随机数的生成

可厉害的土豆

- 在密码学中，生成随机位流是一个重要的密码函数。
- 大量基于密码学的网络安全算法和协议都使用随机二进制数。
- 用于密钥分发、互相认证、生成会话密钥等。

对生成随机数的有两个要求：随机性 和 不可预测性

- 随机性：在某种明确定义的统计意义下，数序列是随机的。
  - 分布均匀性：序列中的位分布应是均匀的，即0和1出现的概率相等。
  - 独立性：序列中的任何子序列都不能由其他子序列推导出来。
- 不可预测性：序列后续成员不可预测。

一般来说，有两种不同的策略来生成随机数：

- 真随机数生成器(True Random Number Generator，TRNG)

  非确定性随机位生成器(Non-deterministic Random Bit Generator, NRBG)

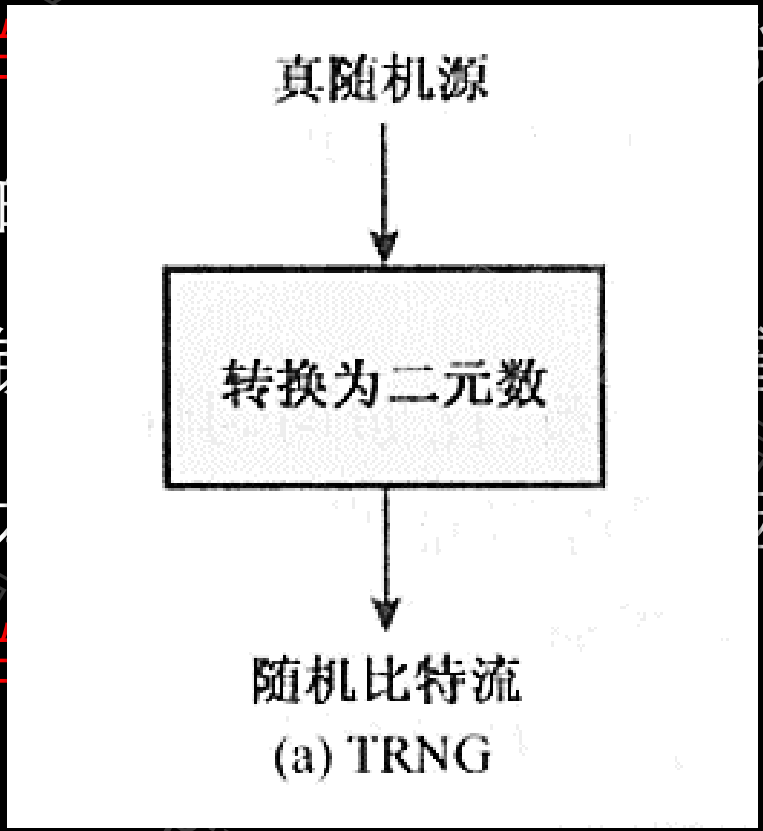
- 伪随机数生成器(Pseudo-Random Number Generator, PRNG)（占主导地位）

  确定性随机位生成器(Deterministic Random Bit Generator, DRBG)

真随机数生成器(TRNG)

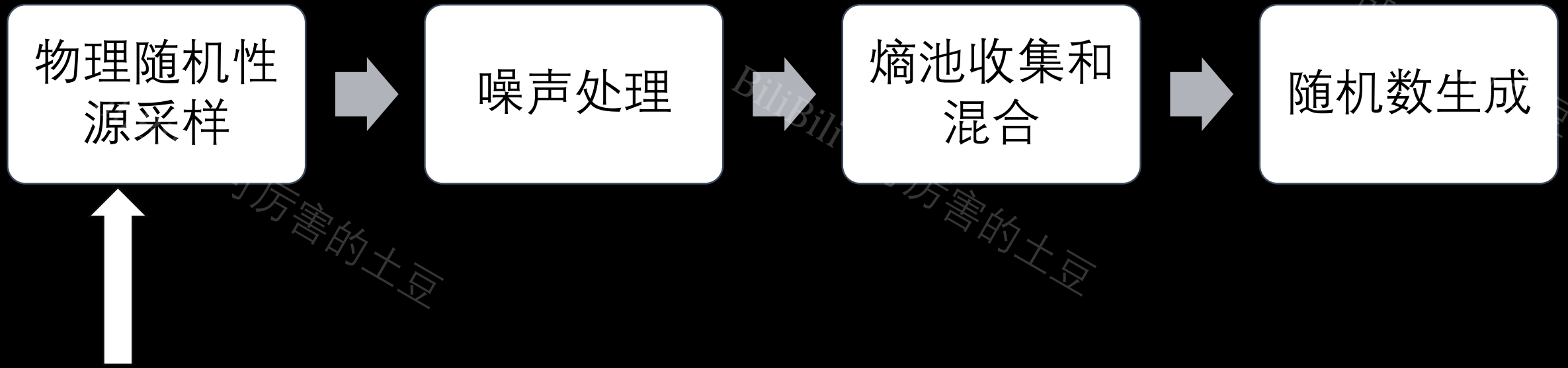
- 使用某种不确定的[illegible]随机位。
- RFC 4086中列出的[illegible]

