Blum, Blum, Shub。它也许是特意构造算法里密码强度有最强公开证明的一个了。产生过程如下：首先，选择两个大素数 p 和 q，且要求

$$p \equiv q \equiv 3 (\text{mod } 4)$$

该符号在第 4 章中有完整解释，表示（$p \bmod 4$）=（$q \bmod 4$）= 3。例如，素数 7 和 11 满足 7≡11≡3(mod 4)。令 $n = p \times q$。接着，选择一个随机数 s，且要求 s 与 n 互素，即 p 或 q 都不是 s 的因子。然后 BBS 按下列算法产生位 B_i 序列：

$$X_0 = s^2 \bmod n$$
$$\textbf{for } i = 1 \textbf{ to } \infty$$
$$X_i = (X_{i-1})^2 \bmod n$$
$$B_i = X_i \bmod 2$$

因此每个循环都是取的最低有效位。表 8.1 给出了 BBS 运算的一个例子。这里，n=192649=383×503，种子 s=101355。

BBS 被称为密码安全伪随机位发生器（CSPRBG）。它能经受住续位测试，在文献[MENE97]中续位测试定义如下：“称某伪随机位发生器可通过续位测试，若不存在多项式时间复杂度的算法[①]，对于某输出序列的最初 k 位输入，可以以超过 1/2 的概率预测出第（k+1）位”。换句话说，给定序列的最开始 k 位，没有有效算法可以让你以超过 1/2 的概率确定下一位是 1 还是 0。所以对于实际应用，这个序列是不可预测的。BBS 的安全性是基于对 n 的因子分解的困难性上的，即给定 n，我们不能确定它的素因子 p 和 q。

表 8.1 BBS 伪随机数发生器的一个例子

i	X_i	B_i	i	X_i	B_i
0	20749		11	137922	0
1	143135	1	12	123175	1
2	177671	1	13	8630	0
3	97048	0	14	114386	0
4	89992	0	15	14863	1
5	174051	1	16	133015	1
6	80649	1	17	106065	1
7	45663	1	18	45870	0
8	69442	0	19	137171	1
9	186894	0	20	48060	0
10	177046	0			

