i=0; i<4; i++)
{
while ( !(pRNGStatus & 0x01 ) )
{
//如果RNG还不可用，继续轮询
}
//如果RNG Status可用，则读取数据
( *pRandomDataPointer) ++ = *pRNGData;
}
```

3 实验

要能直接读取Intel RNG中的寄存器信息，需要安装Intel Security Driver(ISD)[8]。根据上面的伪代码，我们用Microsoft Visual C++ 6编制了计算机程序，在兼容机(intel PIII 667, 主板：ASUS TUSL2-C ，Intel 815E芯片组，512M KingMax PC133 RAM)上进行了测试。操作系统为Microsoft Windows 2000 Pro-fessional 简体中文版(已安装Service Pack 4)。根据最少需要80Mbit连续RNG输出的原则[6][9]，我们生成了100Mbit连续随机数，部分数据如下(200个，域)：

表 1 生成的 200 个随机数

Tab.1 The 200 random numbers generated

0.0189562	0.2455971	0.7501397	0.7738837	0.5658537	0.9681778
0.5131224	0.379891	0.3217094	0.0087847	0.670883	0.5165024
0.6405893	0.3734425	0.4193796	0.4770382	0.0934588	0.0202942
0.9146811	0.142218	0.5648365	0.9818573	0.4546964	0.5235118
0.0994502	0.2143726	0.4950465	0.3963824	0.9981167	0.6254366
0.2157137	0.5757166	0.7462568	0.9881563	0.173239	0.9752044
0.3520151	0.6061378	0.7326058	0.2659794	0.2883326	0.7999557
0.6872617	0.3701395	0.6229448	0.0059153	0.996528	0.57723
0.2245937	0.2252862	0.4395079	0.6273572	0.4183999	0.1756446
0.5919741	0.8220823	0.3440372	0.470711	0.4612481	0.3523884
0.4550593	0.551638	0.3661047	0.3898244	0.8864507	0.576909
0.2996686	0.5470893	0.2391061	0.9318526	0.5484601	0.7389704
0.1065901	0.3876584	0.1694808	0.4409012	0.0299995	0.922616
0.0764988	0.9944125	0.2934536	0.8450856	0.7300154	0.2967225
0.552414	0.8027337	0.395688	0.1726395	0.3488014	0.7317781
0.5429521	0.4932401	0.2126036	0.7770578	0.2600763	0.1787157
0.9900961	0.2266951	0.9045249	0.0735566	0.5418277	0.2952597
0.7199887	0.2997218	0.8843332	0.9755338	0.1434947	0.0538404
0.894285	0.7320632	0.2089497	0.6596838	0.5643061	0.4436777
0.9528908	0.4641166	0 .3080883	0.6280128	0.0094999	0.2932748
0.0892192	0.1606684	0.2527966	0.5797736	0.0453309	0.1535448
0.5147923	0.19002	0.0139874	0.8007123	0.0224787	0.0749317
0.2470483	0.9670101	0.8762832	0.0446412	0.2967659	0.9787027