  物理噪声、键[illegible]音频、视频输入等。
- 这些物理过程在本[illegible]因此TRNG生成的数字序列被认为是真正的[illegible]



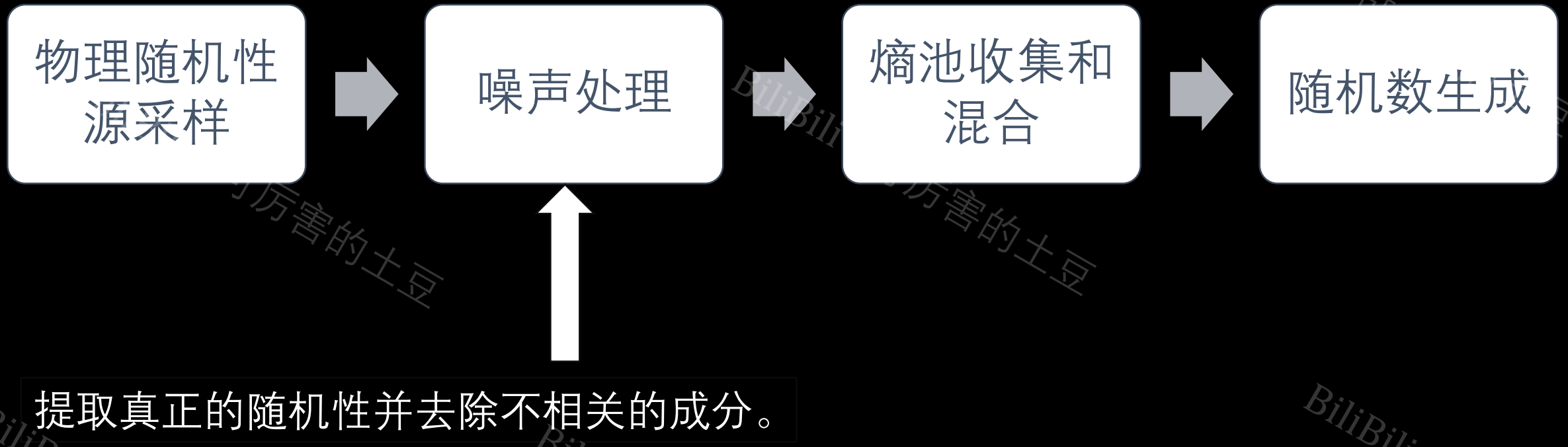
(a) TRNG

真随机数生成器(TRNG)

