基于广义信息域离散轨迹变换的随机数生成器*

张国基[1)] 李璇[2)†] 刘清[2)] 张夏衍[1)]

1) (华南理工大学理学院, 广州 510640)

2) (华南理工大学计算机科学与工程学院, 广州 510640)

(2011 年 6 月 28 日收到; 2011 年 7 月 13 日收到修改稿)

广义信息域是所有可表示为二进制编码的数字信息构成的空间. 本文提出一种基于广义信息域离散轨迹变换的随机数生成器. 该生成器将广义信息域作为熵源空间, 把用户选择的数字信息作为熵源输出, 在对熵源输出进行重构处理的基础上使用离散轨迹变换方法生成随机数. 本文提出的生成器在平衡度、周期和抗碰撞等性能上均表现优良, 并通过美国国家标准技术研究院测试证明其具有理想的随机性, 可以供用户快速方便地生成高安全随机数.

关键词: 广义信息域, 随机数生成器, 熵源, 随机性

PACS: 05.40.−a, 05.40.Fb

1 引 言

随机数在信息安全领域中有着重要应用, 密钥生成、数字签名、身份认证以及密码协议等众多安全技术都需要用到随机数. 随机数生成器分为伪随机数生成器 (PRNG) 和真随机数生成器 (TRNG). PRNG 通过确定性算法作用于初始值 (称为种子) 迭代产生随机数. 目前研究最多的 PRNG 主要基于线性反馈移位寄存器 [1−3]、线性同余生成器 [4,5] 和混沌系统 [6−8]. TRNG 是一类建立在熵源基础上的随机数生成器, 熵源是指产生不确定输出的部件、设备或事件. TRNG 通过对熵源的输出进行捕获或处理生成包含熵的比特串 [9]. 随着计算机运算能力的提高, PRNG 被破解的可能性也不断增加, 在安全性要求较高的应用中 (如密码学领域) 采用 TRNG 通常比 PRNG 更加安全 [10].

物理熵源是 TRNG 常用的一类熵源, 包括电路或电子元件的热噪声 [11], 电子振荡器的频率抖动 [12], 光反馈半导体激光器产生的混沌激光 [13−15] 等. 基于物理熵源的 TRNG 先要采集熵源输出的信息, 再经过模数转换等处理后提取随机数 [14]. 由于这个过程通常需要使用额外的电路 [11,12] 或专用设备, 如半导体激光器、模数转换器、光纤反射镜等 [13−15], 从成本和便利性两方面阻碍了物理 TRNG 的应用 [10]. 另一类 TRNG 典型熵源是生物的随机行为. 廖晓峰教授的研究小组设计了基于鼠标移动 [10,16]、数码照片拍摄 [17] 等用户行为熵源的 TRNG, 使用混沌图像加密算法或混沌 hash 函数对鼠标轨迹图片及数码照片进行后处理, 以消除用户行为相似性的影响, 同时根据后处理方法的特点设计了不同的 2D/1D 转换方式产生随机数. 该类 TRNG 成本低且使用方便, 但其熵源空间较小, 产生随机数的速度较慢, 同时 2D/1D 转换方式也限制了其生成随机数的长度.

广义信息域 (GID) 是所有可表示为二进制编码的数字信息 (DI) 构成的空间, 包括了计算机、手机、U 盘等电子设备中存储的 DI、互联网上传播的 DI、以及对物理熵源输出采取模数转化后得到的 DI, 如各类视频、音频、图像、文档等. 用户在 GID 中任意选择某个 DI 是极其随机的行为, 可看作不确定熵源事件. 本文提出一种基于广义信息域离散轨迹变换的随机数生成器 (RGDG), GID 是其熵源空间, DI 为熵源输出. 使用 GID 的主要优点在于: 熵源空间大; 易于获取和处理, 用户无须添加成本购买额外的设备; 熵源输出要经过用户的参与和控制, 用户使用时会更加放心. 但是用户选择 DI 的任意性也导致 DI 可能存在短长度、弱随机统计特性、相似性或重复性的问题. 所以 RGDG

* 广东省自然科学基金 (批准号: 8151064101000033) 资助的课题.

† E-mail: l.xuan01@mail.scut.edu.cn

© 2012 中国物理学会 Chinese Physical Society

在对 DI 重构处理的基础上采取离散轨迹变换方法产生随机数, 有效地弥补了熵源可能存在的缺陷, 也保证了优良的平衡度、周期、抗碰撞性和速度, 并通过了美国国家标准技术研究院 (NIST) 随机性测试.

2 RGDG 原理

RGDG 生成随机数的原理主要包括两部分. 首先对用户选择的 DI 进行多轮重构处理, 将其扩展为具有一定规模且编码出现频次均匀的数据空间; 再通过带反馈机理的地址变换映射实现数据空间上的离散轨迹变换, 析出随机数.

2.1 重构处理

为了消除熵源可能存在的短长度和弱随机统计特性等缺陷, 需对用户选择的 DI 进行重构处理, 生成新的数据空间, 记作 $\boldsymbol{BG}$.