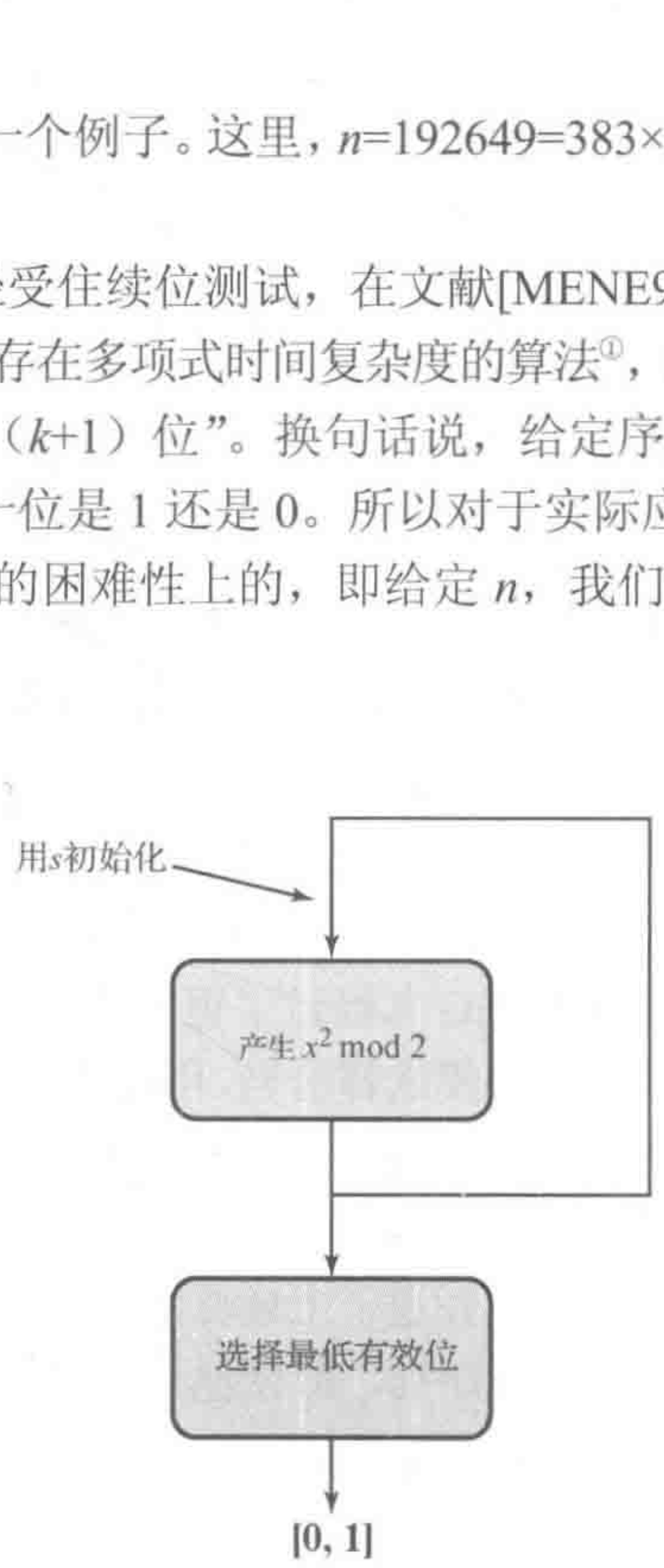


图 8.3 BBS 框图

8.3 使用分组密码的伪随机数产生

构造 PRNG 的常用方法是使用对称分组密码作为 PRNG 机制的核心。对于任意的明文分组，对称分组密码产生一个明显随机的分组输出。即密文里没有规则性或模式可用于推导明文。因此，对称分组密码很适合用来构造伪随机数发生器。

如果使用一个成熟的标准分组密码，如 DES 或 AES，那么：PRNG 的安全特性就得到保证了。而且，许多应用已经在使用 DES 或 AES，所以包含分组密码作为 PRNG 的一部分是简单直接的。

① 阶为 k 的多项式时间算法是指算法的运行时间由阶为 k 的多项式限定。

8.3.1　使用分组密码工作模式的 PRNG

两种使用分组密码构建 PRNG 的方法获得了广泛的接受：CTR 模式和 OFB 模式。SP 800-90A、ANSI 标准 X9.82 以及 RFC 4086 都推荐了 CTR 模式。X9.82 和 RFC 4086 推荐了 OFB 模式。

图 8.4 展示了两种方法。每一种情况里，种子由两部分组成：加密密钥值以及每产生一个随机数分组后都要有更新的 V 值。因此，对于 AES，种子由 128 位的密钥和 128 位的 V 值构成。当为 CTR 模式时，V 的值每加密一次后增加 1。当 OFB 模式时，V 的值更新为前一个 PRNG 分组的值。两种情况下，一次生产一个伪随机位分组（如对于 AES，一次生产 128 位的 PRNG 位）。