0.4167025	0.0000696	0.5789463	0.4777745	0.5951046	0.5285646
0.8620476	0.6057532	0.9352996	0.8043424	0.4975013	0.759161
0.9868408	0.9414447	0.3906571	0.1749776	0.3423593	0.5784809
0.160597	0.1103672	0.6464102	0.9816703	0.3766149	0.0563468
0.0155921	0.7400952	0.5916386	0.8553198	0.2053189	0.2747398
0.4177088	0.1770563	0.1529314	0.8921459	0.572069	0.5760177
0.5437102	0.6598239	0.2055213	0.4326412	0.8843047	0.6162253
0.5869474	0.3851733	0.0149354	0.8455775	0.4514926	0.5268177
0.5583608	0.2673801	0.46855	0.3667935	0.9361369	0.4696113
0.1702609	0.1403611	0.8243053	0.9828174	0.5702752	0.7995637
0.0451431	0.4683332				

随机数的随机性检验常用χ^2检验(罕用KS检验和经验检验)；分布均匀性检验我们使用χ^2拟合优度检验法；独立性常用游程检验法检验[10]。我们使用美国国家技术标准局 (NIST)的FIPS(Federal Informa-tion Processing Standards Publication)140-1[1][11][12]对所生成的随机数进行了测试。测试结果如下：

表 2 FIPS 140-1 测试结果

Tab.2 Result of FIPS 140-1

Run #	Monobit Test	Poker Test	Runs Test	Long Run Test
1	X=9,973	X=11.43	Passed	Passed
2	X=9,798	X=27.59	Passed	Passed
3	X=9.878	X=9.73	Passed	Passed
4	X=9.894	X=13.54	Passed	Passed
5	X=10.043	X=11.03	Passed	Passed

表2显示的结果分析如下：

1.Monobit检验结果符合 ，检验通过，表明其bit-stream中的0和1没有显著性差异。

2.扑克检验(Poker Test)结果符合 ，检验通过，表明产生的随机数的组合规律与理论值差异不明显。

3.运行检验(Run Test)通过，表明随机数通过独立性检验。

为了验证其分布的均匀性，我们下面再用χ^2拟合优度检验法对这些数作分布均匀性检验。假设H_0：r_1，r_2，r_3,.....，r_n为均匀总体的随机样本。将样本r_1，r_2，r_3,.....，r_n (n=500)的取值范围分布在m个(m=10)等宽的区间，用$\left[\frac{i-1}{m},\frac{i}{m}\right]$(i=1，2，3...，m)来表示第I个小区间，即分成[0.0,0.1)，[0.1,0.2)，[0.2,0.3)，[0.3,0.4)，[0.4,0.5)，[0.5,0.6)，[0.6,0.7),[0.7,0.8)，[0.8,0.9)，[0.9,1.0)，共10个区间。设{r_j}(j=1，2，3...，m)落入每个小区间的数目为n_i(i=1，2，3...，m)。根据假设，{r_j}落入每个小区间的概率为$\frac{1}{m}$，第I个小区间的理论频数$\frac{n}{m}$，统计量

$$V=\sum_{i=1}^{m}\frac{(n_i-u_i)^2}{u_i}=\frac{m}{n}\sum_{i=1}^{m}(n_i-\frac{m}{n})^2$$

渐进服从$\chi^2$2(m-1)。

使用Origin 7.5(SR1)对500个随机数进行频数统计，得出以下数值：

表 3 分区频数统计结果

Tab.3 Result of frequency count by sections

I	1	2	3	4	5	6	7	8	9	10