设第 i 轮重构变换的结果为 $\boldsymbol{PB}_i$, 每轮重构变换的参数包括重构初始位置 $m_i(m_i \in \mathbb{Z}^+)$ 和重构长度 $l_i(l_i \in \mathbb{Z}^+)$ $(i = 1, 2, \cdots, S)$. DI 记作 $\boldsymbol{PB}_0 = (b_0^0, b_1^0, \cdots, b_{l_0-1}^0)$, l_0 是 DI 的长度 (以字节为单位). DI 经过 S 轮重构变换得到 $\boldsymbol{PB}_S$ 即 $\boldsymbol{BG}$, 如公式 (1):

$$\begin{aligned}\boldsymbol{PB}_i =&(b_0^i, b_1^i, \cdots, b_{l_i-1}^i) = g(\boldsymbol{PB}_{i-1}, m_i, l_i),\\ &i = 1, 2, \cdots, S.\end{aligned} \tag{1}$$

其中, 函数 g 表示 $\boldsymbol{PB}_i$ 的生成过程, 具体如公式 (2) 和 (3) 所示: 设置计数器向量 $\boldsymbol{c} = (c_0, c_1, \cdots, c_{255})$, c_k 记录重构过程中编码 $k \in \{0, 1, \cdots, 255\}$ 的出现频次, $\boldsymbol{c}_{\min} = \min\{c_0, c_1, \cdots, c_{255}\}$. 每轮重构前将向量 $\boldsymbol{c}$ 初始化为零向量. $\boldsymbol{PB}_i$ 的生成过程用 (2) 和 (3) 式表示:

$$b'_j = RL(b^{i-1}_{(m_i+j) \bmod l_{i-1}}, \lfloor (m_i + j)/l_{i-1} \rfloor), \tag{2}$$

$$b_j^i = \begin{cases} b'_j & c_{b'_j} - c_{\min} < q, \\ \min\{k | c_k = c_{\min}\} & c_{b'_j} - c_{\min} \geqslant q, \end{cases} \tag{3}$$

其中, (2) 式中的 RL 表示循环左移的操作, 是通过将 $b^{i-1}_{(m_i+j) \bmod l_{i-1}}$ 循环左移 $\lfloor (m_i + j)/l_{i-1} \rfloor$ 位得到 b'_j $(j = 0, 1, \cdots, l_i - 1)$. (3) 式中 q 为约束参数, $q \in \mathbb{Z}^+$.

下面以第 i 轮重构变换为例, 说明生成 $\boldsymbol{PB}_i$ 的具体步骤.

1) 根据第 i 轮的重构初始位置 m_i 定位至 $\boldsymbol{PB}_{i-1}$ 中的字节 $b^{i-1}_{m_i \bmod l_{i-1}}$; 用 j 记录该轮已重构长度, 令 $j = 0$.

2) 对当前字节 $b^{i-1}_{(m_i+j) \bmod l_{i-1}}$ 循环左移 $\lfloor (m_i + j)/l_{i-1} \rfloor$ 位得到 b'_j. 按照公式 (3), 判断 b'_j 在该轮重构中的当前出现频次是否满足 $c_{b'_j} - c_{\min} < q$: 若满足, 将 b'_j 加入 $\boldsymbol{PB}_i$ 即 $b_j^i = b'_j$; 否则, 将 0—255 中出现频次最少的字节编码 $\{k | c_k = c_{\min}\}$ 加入 $\boldsymbol{PB}_i$, 当 $\{k | c_k = c_{\min}\}$ 包含多个元素时, 选取编码最小的元素即 $b_j^i = \min\{k | c_k = c_{\min}\}$. 所以, 每次加入 $\boldsymbol{PB}_i$ 的字节编码 b_j^i 均满足约束条件 (a): $c_{b_j^i} - c_{\min} < q$.

3) 加入 $\boldsymbol{PB}_i$ 的字节编码所对应的计数器分量加 1, 并令 $j = j + 1$.

4) 定位至 $\boldsymbol{PB}_{i-1}$ 的下一个字节 $b^{i-1}_{(m_i+j) \bmod l_{i-1}}$. 当定位地址到达 $\boldsymbol{PB}_{i-1}$ 最后一个字节后, 通过 mod 操作重新定位至 $\boldsymbol{PB}_{i-1}$ 的首字节, 即将 $\boldsymbol{PB}_{i-1}$ 看作一个首尾相连的环状结构, 此时如公式 (2) 中 $\lfloor (m_i + j)/l_{i-1} \rfloor$ 所示, 循环左移位数也增加 1 位.

5) 返回步骤 2), 直至 $\boldsymbol{PB}_i$ 达到长度要求.

最后一轮得到的重构结果 $\boldsymbol{PB}_S$ 作为析出随机数的数据空间 $\boldsymbol{BG}$.

重构过程中设置约束条件 (a) 是为了控制 $\boldsymbol{PB}_i$ 中所有字节编码的出现频次均匀, 使得生成的数据空间具有较好的统计分布特性. 约束参数 q 越小, 约束条件越强, $\boldsymbol{PB}_i$ 中编码分布越均匀. 可证明: 当 $l_i > 255 \times q$ 时, $\boldsymbol{PB}_i$ 中 $\{0, 1, \cdots, 255\}$ 各字节编码皆存在且任意两个编码的频次之差不大于q.

定理 1 当重构长度 $l_i > 255 \times q$ 时, $\boldsymbol{PB}_i$ 中 $\{0, 1, \cdots, 255\}$ 各字节编码皆存在且任意两个编码的频次之差不大于 q.

证明 由步骤 2), 每次加入 $\boldsymbol{PB}_i$ 的字节编码 b_j^i 均需满足约束条件 (a), 所以 $\boldsymbol{PB}_i$ 中任意两个字节编码的频次之差不大于 q.

反证法, 假设 $\exists k_0 \in \{0, 1, \cdots, 255\}$ 且 $k_0 \notin \boldsymbol{PB}_i$, 则 $\boldsymbol{PB}_i$ 中出现的编码种数 $n_i \leqslant 255$ 且 $c_{\min} = \min\{c_0, c_1, \cdots, c_{255}\} = c_{k_0} = 0$, $\boldsymbol{PB}_i$ 中所有编码的平均出现频次为 $\dfrac{l_i}{n_i}$. 记频次最高的编码为 k_1, 则 $c_{k_1} = c_{\max} \geqslant \dfrac{l_i}{n_i}$. 当重构长度 $l_i > 255 \times q$ 时, 有 $c_{k_1} - c_{\min} \geqslant \frac{l_i}{n_i} - 0 > \dfrac{255 \times q}{255} - 0 = q$, 这与约束条件 (a) 矛盾. 所以关于 $k_0 \notin \boldsymbol{PB}_i$ 的假设不成立, 即 $\boldsymbol{PB}_i$ 中 $\{0, 1, \cdots, 255\}$ 各字节编码皆

存在. 定理 1 得证.