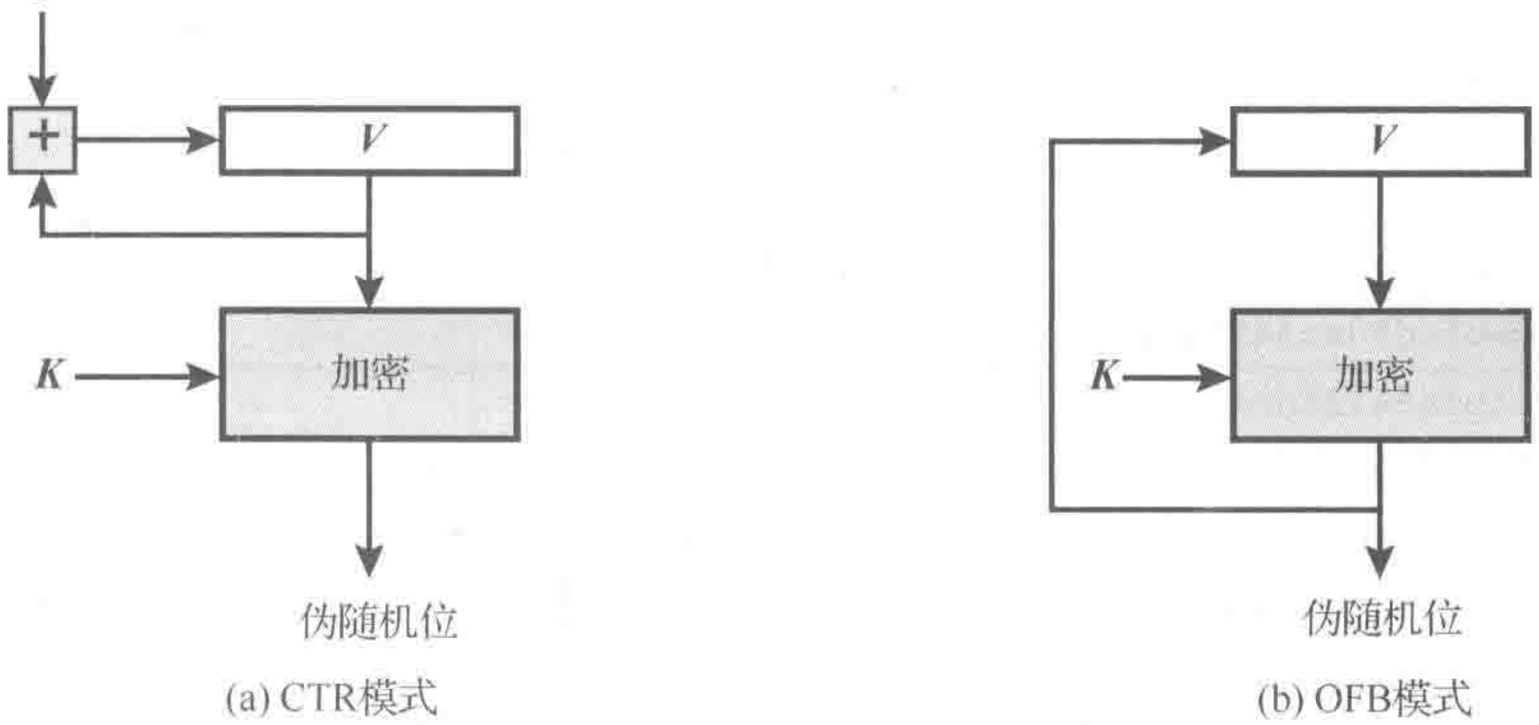


图 8.4　基于分组密码的 PRNG 机制

用于 PRNG 的 CTR 算法总结如下：

```
While (len(temp) <需要的位数量) do
        V=(V+1) mod 2^128
        output_block=E(Key,V)
        temp=temp ‖output_block
```

OFB 算法可以总结如下：

```
While (len(temp) <需要的位数量) do
        V=E(Key,V)
        temp=temp ‖V
```

为了了解这两个 PRNG 的性能，考虑如下的短实验。256 位的随机位序列是从 random.org 处获得的，通过使用三个相互之间可以调节的无线电来获取空气噪声。这 256 位形成种子，如下分配：

密钥:	Cfb0ef3108d49cc4562d5810b0a9af60
V:	4c89af496176b728ed1e2ea8ba27f5a4

256 位的种子里 1 的总数为 124，或者是 48%，很接近理想情况下的 50%。

对于 OFB 的 PRNG，表 8.2 显示了前 8 个输出分组（1024 位），带有两个粗略的安全度量指标。第二列显示了每个 128 位分组里 1 的分数。这对应于 NIST 的一个测试。结果表明输出为 0 和 1 的数量大约相等。第三列显示的是相邻分组匹配位的分数。如果这个数值偏离 0.5 比较大，这就表明分组间有相关，这是一个安全弱点。结果表明没有相关性。

表 8.3 显示了 CTR 模式下使用相同的密钥和 V 值的结果。结果再一次让人满意。

表 8.2 使用 OFB 的 PRNG 的例子结果

输出分组	位 1 所占的分数	和前一分组匹配位所占分数
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
5e17b22b14677a4d66890f87565eae64	0.51	0.52
fd18284ac82251dfb3aa62c326cd46cc	0.47	0.54
c8e545198a758ef5dd86b41946389bd5	0.50	0.44
fe7bae0e23019542962e2c52d215a2e3	0.47	0.48
14fdf5ec99469598ae0379472803accd	0.49	0.52
6aeca972e5a3ef17bd1a1b775fc8b929	0.57	0.48
f7e97badf359d128f00d9b4ae323db64	0.55	0.45

表 8.3 使用 CTR 的 PRNG 的例子结果

输出分组	位 1 所占分数	和前一分组匹配位所占分数
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
60809669a3e092a01b463472fdcae420	0.41	0.41
d4e6e170b46b0573eedf88ee39bff33d	0.59	0.45
5f8fcfc5deca18ea246785d7fadc76f8	0.59	0.52
90e63ed27bb07868c753545bdd57ee28	0.53	0.52
0125856fdf4a17f747c7833695c52235	0.50	0.47
f4be2d179b0f2548fd748c8fc7c81990	0.51	0.48
1151fc48f90eebac658a3911515c3c66	0.47	0.45