n_i	56	51	45	48	53	55	49	48	46	49

$$V=\frac{m}{n}\sum_{i=1}^{m}(n_i-\frac{n}{m})^2=\frac{1}{50}\sum_{i=1}^{10}(n_i-50)^2=3.24$$

给定显著性水平α=0.05，取m=10，则自由度λ=m-1=9。查字χ^2分布临界值为18.307。由上面给出的计算值可知，V<18.307，所以假设H_0成立。r_n为来自均匀分布的总体的随机样本。

为了进一步验证其分布的均匀性，我们对每个数字的频数作χ^2拟合优度检验。使用统计软件SPSS 11.5分别统计每个数字出现的频数，结果如下：

表 4 500 个 域随机数字的频数统计

Tab.4 Frequency of 500 random numbers in domain

Number	Frequency	Number	Frequency	Number	Frequency	Number	Frequency	Number	Frequency
0	2								
0.01	5	0.21	4	0.41	5	0.61	3	0.81	3
0.02	7	0.22	3	0.42	4	0.62	3	0.82	5
0.03	11	0.23	6	0.43	5	0.63	3	0.83	4
0.04	4	0.24	5	0.44	5	0.64	3	0.84	7
0.05	5	0.25	7	0.45	5	0.65	3	0.85	6
0.06	4	0.26	6	0.46	6	0.66	2	0.86	7
0.07	4	0.27	9	0.47	5	0.67	3	0.87	9
0.08	6	0.28	6	0.48	4	0.68	2	0.88	3
0.09	2	0.29	5	0.49	3	0.69	5	0.89	12
0.10	2	0.30	4	0.50	9	0.70	3	0.90	8
0.11	4	0.31	4	0.51	7	0.71	4	0.91	6
0.12	7	0.32	6	0.52	5	0.72	3	0.92	6
0.13	3	0.33	3	0.53	8	0.73	3	0.93	7
0.14	6	0.34	7	0.54	5	0.74	2	0.94	8
0.15	5	0.35	4	0.55	8	0.75	2	0.95	7
0.16	6	0.36	3	0.56	5	0.76	2	0.96	2
0.17	2	0.37	2	0.57	4	0.77	3	0.97	4
0.18	9	0.38	5	0.58	4	0.78	6	0.98	3
0.19	6	0.39	8	0.59	6	0.79	5	0.99	5
0.20	5	0.40	5	0.60	8	0.80	6		
								1	4

考虑到使用round()函数对于小数四舍五入的影响，实验数据中，排除数字0和1的频数2和4。则剩下99个数字的理论理想频数为A=494/99=4.99，自由度v=99-1=98。假设：H_0：r_1，r_2，r_3,.....，r_n为来自均匀分布的总体的随机样本，H_1：r_1，r_2，r_3,.....，r_n为来自非均匀分布的总体的随机样本。定α=0.05。χ^2检验的统计量V为：

$$V=\frac{1}{n}\sum_{i=1}^{n}(n_i-A)^2=\frac{1}{99}\sum_{i=1}^{99}(n_i-4.99)^2=4.3534$$

查表$\chi^2_{0.05,\ 90}$=113.14，V$\chi^2_{20.05,\ 90}$，根据χ^2分布规律，V$\chi^2_{20.05,\ 90}$，$\chi^2_{20.05,\ 98}$，P>0.05。故在α=0.05水平不拒绝H_0。我们可以认为r_n为来自均匀分布总体的随机样本。

众所周知，一般的医学统计和医用数理统计书后面都备有随机数表。为了比较本研究中所建立的基于Intel RNG的随机数生成器与随机数表法所获取随机数的效果进行比较，我们选取了《卫生统计学》[14]后面随机数表中的所有随机数字(7500个，[0, 99])。我们将基于Intel RNG的随机数生成器所生成的7500个随机数乘以100后，截掉尾数，使其从(0, 1)域转换成[0, 100]域。由于截尾有四舍五入的问题，因此其界值1和100不作为统计之用。对随机数表(记为A组)和随机数生成器所生成的随机数(记为B组)使用Ori-gin7.5(SR1)(作图)和SPSS 11.5(统计频数)作频数图如下：