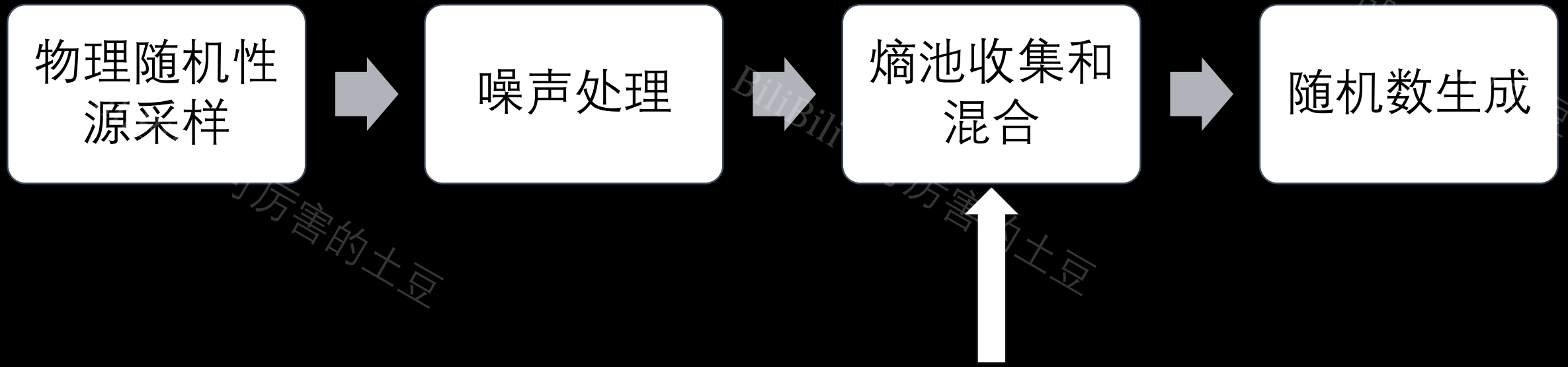


TRNG利用物理过程的不可预测性，从一个或多个物理随机性源中采样数据。

真随机数生成器(TRNG)



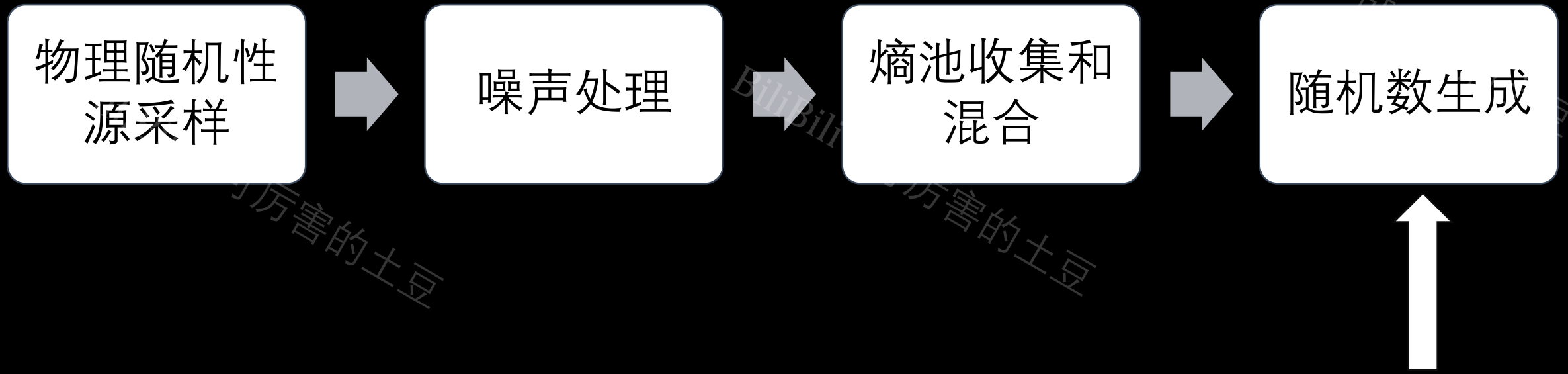
真随机数生成器(TRNG)



处理后的随机性数据被送入一个称为熵池（entropy pool）的缓冲区。

保证随机性和不可预测性。

真随机数生成器(TRNG)