2.2 随机数析出

RGDG 通过对 DI 的重构处理生成具有新的长度与分布的数据空间 $\boldsymbol{BG}$, 初步消除了对 DI 大小及内容的依赖性. 在此基础上, RGDG 再通过 $\boldsymbol{BG}$ 上的离散轨迹变换析出性能优良的随机数.

算法在每次地址变换时析出 4 bit 数据, 设置计数器向量 $\boldsymbol{c}' = (c'_0, c'_1, \cdots, c'_{15})$, c'_k 记录生成随机数的过程中编码 $k \in \{0, 1, \cdots, 15\}$ 的出现频次. $\boldsymbol{c}'$ 初始化为零向量, 令 $\boldsymbol{c}'_{\min} = \min\{c'_0, c'_1, \cdots, c'_{15}\}$. 另外, 每拼入 4 bit 数据至随机数时需将计数器向量重新设置为 $\boldsymbol{c}' = (c'_0 - c'_{\min}, c'_1 - c'_{\min}, \cdots, c'_{15} - c'_{\min})$, 称为计数器向量的清零设置, 该操作是为了方便计数器向量的比较.

RGDG 析出随机数的具体步骤如下.

1) 设置初始参数 $IV = (t, r, u)$: t为系统时间戳, r 为系统随机数, t 与 r 由系统自动生成, 长度为 4 字节; u 是用户任意输入的个性化信息.

2) IV 规格化: 选择低碰撞不可逆的 Hash 函数将初始参数 $IV = (t, r, u)$ 映射为 160 bit 的二进制位串, 按 32 bit 的长度将其分割成 5 维向量, 记作 $\boldsymbol{AI} = (x_0, x_1, \cdots, x_4)$.

3) 使用 $\boldsymbol{AI}$ 的函数计算地址 ma (按位):

$$ma = \left[\sum_{i=1}^{4}\left(a_i \prod_{j=0}^{i-1} d_j\right) + a_0\right] \bmod (8l_S), \quad (4)$$

其中 $a_i = x_i \bmod d_i$, $\boldsymbol{D} = (d_0, d_1, \cdots, d_4)$ 是地址变换参数, $d_i \in \mathbb{Z}^+ (i = 0, 1, \cdots, 4)$.

4) 随机数析出: 从 $\boldsymbol{BG}$ 的第 ma 位开始, 顺序析出 4 bit 数据, 记为 M. 若 M 不满足约束条件 (b): $c'_M - c'_{\min} < q$, 则析出地址前移 1 位即令 $ma = ma + 1$, 重新析出 4 bit 数据进行约束判断, 重复此操作直至得到满足约束条件 (b) 的$\boldsymbol{M}$. 此处 q 即为重构处理时的约束参数 q.

同时, 建立堆栈存储计数器向量: 在 RGDG 开始运行后, 每生成 1 kb 随机数时将当前计数器向量 $\boldsymbol{c}' = (c'_0, c'_1, \cdots, c'_{15})$ 存入堆栈, 如图 1 所示, 即每到 kb 节点处存储当前的 $\boldsymbol{c}'$. 设置堆栈大小为 100 × 16 字节, 可存储 100 个节点. 当存储节点数超过 100 时, 将最早加入堆栈的节点计数器向量进行出栈处理.

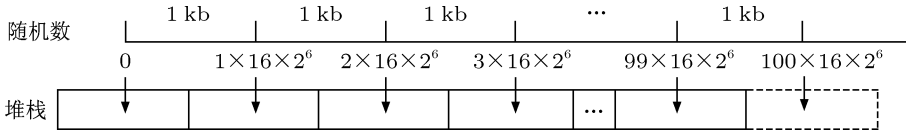

图 1 计数器向量的堆栈存储图

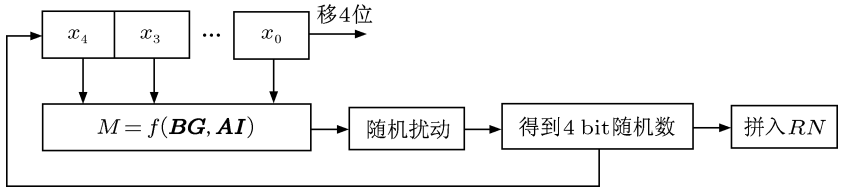

图 2 离散轨迹变换生成随机数的流程图

每次得到满足约束条件 (b) 的 M 时, 将计数器向量 $(c'_0, \cdots, c'_{M-1}, c'_M + 1, c'_{M+1}, \cdots c'_{15})$ 依次与存储在堆栈中的节点计数器向量比较: 若堆栈中不存在与该向量相等的节点, 直接将 M 拼入随机数; 否则, 随机选择 $\{rn_x \,|rn_x \in \{0, 1, \cdots, 15\} \cap rn_x \neq M\}$ 替换 M 拼入随机数. 对应的计数器分量加 1.

约束条件 (b) 能够保证随机数具有优良的平衡度, 同时通过计数器向量的比较避免了小周期的存在. 第 3 节对生成器的平衡度和周期进行了详细分析, 说明序列具有良好的平衡度且周期长度大于 100 kb.

5) 产生新的 $\boldsymbol{AI}$: $\boldsymbol{AI} = (x_0, x_1, \cdots, x_4)$ 看作 160 bit 的位串, 左移 4 位后再将最后 4 位用步骤 4) 中拼入随机数的 4 bit 数据替换, 得到新的 $\boldsymbol{AI}$. 若随机数长度已足够, 结束; 否则返回步骤 3).