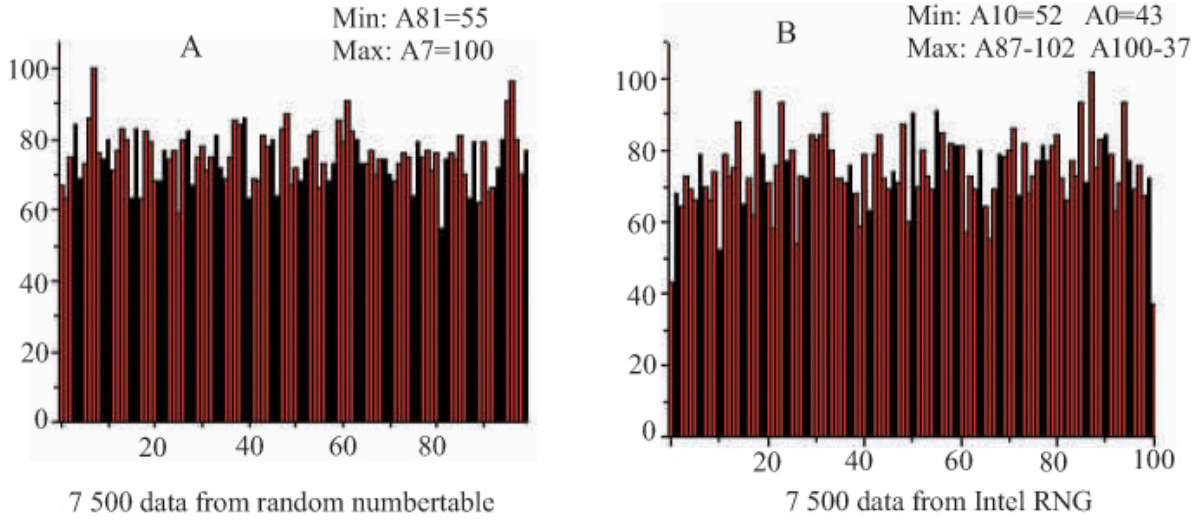

图2 7500个随机数的频数图

Fig.2 Chart for the frequency of 7500 random numbers

计算参数值如下(表5)：

表5 A组和B组频数统计参数

Tab.5 Statistical parameters of groups a and b

	Max. Frequency	Min. Frequency	Range Max.	Shift	Mean	MeanSkewing	*SD*	*SE*	*CV*%
A	100	55	45	50	49.368 4	0.631 6	28.915 88	0.333 89	0.385 55
B	102	52	50	50	50.551 2	0.551 2	28.764 68	0.332 15	0.383 53

表5中，均值偏移接近于0，变异系数也很小(<0.5%)，因此比较A组和B组的随机数的效果常用平均绝对误差、标准差等参数为准[15]。以上结果显示：均值偏移：A>B；SD和SE：A>B。因此，虽然两者的差别没有显著性(CV差别0.00202%)，但是由于B组的均值偏移小于A组，因此B组随机数的性能优于A。理想的均匀分布随机数应具有更好的均匀性与距理论频数更小的偏移，因此我们认为基于Intel RNG的随机数生成所产生的随机数比基于随机数表中的随机数具有更好的统计学性能。

4 讨论

以上真随机数生成方法可以用于任何基于Intel芯片组，包括810、815、820、840和840以后的所有Intel系列芯片组的个人计算机。这种生成方法能生成真正的随机数，而非依靠拟蒙特卡罗方法，例如计算机算法所生成的拟随机数，因此具有极高的随机数性能。

由于Intel的芯片组只能和Intel的CPU一起工作，但是由于Intel CPU和Intel 的芯片组已经占据了市场的大部份额，因此使用Intel RNG作为真随机数发生源还是具有较好的普遍性。同时，目前AMD、SIS、VIA等厂商都已经在各自芯片组中集成了随机数生成器，而且已经提交到Microsoft公司的可信赖计算平台联盟(Trusted Computing Platform Alliance)[13]，因此在Windows系统中，只要安装了相应的驱动程序，了解读取的寄存器地址，均可以通过直接调用的方式获取基于硬件的真随机数。

既然随机数表随处可得，我们为什么还需要构建真随机数生成器？这是因为：

1.周期性问题：一般统计学与数理统计书后面所附的随机数表中的随机数字的数目都是有限的。在本研究中《卫生统计学》中所附的随机数数字已经是很多了，但是也只有7500个。在需要大量使用随机数字的场合，例如流行病学调查，大样本的医学实验、生物体视学中分层随机抽样中，则可能会因为重复导致不可避免的周期性问题。

2.使用习惯问题：使用者在使用过程中，常常习惯与在某个固定的范围之内，例如随机数表的第1页，去选取初始值，这会大大降低所获取随机数字的随机性。

3.排列顺序的确定性：不管随机数表有多大，一旦初始值确定，则后面的数字均是可以预测的。因为这些数字都是预先固定好的，这大大降低了所选数字的随机性。

4.应用范围有限：随机数表多用于手工获取随机数字的场合，对于需要自动获取的场合，例如使用程序调用等，则会受到很大的限制。

5.实际应用中的问题：例如随机数字的保存、使用、核对和检验等，均受到限制。

6.其他问题：例如可能因为印刷问题导致错误等。

使用本研究中的基于Intel RNG的真随机数生成器基于硬件发生器源，所生成的随机数具有不可预测性，独立性和分布的均匀性。而且由于基于Intel 810以上芯片组的个人电脑占有非常大的市场份额，因此本随机数生成器可望得到广泛的应用。本方法使用简单方便，可以克服随机数表法可能存在的上述问题，完全可以满足生物医学领域对于随机数字的需要，具有一定的实用性。

(责任编辑：段咏慧)

参考文献：

[1] Knuth DE. 苏运霖译. 计算机程序设计艺术第二卷：半数值算法 [M]. 第三版. 北京：国防工业出版社, 2002. 36-67.

[2] 杨 波. 现代密码学[M]. 北京：清华大学出版社, 2003. 128-9.