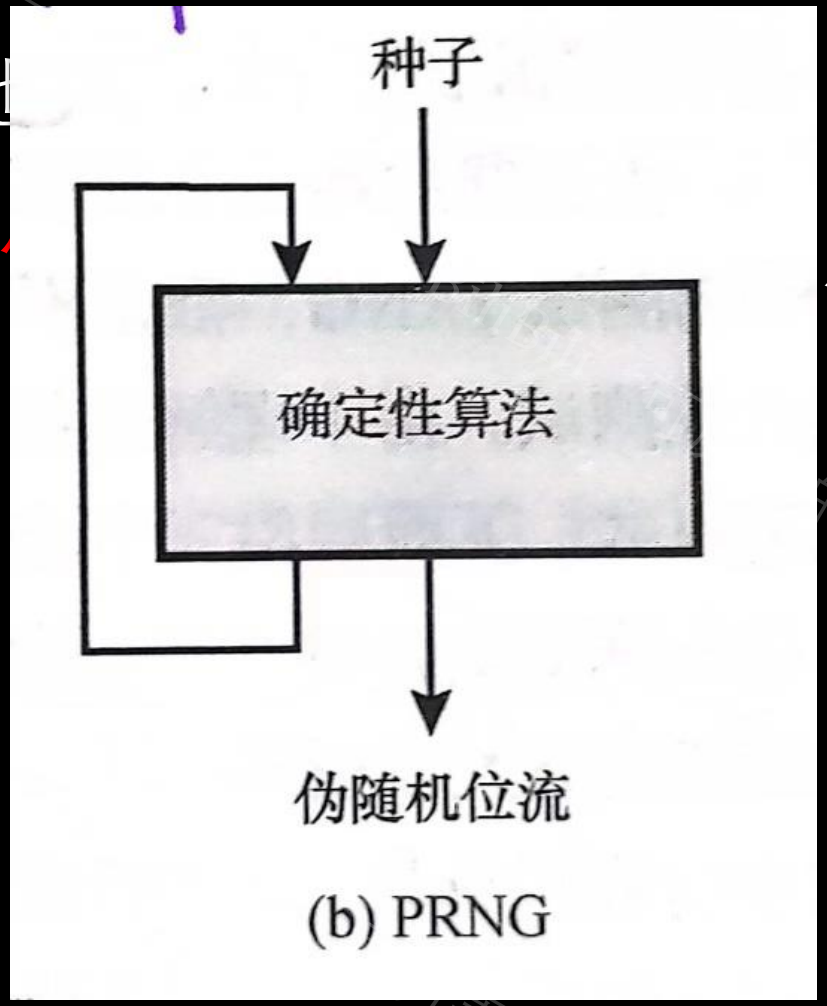
从熵池中提取数据，并通过一系列的加工步骤来生成最终的随机数。

真随机数生成器(TRNG)

- TRNG 的关键在于利用物理过程的不可预测性来产生真正的随机数。
- 缺点：成本高、效率低、环境敏感、输出数据可能存在一些偏差。
- 设计和实现TRNG时，需要综合考虑这些因素，采取适当的措施来减少偏差和提高生成的随机数的质量。

伪随机数生成器(PRNG)

- 使用算法确定性地
- PRNG取一个固定
用一个确定性算法生成输出位序列。



知道算法和种子，

就可以重现随机位流。

伪随机数生成器(PRNG)

种子必须是不可预测的，一般由TRNG生成。

为什么可以使用TRNG，还要使用PRNG?

若应用为流密码，TRNG不实用。

若只需生成有限数量的随机数，通常也希望使用TRNG做种子，使用PRF（伪随机函数）输出。

伪随机数生成器与伪随机函数区别是：

伪随机数生成器生成的随机位流不限长度，

伪随机函数输出的长度固定。

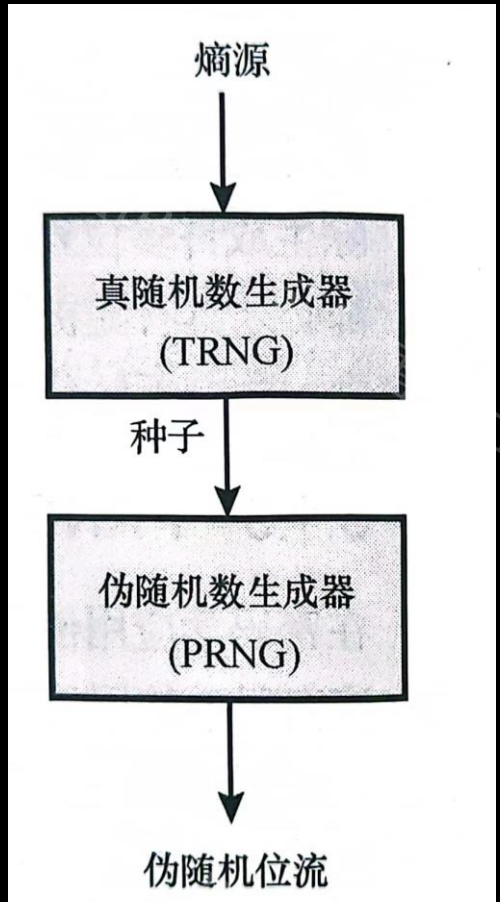


图 8.2　输入 PRNG 的种子的生成

伪随机数生成器(PRNG)