如上述步骤, 本文算法使用 $\boldsymbol{AI}$ 的函数计算变换地址, 根据该地址从 $\boldsymbol{BG}$ 中析出 4 bit 数据 M, 记为 $M = f(\boldsymbol{BG}, \boldsymbol{AI})$. 另外, 通过设置约束条件 (b) 和计数器比较换码的方法实现对 M 的随机扰动,

得到 4 bit 数据, 拼入随机数 RN. 同时, 引入反馈机理, 让得到的 4 bit 随机数参与新 $\boldsymbol{AI}$ 的产生, 再通过新 $\boldsymbol{AI}$ 计算下一步的地址. RGDG 结合上述步骤最终实现了数据空间 $\boldsymbol{BG}$ 上的离散轨迹变换, 生成随机数. 基本流程如图 2 所示.

离散轨迹变换方法与分块扫描或计算距离夹角等 2D/1D 转换方式 [10,16,17] 相比, 其主要优点在于: 采用地址变换析出随机数, 使得随机数长度不受数据空间 $\boldsymbol{BG}$ 的大小限制; 通过设置约束条件及计数器比较换码的方法保证生成的随机数具有优良的平衡度和较长的周期; 初始参数 $IV=(t,r,u)$ 中引入系统时间戳 t 等非确定熵源, 结合反馈机理增强了生成器的抗碰撞性, 消除了 DI 相似性或重复性的影响.

2.3 参数说明

由于本文算法的参数较多, 本小节详细说明算法所使用参数的意义及建议的取值范围. RGDG 的参数包括: 重构轮数 S, 约束参数 q, 重构初始位置 m_i, 重构长度 $l_i(i=1,2,\cdots,S)$, 初始参数 IV 和地址变换参数 $\boldsymbol{D}$.

重构轮数 S: 一般对于用户选择的 DI, 重构 1 或 2 轮就可以得到性能良好的随机数. 但当 DI 的长度很短或统计特性较差时, 只采用 1 或 2 轮重构变换得到的结果还是存在着一定的规律性, 会导致随机数的性能不佳. 所以, 为了保证 RGDG 对于用户任意选择的 DI 都能适用, 建议对 DI 进行多轮重构. 选取多个不同大小、内容与格式的数字信息 DI 进行实验, 包括小于 1 k 的文档, 发现重构 3 轮以上即 $S\geqslant 3$ 时, 均能够生成性能优良的随机数. 但重构轮数增加时, 算法速度略有减慢.

约束参数 q: 重构处理时约束条件 (a) 和随机数析出时约束条件 (b) 均使用了约束参数 $q\in\mathbb{Z}^+$, 这两个约束分别控制重构过程及随机数析出过程中各编码出现频次均匀, 使生成的随机数具有良好的随机特性如平衡度等. 由约束条件 (a) 和 (b) 可见, 约束参数 q 越小, 约束条件越强, 各编码的出现频次越均匀. 实验发现: $q<3$ 时, 生成的随机数在 NIST 测试中的通过率只有 40%, 这是由于约束条件对算法的控制过强, 使得生成的随机数具有了一些固定模式; $3\leqslant q\leqslant 14$ 时随机数在 NIST 测试中的通过率高于 96%; $q>14$ 时随机数在 NIST 测试中的通过率低于 90%, 可见 q 的最佳取值范围为 $3\leqslant q\leqslant 14$.

重构初始位置 m_i: 如 2.1 节所述, 第 i 轮重构时根据该轮的重构初始位置 m_i 定位至 $\boldsymbol{PB}_{i-1}$ 中的字节 $b^{i-1}_{m_i \bmod l_{i-1}}$. 算法对参数 m_i 具有鲁棒性, 其取值不影响算法性能, 可由用户任意选定.

重构长度 l_i: 由定理 1, 为保证重构结果 $\boldsymbol{PB}_S$ 中 $\{0,1,\cdots,255\}$ 各字节编码皆存在且任意两个编码的频次之差不大于约束参数 q, 重构的最小长度要求为 $l_i>255\times q$. 实验发现, 在达到最小长度要求的前提下, 重构长度 l_i 还与用户要求的随机数长度 $l_{\rm RN}$ 相关, 当 $l_S=l_{\rm RN}$ 且 $l_{i+1}=2l_i$ 时, 对于用户要求的任意 $l_{\rm RN}$, RGDG 均能通过 NIST 测试.

初始参数 IV: $IV=(t,r,u)$ 包括系统时间戳 t, 系统随机数 r 和用户任意输入的个性化信息 u. 其中, t 表示用户运行生成器的当前时刻, r 是系统自动生成的随机数, t 与 r 在每次运行发生器时自动更新, 无需用户参与. u 在每次运行发生器时提供给用户输入, 没有大小或内容的要求, 作为用户的个性化信息, 其取值不影响算法性能.