[3] Gary M, John V. 使您的软件运行起来：消除偏差[URL]. http://www-900.ibm.com/developerWorks/cn/security/beating/index.shtml

[4] Schneier B, 吴世忠, 祝世雄, 张文证, 译. 应用密码学－协议、算法与C源程序[M]. 第二版. 北京：机械工业出版社, 2001. 302.

[5] Horowitz P, Hill W. The Art of Electronics[M]. Cambridge: Cam-bridge University Press, 1980. 122-43.

[6] Benjamin J, Paul K. Cryptography Research, Inc. White paper pre-pared for Intel corporation [URL]. http://www.cryptography.com/

[7] Intel Inc. Intel 82802 firmware hub: random number generator [URL]. http://www.intel.com

[8] Intel Security Driver(ISD) download page[URL]. http://developer.intel.com/design/software/drivers/platform/3463/isecdrv_enu.htm

[9] Intel Platform Security Division. The Intel Random Number Genera-tor[URL]. http://www.intel.com/design/security/rng/rng.htm

[10] DIEHARD Test: Marsaglia, George. DIEHARD Statistical Tests, Florida State University[URL]. http://stat.fsu.edu/~geo/diehard.html

[11] NIST Official website[URL]. http://www.nist.gov/public_affairs/ re-leases/digsigst.htm

[12] NIST. Security Requirements for Cryptographic Modules, Federal Information Processing Standards Publication 140-1[M]. Virginia: National Technical Information Service, 1994. 23-7.

[13] Intel: Intel Trusted Computing: Integrated Security that Starts as the Platform level[URL]. http://www.intel.com/home/scenes/stories/ trustedcomputing.htm

[14] 四川医学院. 卫生统计学[M]. 北京：人民卫生出版社, 1978：215-20.

[15] 郭祖超. 医用数理统计方法 [M]. 第二版. 北京：人民卫生出版社, 1987. 39.

参考文献：

[1] Knuth DE. 苏运霖译. 计算机程序设计艺术第二卷：半数值算法 [M]. 第三版. 北京：国防工业出版社, 2002. 36-67.

[2] 杨 波. 现代密码学[M]. 北京：清华大学出版社, 2003. 128-9.

[3] Gary M, John V. 使您的软件运行起来：消除偏差[URL]. http://www-900.ibm.com/developerWorks/cn/security/beating/index.shtml

[4] Schneier B, 吴世忠, 祝世雄, 张文证, 译. 应用密码学－协议、算法与C源程序[M]. 第二版. 北京：机械工业出版社, 2001. 302.

[5] Horowitz P, Hill W. The Art of Electronics[M]. Cambridge: Cam-bridge University Press, 1980. 122-43.

[6] Benjamin J, Paul K. Cryptography Research, Inc. White paper pre-pared for Intel corporation [URL]. http://www.cryptography.com/

[7] Intel Inc. Intel 82802 firmware hub: random number generator [URL]. http://www.intel.com

[8] Intel Security Driver(ISD) download page[URL]. http://developer.intel.com/design/software/drivers/platform/3463/isecdrv_enu.htm

[9] Intel Platform Security Division. The Intel Random Number Genera-tor[URL].

http://www.intel.com/design/security/rng/rng.htm
[10] DIEHARD Test: Marsaglia, George. DIEHARD Statistical Tests, Florida State University[URL]. http://stat.fsu.edu/~geo/diehard.html
[11] NIST Official website[URL]. http://www.nist.gov/public_affairs/ re-leases/digsigst.htm
[12] NIST. Security Requirements for Cryptographic Modules, Federal Information Processing Standards Publication 140-1[M]. Virginia: National Technical Information Service, 1994. 23-7.
[13] Intel: Intel Trusted Computing: Integrated Security that Starts as the Platform level[URL]. http://www.intel.com/home/scenes/stories/ trustedcomputing.htm
[14] 四川医学院. 卫生统计学[M]. 北京: 人民卫生出版社, 1978: 215-20.
[15] 郭祖超. 医用数理统计方法 [M]. 第二版. 北京: 人民卫生出版社, 1987. 39.

回结果列表