8.3.2 ANSI X9.17 伪随机数发生器

ANSI X9.17 中所给出的伪随机数发生器是密码学意义上最强的伪随机数发生器之一。许多应用使用了这种方法，包括金融安全应用和 PGP（后者详见第 19 章）。

图 8.5 说明了算法流程，它使用了 3DES 来加密。构成成分如下：

- **输入** 用 2 个伪随机数输入来驱动发生器。一个是 64 位的数，代表当前的日期和时间，每产生一个伪随机数它均要更新。另一个是 64 位的种子，可以被初始化为任意值，并在生成随机数的过程中被更新。
- **密钥** 发生器使用 3 个 3DES 加密模块。3 个加密模块都使用一个相同的 56 位密钥对，这个密钥对必须保密，且只在产生随机数时才使用。
- **输出** 输出包括 64 位的伪随机数和 64 位的种子。

定义以下变量：

DT_i：算法第 i 轮开始时的日期/时间值。

V_i：算法第 i 轮开始时的种子值。

R_i：算法第 i 轮所产生的伪随机数。

K_1, K_2：各阶段算法所用的 DES 密钥。

有

$$R_i = \text{EDE}([K_1,K_2], [V_i \oplus \text{EDE}([K_1, K_2], DT_i)])$$
$$V_{i+1} = \text{EDE}([K_1,K_2], [R_i \oplus \text{EDE}([K_1, K_2], DT_i)])$$

其中，$\text{EDE}([K_1, K_2], X)$代表加密-解密-加密序列加密 X，使用的算法为两个密钥的 3DES。

该方法的密码强度来自几个方面，包括 112 位密钥和 3 个 3DES 共计 9 次 DES 加密。整个方案的输入是两个伪随机数，日期和时间值，一个由发生器产生的种子，且其区别于发生器产生的伪随机数。所以敌手要分析的东西太多了，即使他知道了一个 R_i，也不能由 R_i 推导出 V_{i+1}。因为产生 V_{i+1} 时又用了一次 3DES 运算。