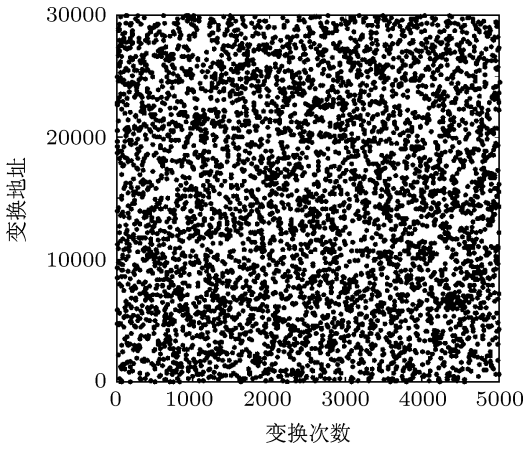

图 3 离散轨迹变换得到的变换地址 $\boldsymbol{BG}$ 长度 =30000 bit

地址变换参数 $\boldsymbol{D}$: $\boldsymbol{D}=(d_0,d_1,\cdots,d_4)$ 是地址变换函数的参数, 在调用公式 (4) $ma=\left[\sum\limits_{i=1}^{4}\left(a_i\prod\limits_{j=0}^{i-1}d_j\right)+a_0\right]\bmod(8l_S)$ 计算变换地址 ma 时用到, 式中 $a_i=x_i \bmod d_i(i=0,1,\cdots,4)$, x_i 是 $\boldsymbol{AI}$ 的分量. $\boldsymbol{D}$ 和 l_S 确定时, 地址变换函数可看作 $(a_0,a_1,\cdots,a_4)$ 的函数. 由于 a_i 是 x_i 对 d_i 取模的结果, 地址变换参数 d_i 的取值决定了 a_i 变量的变化范围. d_i 越大, a_i 的取值范围越大, ma 的变化空间越大. 选取区间在 [100, 1000], [1000, 5000], [5000, 10000], [10000, 50000] 内的不同 d_i 进行分组实验, 每组实验在长度 $l_S\in[5\times10^3,5\times10^6]$ 的多个 BG 上计算变换地址ma. 实验发现, 当 $\min\{d_i\}\geqslant 1000$ 时, ma 在不同规模 BG 上的

轨迹点均呈现出较好的离散度, 生成的随机数在 NIST 测试中表现优良. 图 3 显示了当地址变换参数 $\boldsymbol{D}=(5678,5433,12443,3446,44089)$, $\boldsymbol{BG}$ 长度 $l_S=3\times10^4$ 时由公式 (4) 计算得到的 5000 个变换地址在 $\boldsymbol{BG}$ 上的分布情况.

3 性能分析

本节对 RGDG 算法的随机性及安全性进行验证, 包括平衡度分析、周期分析、NIST 测试、抗碰撞分析及速度分析. 限于篇幅, 此节仅在 NIST 测试中报告多组参数取值下的实验结果, 其他实验只报告固定参数取值下的实验结果. 有关算法参数的说明可参照 2.3 节. 本节的具体参数设置如下:

重构轮数 $S=3$, 重构初始位置 $m=(4789,6892,94)$, 重构长度 $l=(\max(\lfloor l_{\rm RN}/4\rfloor,256\times12\times5),\max(\lfloor l_{\rm RN}/2\rfloor,256\times12\times10),\max(l_{\rm RN},256\times12\times20))$, 约束参数 $q=12$, IV 中用户个性化信息 u 设置为 random, 地址变换参数 $\boldsymbol{D}=(5678,5433,12443,3446,44089)$.

3.1 平衡度分析

平衡度是衡量随机数生成器性能的重要指标, 平衡度的值越小, 说明序列中 0 与 1 的个数越接近 [8]. 设随机序列的长度为 L bit, 序列中 0 与 1 的比特个数分别为 $N(0)$ 和 $N(1)$, 序列的平衡度定义为

$$E=\frac{|N(0)-N(1)|}{L}. \tag{5}$$

图 4 是 RGDG 使用 adobe 官方 flash 播放器的插件安装包 (v10.0.22.87) Install_ flash_player_ax.zip (1.83 MB) 作为 DI 时生成随机序列的平衡度随序列长度变化的实验结果, 序列长度由 5000 字节增加到 100000 字节. 实验结果显示, 序列平衡度均小于 0.014, 并且随序列长度的增加而逐渐减少.

定理 2 RGDG 生成随机数的平衡度

$$E=\frac{|N(0)-N(1)|}{L}\to0(L\to\infty).$$

证 明 已知 $\boldsymbol{c}'=(c'_0,c'_1,\cdots,c'_{15})$ 表示 $\{0,1,\cdots,15\}$ 在随机数中出现的频次,

$$\begin{aligned}&|N(0)-N(1)|\\=&\big|4c'_0+2c'_1+2c'_2+2c'_4+2c'_8+(-2)c'_7+(-2)c'_{11}\\&+(-2)c'_{13}+(-2)c'_{14}+(-4)c'_{15}\big|\\\leqslant&\,4\,|c'_0-c'_{15}|+2\,|c'_1-c'_{14}|+2\,|c'_2-c'_{13}|\\&+2\,|c'_4-c'_{11}|+2\,|c'_8-c'_7|.\end{aligned}$$

由于 RGDG 生成随机数时满足约束条件 (b), 则 $\forall x_1,x_2\in\{0,1,\cdots,15\}$, $|c'_{x_1}-c'_{x_2}|\leqslant q$, 故有

$$\begin{aligned}&|N(0)-N(1)|\\\leqslant&\,4\,|c'_0-c'_{15}|+2\,|c'_1-c'_{14}|+2\,|c'_2-c'_{13}|\\&+2\,|c'_4-c'_{11}|+2\,|c'_8-c'_7|\leqslant12q.\end{aligned}$$

所以当 $L\to\infty$ 时,

$$E=\frac{|N(0)-N(1)|}{L}\leqslant\frac{12q}{L}\to0.$$

图 4 的实验结果和定理 2 说明了 RGDG 生成的随机数具有良好的平衡度.