- PRNG算法大致分为两类：
- 专用算法：为生成伪随机位流而专门设计。
- 基于现有密码算法的算法：密码算法会随机化输入数据。
  - 对称分组密码
  - 哈希函数
  - 消息认证码

伪随机数生成器(PRNG)——线性同余生成器（LCG）

- 基于一个线性递归方程来产生一系列的伪随机数。

$$X_{n+1} = (aX_n + c) \bmod m$$

X_n——当前的随机数（或种子）　$0 <= X_n < m$

X_{n+1}——下一个随机数　　$0 < X_{n+1} < m$

a——乘法因子　$0<a<m$

c——增量　　$0<= c<m$

m——模数　　$m>0$

伪随机数生成器(PRNG)——线性同余生成器（LCG）

$$X_{n+1} = (aX_n + c) \bmod m$$

a=3，c=0，m=7，X_0=2

$X_0 = 2$ (初始种子值)

$X_1 = (3 * 2) \bmod 7 = 6$

$X_2 = (3 * 6) \bmod 7 = 4$

$X_3 = (3 * 4) \bmod 7 = 5$

$X_4 = (3 * 5) \bmod 7 = 1$

$X_5 = (3 * 1) \bmod 7 = 3$

$X_6 = (3 * 3) \bmod 7 = 2$（周期开始重复）

伪随机数生成器(PRNG)——线性同余生成器（LCG）

LCG 的性能和随机性质取决于选择的参数（a、c、m）以及初始种子值。

恰当地选择这些参数可以产生较好的伪随机数序列，但不当的选择可能导致周期性或可预测性。

在密码学和安全应用中，LCG 不被广泛使用，因为它的随机性性能不足以满足高安全性要求。

伪随机数生成器(PRNG)——BBS生成器

- Blum-Blum-Shub（BBS）算法是一种伪随机数生成算法，由Lenore Blum、Manuel Blum和Michael Shub 在1986年提出。
- 利用两个大素数的乘积作为模数，然后选择一个初始值（种子）作为计算的起点，然后迭代地计算下一个伪随机数。

伪随机数生成器(PRNG)——BBS生成器

- 参数选择： 选择两个大素数 p 和 q，要求它们满足以下条件：

  $p \equiv q \equiv 3 \pmod 4$ 计算 $n = p * q$。

- 种子选择： 选择一个整数种子 s，要求 s 与 n 互质。
- 迭代计算： 对于每个迭代步骤 i，计算 $s_i = (s_{i-1} \hat{} 2) \bmod n$。
- 输出伪随机数：计算 $s_i \bmod 2$ 输出为伪随机比特。

伪随机数生成器(PRNG)——BBS生成器

- 参数选择：p = 11　q = 19　n = p*q = 209　种子 s = 7

- 迭代计算：
  - s_0 = 7 (初始种子)
  - $s_1 = (7^2) \bmod n = 49 \% 209 = 49$

- 输出伪随机比特：
  - 生成的伪随机比特为 49 mod 2 = 1

使用分组密码生成伪随机数

- 广为人们接受的、使用分组密码构建PRNG的两种方式是CTR模式和OFB模式。
- 在每种情形中，种子都由两部分构成：
  - 加密密钥值
  - 每生成一个伪随机数分组后都将更新的V值

使用分组密码生成伪随机数

- 对于AES-128，种子由一个128位的密钥和一个128位的V值构成。
- 在CTR模式下，V值每加密一次就增1
- 在OFB模式下，V值更新为前一个PRNG分组的值

使用分组密码生成伪随机数

- 一次生成一个伪随机位分组（例如对于AES，一次生成128位的PRNG）

```
while   (len (temp) < requested_number_of_bits) do
        V = (V + 1) mod 2^128
        output_block = E(Key, V)
        temp = temp || output_block
```

OFB 算法总结如下。

```
while   (len (temp) < requested_number_of_bits) do
        V =  E(Key, V)
        temp = temp || V
```

感谢观看

祝你

每顿饭都吃饱

每晚都睡好

身体健康

学业有成

工作顺利

天天开心

~

可厉害的土豆