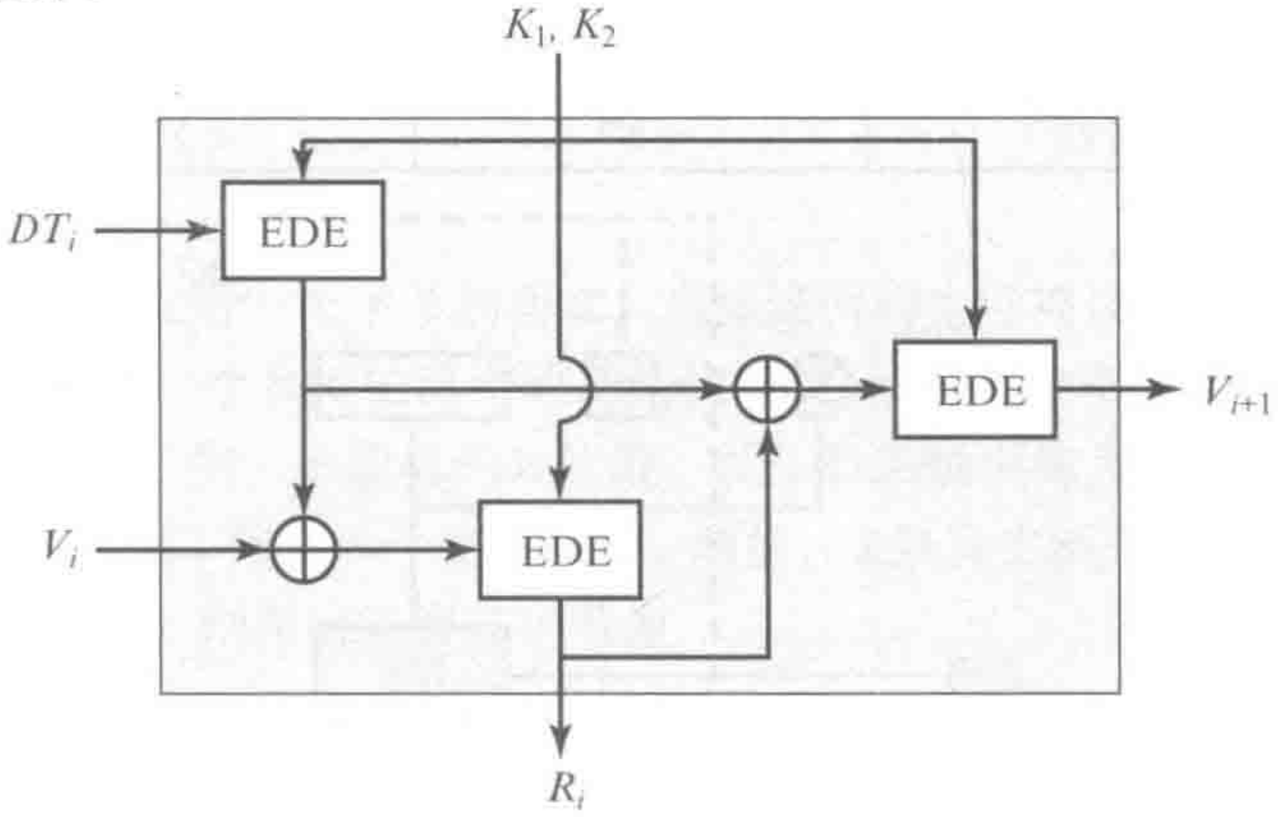


图 8.5　ANSI X9.17 伪随机数发生器

8.3.3　NIST CTR_DRBG

现在我们考虑定义于 NIST SP 800-90 中的基于 CTR 工作模式的 PRNG。这种 RPNG 被称为 CTR_DRBG(counter mode-deterministic random bit generator)。CTR_DRBG 被广泛地实现并且是最近的英特尔处理器芯片的（在 8.6 节讨论）硬件随机数发生器的一部分。

DRBG 假设熵源可提供随机位，熵源通常是基于一些物理源的 TRNG。其他源也是可能的，如果他们能被应用程序测量所需的熵。熵是信息论的概念用来度量不可预测性或随机性，详情参见附录 F。DRBG 使用的加密算法可能是带 3 个密钥的 3DES 或者密钥长度为 128 位、192 位或 256 位的 AES。

与该算法有关的 4 个参数为

- **输出分组长度（outlen）** 加密算法的输出分组的长度。
- **密钥长度（keylen）** 加密密钥的长度。
- **种子长度（seedlen）** 种子是一个位串作为 DRBG 机制的输入。种子确定了 DRBG 的一部分内部状态，它的熵必须能保证 DRBG 安全性。seedlin = outlen + keylen。
- **补种间隔（reseed_interval）** 加密密钥的长度。它是用新种子更新算法前的输出分组的最大数量。