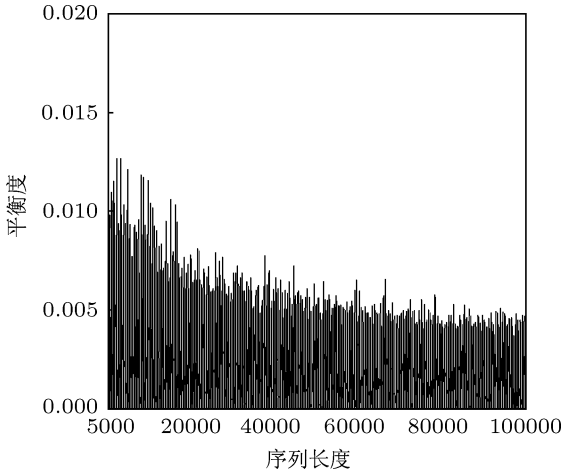

图 4 平衡度随序列长度的变化图

3.2 周期分析

随机数的周期长度是保证其安全性的重要因素, 短周期随机数的安全性低, 不适合应用于密码学等对安全性要求较高的领域.

RGDG 生成随机数时满足约束条件 (b), 则 $\forall x_1,x_2\in\{0,1,\cdots,15\}$, $|c'_{x_1}-c'_{x_2}|\leqslant q$. 假设 RGDG 序列存在周期, 若周期内各编码数据非等频次出现, 则一定存在编码 k 与 p 使得 $c'_p-c'_k>0$. 设 $c'_p-c'_k=a\in\mathbb{Z}^+$, 当周期循环次数超过 $\lceil q/a\rceil$ 时, $c'_p-c'_k>a\times\lceil q/a\rceil\geqslant q$, 这与约束条件 (b) 矛盾. 故周期内 $\{0,1,\cdots,15\}$ 各编码的出现频次一定相等. 不妨设 RGDG 序列的周期为 $P\in\mathbb{Z}^+$, 则根据计数器向量的清零设置, 相隔P bit 的计数器向量 $\boldsymbol{c}'=(c'_0,c'_1,\cdots,c'_{15})$ 必定相等.

假设 $P\leqslant100\,\text{kb}=100\times2^{10}$ bit, 已证明周期内 $\{0,1,\cdots,15\}$ 各编码的出现频次相等, 故 $P\in\{h\times2^6\,|\,h=1,2,\cdots,16\times100\}$. 如 2.2 节

所述, RGDG 设置了 100×16 字节的堆栈, 保存 100 个节点的计数器向量, 每次将 4 bit 数据加入随机数前均需将当前计数器向量与堆栈中的节点计数器向量进行比较, 一旦相等就进行随机换码, 以避免随机数出现 $P\in\{h\ \times2^6\,|\,h=1,2,\cdots,16\times100\}$ 的周期. 所以 RGDG 通过计数器向量的比较策略, 排除了周期长度 $P\leqslant100$ kb 的可能性. 另外, 增大堆栈的容量能增加保存的计数器向量个数, 加大计数器比较的距离, 以克服随机数中可能存在的更长周期.

综上, RGDG 通过约束条件 (b) 及计数器向量的比较策略保证了随机数的周期长度 $P>100$ kb.

3.3 NIST 测试

这一小节采用 NIST 公布的随机数测试包 [18], 对 RGDG 生成的随机数进行严格的随机性测试. NIST 测试包共有 15 个测试项, 如表 1 所示.

表 1 NIST 测试包

编号	测试项的名称	测试项的目的
1	频数测试	0 的个数与 1 的个数是否均衡
2	块内频数测试	M 位块中 0 与 1 的个数是否均衡
3	累加测试	0,1 转化为 −1,1, 计算累加和检验 0, 1 的分布情况
4	游程测试	不同长度游程的个数
5	块内最长游程测试	M 位块中最长 1 游程的分布
6	矩阵秩测试	固定长度子序列的线性相关度
7	离散傅里叶测试	离散傅里叶变换后频谱尖峰的分布, 测试周期特征
8	非重叠模块测试	非重叠方式匹配给定模块的次数
9	重叠模块测试	重叠方式匹配给定模块的次数
10	通用统计测试	是否能在信息不丢失的情况下被明显压缩
11	近似熵测试	M 位与 $M+1$ 位可重叠序列模式的熵的差的分布
12	随机游动测试	每一个 “0- 回归” 模式中各种状态出现的次数
13	随机游动变量测试	某一个状态在所有 “0- 回归” 模式中出现的总次数
14	连续性测试	M 位可重叠子序列的每一种模式的个数是否相等
15	线性复杂度测试	序列的线性复杂度

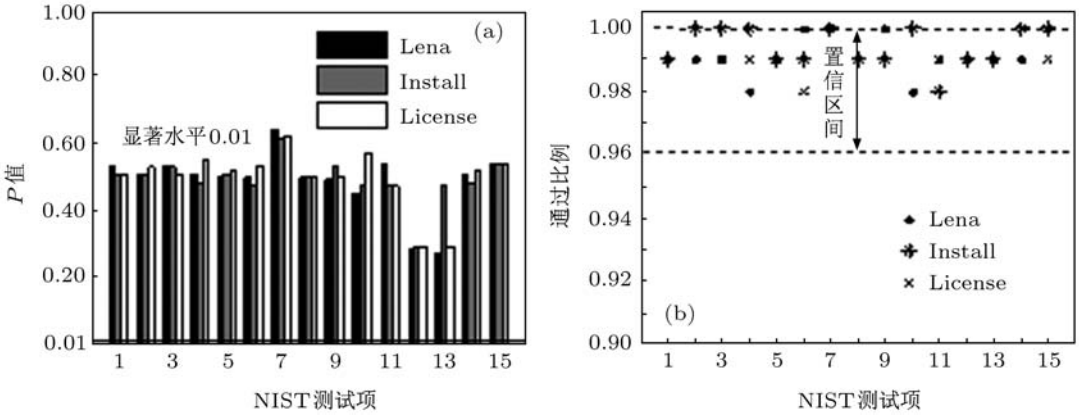

图 5 NIST 测试结果图 (a) p 值; (b) 通过比例

NIST 的测试结论是统计意义上的, 统计假设为 “测试序列是随机的”. 每一项测试均需计算 P 值及通过比例. P 值高于显著水平 α 时接受统计假设, 显著水平 α 通常取为 0.01. 通过比例在置信区间内则说明该生成器通过了 NIST 测试, 置信区间与参与测试的随机数个数相关.

为了说明 RGDG 的性能不受 DI 大小、内容及格式的影响, 这里分别使用 adobe 官方 flash 播放器的插件安装包 (v10.0.22.87) Install_flash_player_ax.zip(1.83 MB), 512×512 的标准图像 Lena.jpg (43 kb) 以及 matlab 软件包 (v7.6.0.324) 的文本文件 License.txt (73 kb) 作为用户所选 DI, 应用 RGDG 各生成 100 条 800000 字节的随机数进行 NIST 检测, 3 轮重构长度为 $l=(2\times10^5,4\times10^5,8\times10^5)$. 测试中通过比例的置信区间为 [0.96,1]. 图 5 显示了 NIST 测

试的结果: RGDG 使用 3 个 DI 生成的 100 条随机数在 NIST 测试包的 15 个测试项中平均 P 值都明显大于显著水平 0.01, 通过比例均落在置信区间内且大于 0.98. 测试结果表明 RGDG 通过了严格的 NIST 测试, 具有理想的随机性. 其中, Frequency Test (测试项 1) 和 FFT Test (测试项 7) 的 P 值均高于 0.5, 通过比例分别为 0.99 和 1.00, 这与 3.1 及 3.2 节中对 RGDG 平衡度和周期的分析结果相符合.

另外, 为了验证 RGDG 对算法参数的鲁棒性, 参照 2.3 节中建议的参数取值范围, 任意选取 4 组参数, 对 RGDG 进行上述的 NIST 测试. DI 选取 Lena.jpg. 表 2 给出了每组参数下的 RGDG 在 15 个 NIST 测试项中的最低通过比例 min-prop、最高通过比例 max-prop 和平均通过比例 average-prop. 实验结果说明 4 组参数设置下的 RGDG 在所有 15 项测试中的通过比例均落在 [0.96,1] 的置信区间内, 通过了 NIST 测试, RGDG 表现出了对算法参数的良好鲁棒性.

表 2 RGDG 在各组参数下的 NIST 测试结果

参数组	(1)	(2)	(3)	(4)
S	3	3	4	4
m	(12, 42, 8)	(238, 597, 636)	(25, 69, 547, 2835)	(4565, 5, 34, 144)
l	(2, 4, 8)×10^5	(2, 4, 8)×10^5	(1, 2, 4, 8)×10^5	(1, 2, 4, 8)×10^5
q	3	8	10	14
u	User	Test	—	1234
D	(4678, 5433, 3553, 3446, 1089)	(5287, 7530, 9306, 12513, 2312)	(13151, 34234, 2456, 9105, 1409)	(42457, 17340, 3961, 1285, 15609)
min-prop	0.96	0.97	0.97	0.97
max-prop	1	1	1	1
average-prop	0.98	0.98	0.99	0.99

3.4 抗碰撞分析

抗碰撞性是保障随机数生成器在密码学应用中安全性的重要因素, 可以通过计算随机数之间的绝对距离来衡量. 假设 RN_1 和 RN_2 是长度为 N 字节的随机数, 则 RN_1 和 RN_2 之间的绝对距离定义为

$$d = \frac{\sum_{i=1}^{N} |t(e_i) - t(e_i')|}{N}, \tag{6}$$

其中, e_i 和 e_i' 分别表示 RN_1 和 RN_2 的第 i 字节, 函数 t 将 e_i 和 e_i' 转化为十进制数. 两条独立无碰撞的随机数之间的绝对距离为 $E(d) = \frac{1}{3} \times 256 \approx 85.33$. 生成随机数的绝对距离越接近理想值 85.33, 说明生成器的抗碰撞性越好.

RGDG 引入系统时间戳 t 等非确定熵源作为初始参数, 并采用包含反馈机理的离散轨迹变换方法使得初始参数的差异在变换过程中不断累积, 以增强系统的抗碰撞性, 消除 DI 相似性或重复性的影响. 这里使用 Install_flash_player_ax.zip, Lena.jpg 及 License.txt 作为 DI 进行分组实验: 每组运行 RGDG 生成 100 条 1000 字节的随机数, 计算组内绝对距离及组间绝对距离, 得到最大值、最小值及平均值, 如表 3 所示. 结果表明, DI 相同或不同的情况下, RGDG 生成随机数之间的绝对距离的平均值都非常接近理想值 85.33, 具有优良的抗碰撞性.

表 3 抗碰撞性实验

		Install	Lena	License
组内绝对距离	最大值	92.3640	93.3800	91.6020
	最小值	78.8710	77.5480	77.8000
	平均值	85.8213	84.6543	84.5187
		Install & Lena	Lena & License	License & Install
组间绝对距离	最大值	93.8230	92.1500	92.5240
	最小值	78.7780	77.7650	78.3600
	平均值	85.6140	84.8565	85.2464

3.5 速度分析

选取 Install_flash_player_ax.zip, Lena.jpg 及 License.txt 作为 DI, 对 RGDG 生成随机数的速度进行测试, 测试结果分别为 8.83, 9.43 和 9.41 MB/s. 可见 RGDG 的生成速度不受 DI 影响, 平均速度达到 9 MB/s, 能够满足实际应用的需求.

4 结 论

本文给出了一种基于广义信息域离散轨迹变换的随机数生成器. 该生成器通过重构处理生成新的数据空间, 并引入带反馈机理的函数实现离散轨迹变换, 克服了熵源可能存在的长度或统计特性等缺陷, 能够产生具有优良平衡度、周期及抗碰撞性的随机数, 并通过 NIST 测试证明其具有理想的随机统计特性. 本文生成器使用简单, 熵源空间大, 可以供用户快速、方便、安全地生成随机数. 目前研究小组已将该生成器应用于加密系统中, 并获得了两项专利 [19,20], 在文件加密、ftp 加密传输等方面都得到了实际的应用.