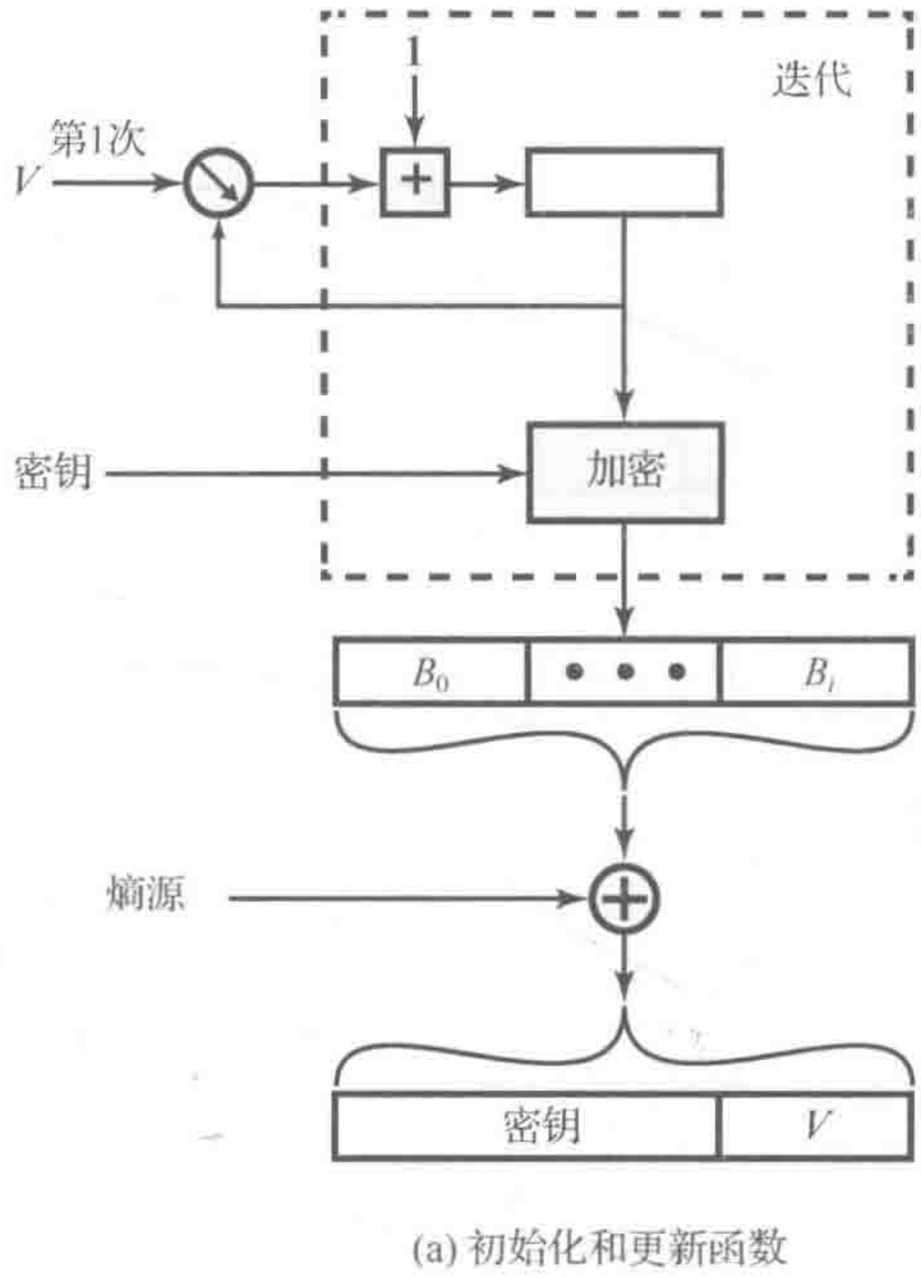
表 8.4 列出了 SP 800-90A 的这些参数的值。

初始化　如图 8.6 所示，CTR_DRBG 由两个主要函数组成。我们首先考虑 CTR_DRBG 如何使用初始化和更新函数初始化［参见图 8.6(a)］。CTR 分组模式需要一个密钥 K 和一个 SP 800-90A 的初试计数器值 V。K 和 V 的组合称为种子。开始 DRGB 操作时需要任意的 K 和 V。如 8.6 节介绍的英特尔数字随机数发生器使用 K=0 和 V=0。这些值是 CTR 工作模式用来产生 seedlen 个位的参数。此外，全部的 seedlen 个位必须是熵源提供的，熵源通常是某种形式的 TRNG。

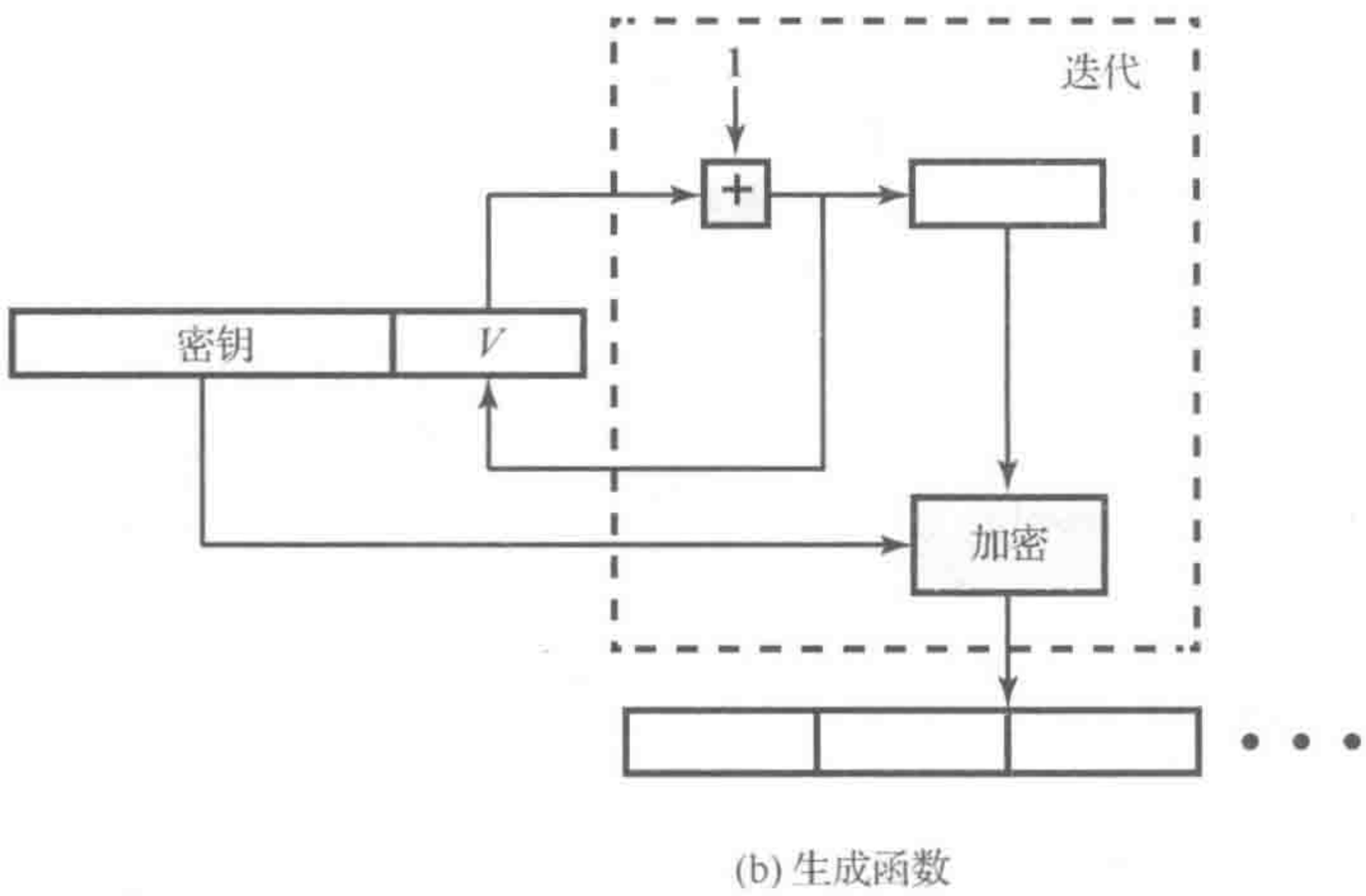
在这些输入下迭代 CTR 模式产生一个输出分组序列，每次加密后 V 加 1。这个过程持续到产生了最后一个 seedlen 位。最左边的 seedlen 输出位异或 seedlen 位从而产生一个新的种子。依次地，种子的最左边 keylen 位产生新密钥，种子的最右边 outlen 位产生新的计数器值 V。

表 8.4 CTR_DRBG 参数

	3DES	AES-128	AES-192	AES-256
outlen	64	128	128	128
keylen	168	128	192	256
seedlen	232	256	320	384
reseed_interval	$\leqslant 2^{32}$	$\leqslant 2^{48}$	$\leqslant 2^{48}$	$\leqslant 2^{48}$



(a) 初始化和更新函数



(b) 生成函数

图 8.6 CTR_DRBG 函数

产生 一旦获取 Key 和 V 的值后 DRBG 就进入产生阶段且能生成伪随机位，每一次一个分组［参见图 8.6(b)］。迭代加密函数生成期望的伪随机位，每次迭代使用相同的密钥，计数器值 V 每次迭代加 1。

更新 为了提高安全性，PRNG 产生的位数应该受限。CTR_DRGB 使用参数 reseed_interval 来

设置那个限制。在产生阶段，reseed counter 初始化为 1 且每次迭代都增加。当 reseed counter 达到 reseed_interval 时引入更新函数［参见图 8.6(a)］。更新函数和初始化函数相同。在更新阶段，最后被生成函数使用的 Key 和 V 值作为更新函数的输入参数。更新函数使用熵源的 seedlen 新位生成一个新种子（Key, V）。生成函数能保留伪随机位。注意到更新函数的结果会改变生成函数使用 Key 和 V 值。

8.4　流密码

一个典型的流密码每次加密一个字节的明文，当然流密码也可被设计为每次操作一位或者大于一字节的单元。图 8.7 给出了一个典型的流密码的结构图。在该结构中密钥输入到一个伪随机数(位)发生器，该伪随机数发生器产生一串随机的 8 位数。发生器的输出称为**密钥流**，密钥流和明文流的每一个字节进行按位异或运算，得到一个密文字节。例如，如果发生器产生的下一字节为 01101100，而下一明文字节为 11001100，则得出的密文字节为

```
  11001100    明文
⊕ 01101100    密钥流
  10100000    密文
```