[1] Zeng G, Yang Y, Han W B, Fang S Q 2010 *J. Electron. Info. Tech.* **32** 737 (in Chinese) [曾光, 杨阳, 韩文报, 范淑琴 2010 电子与信息学报 **32** 737]

[2] Ma W J, Feng D G 2007 *J. Commun.* **28** 42 (in Chinese) [马卫局, 冯登国 2007 通信学报 **28** 42]

[3] Xiao H, Zhang C R, Xiao G Z, Wang X M 2008 *J. Commun.* **29** 210 (in Chinese) [肖鸿, 张串绒, 肖国镇, 王新梅 2008 通信学报 **29** 210]

[4] Shen H Y, Zhang P, Wang K 2009 *J. Tsinghua University* **49** 191 (in Chinese) [沈华韵, 张鹏, 王侃 2009 清华大学学报 (自然科学版) **49** 191]

[5] Raj S K, Rajesh G K, Vyasa S 2010 *IEEE Trans. Circuits Syst. II* **57** 203

[6] Fu Z J, Zeng Y C, Xu M L 2008 *Acta Phys. Sin.* **57** 4014 (in Chinese) [傅志坚, 曾以成, 徐茂林 2008 物理学报 **57** 4014]

[7] Wang S H, Li D 2010 *Chin. Phys.* B **19** 080505

[8] Zhang X F, Fan J L 2010 *Acta Phys. Sin.* **59** 2289 (in Chinese) [张雪锋, 范九伦 2010 物理学报 **59** 2289]

[9] Information Security Technology Glossary GB/T25069 2010 *Standardization Administration of the People's Republic of China* (in Chinese) [信息安全技术术语 GB/T25069 2010 国家标准化管理委员会] [Online] Available: http: //www.sac.gov.cn/gjbzgg/20104159/

[10] Zhou Q, Hu Y, Liao X F 2008 *Acta Phys. Sin.* **57** 5413 (in Chinese) [周庆, 胡月, 廖晓峰 2008 物理学报 **57** 5413]

[11] Petrie C S, Connelly J A 2000 *IEEE Trans. Circuits Syst.* **47** 615

[12] Bucci M, Germani L, Luzzi R, Trifiletti A, Varanonuovo M 2003 *IEEE Trans. Comp.* **52** 403

[13] Zhang J B, Zhang J Z, Yang Y B, Liang J S 2010 *Acta Phys. Sin.* **59** 7679 (in Chinese) [张继兵, 张建忠, 杨毅彪, 梁君生, 王云才 2010 物理学报 **59** 7679]

[14] Chen S S, Zhang J Z, Yang L Z, Liang J S, Wang Y C 2011 *Acta Phys. Sin.* **60** 010501 (in Chinese) [陈莎莎, 张建忠, 杨玲珍, 梁君生, 王云才 2011 物理学报 **60** 010501]

[15] Guo H, Liu Y, Dang A H, Wei W 2009 *Chin. Sci. Bull.* **54** 3651 (in Chinese) [郭弘, 刘钰, 党安红, 韦韦 2009 科学通报 **54** 3651]

[16] Zhou Q, Liao X F, Wong K W, Hu Y, Xiao D 2009 *Inform. Sci.* **179** 3442

[17] Zhao L, Liao X F, Xiao D, Xiang T, Zhou Q, Duan S 2009 *Chaos Soliton. Fract.* **42** 1692

[18] Special Publication 800-22 Revision 1 2008 *National Institute of Standards and Technology* [Online] Available: http://csrc.nist.gov/publications/nistpubs/800-22-rev1/SP800-22rev1.pdf

[19] Zhang G J, Liu Q, Li F M, Xu J B, Ding Z 2011 *Chinese Patent* ZL 2008 1 0198491. X (in Chinese) [张国基, 刘清, 黎凤鸣, 许洁斌, 丁卓 2011 中国发明专利 ZL 2008 1 0198491.X]

[20] Zhang G J, Xu H, Li F M, Liu Q 2011 *Chinese Patent* ZL 2008 1 0198489.2 (in Chinese) [张国基, 徐浩, 黎凤鸣, 刘清 2011 中国发明专利 ZL 2008 1 0198489.2]

Random number generator based on discrete trajectory transform in generalized information domain*

Zhang Guo-Ji[1)] Li Xuan[2)†] Liu Qing [2)] Zhang Xia-Yan[1)]

1) (*Department of Science, South China University of Technology, Guangzhou* 510640*, China*)

2) (*Department of Computer Science and Engineering, South China University of Technology, Guangzhou* 510640*, China*)

(Received 28 June 2011; revised manuscript received 13 July 2011)

Abstract

Generalized information domain is the space of all digital information that can be expressed by binary code. In this paper, a random number generator based on discrete trajectory transform in generalized information domain is proposed. The generator exploits generalized information domain as the space of entropy source, and uses the digital information selected by the user as the output of entropy source, and then utilizes the discrete trajectory transform method to generate random number based on the reconstruction of the output of entropy source. The proposed generator shows good characteristics of balance, period and anti-collision, and it demonstrates satisfactory randomness through the National Institute of Standards and Technology test. This generator can be utilized to generate high secure random number quickly and conveniently.

Keywords: generalized information domain, random number generator, entropy source, randomness

PACS: 05.40.−a, 05.40.Fb

* Project supported by the Natural Science Foundation of Guangdong Province, China (Grant No. 8151064101000033).

† E-mail: l.xuan01@mail.scut.edu.cn