解密需要使用相同的伪随机序列

```
  10100000    密文
⊕ 01101100    密钥流
  11001100    明文
```

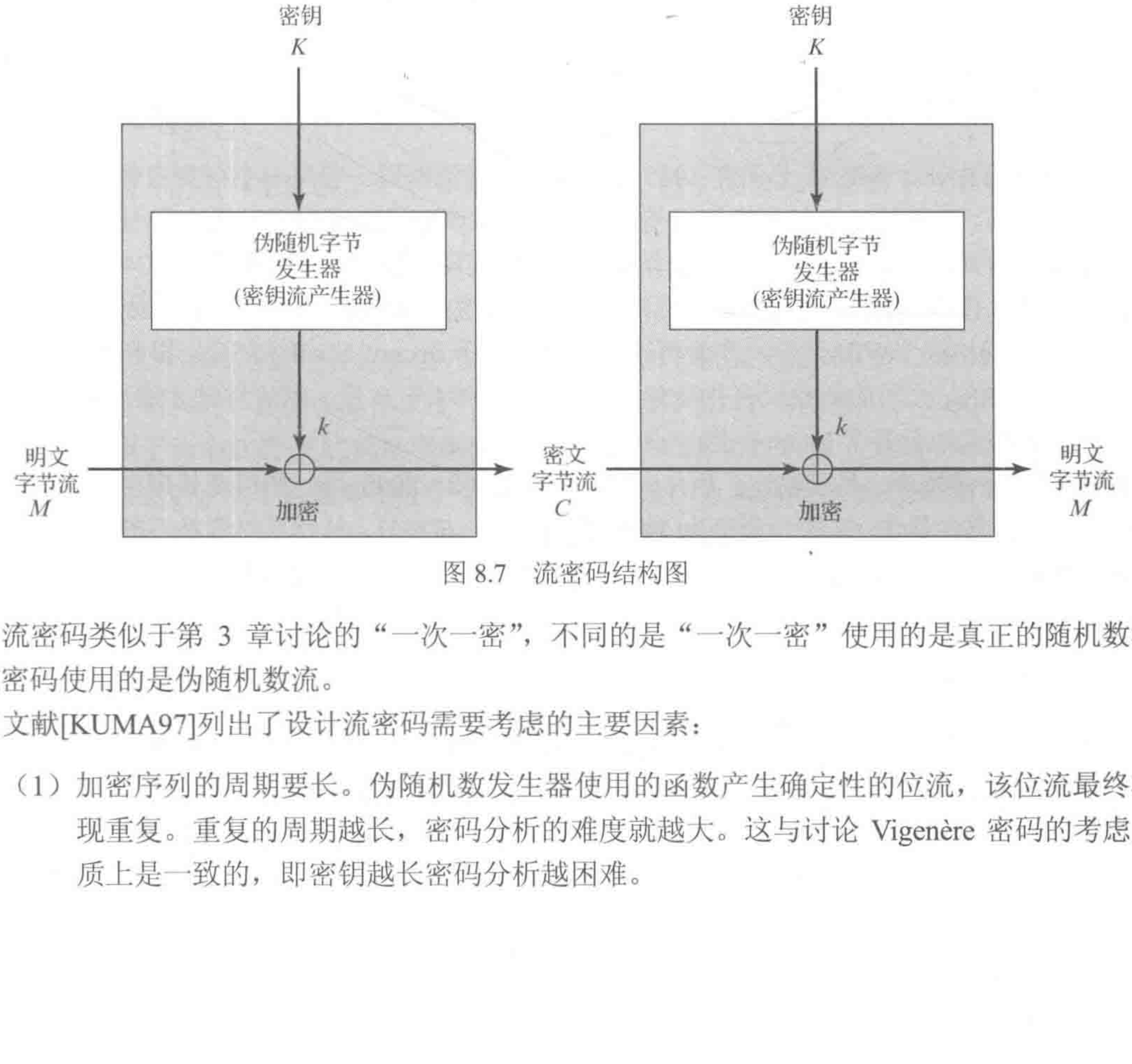


图 8.7　流密码结构图

流密码类似于第 3 章讨论的“一次一密”，不同的是“一次一密”使用的是真正的随机数流，而流密码使用的是伪随机数流。

文献[KUMA97]列出了设计流密码需要考虑的主要因素：

（1）加密序列的周期要长。伪随机数发生器使用的函数产生确定性的位流，该位流最终将出现重复。重复的周期越长，密码分析的难度就越大。这与讨论 Vigenère 密码的考虑从本质上是一致的，即密钥越长密码分析越困难。