[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04L 9/22
H04L 9/26

[12] 发明专利申请公开说明书

[21] 申请号 200410032253.3

[43] 公开日 2004 年 9 月 22 日

[11] 公开号 CN 1531244A

[22] 申请日 2000.2.4
[21] 申请号 200410032253.3
分案原申请号 00803505.9
[30] 优先权
[32] 1999. 2. 8 [33] US [31] 09/246，366
[71] 申请人 高通股份有限公司
地址 美国加利福尼亚州
[72] 发明人 G·G·罗斯 小 R·F·奎克

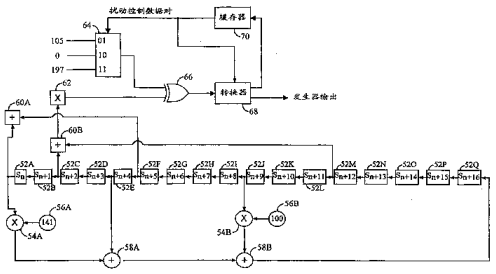
[74] 专利代理机构 上海专利商标事务所
代理人 钱慰民

权利要求书 3 页 说明书 28 页 附图 12 页

[54] 发明名称 生成加密数据流密码的方法和装置

[57] 摘要

产生加密数据流密码的方法和装置是基于一种递归关系的，该递归关系是在大于 GF(2)的有限域上工作的。 可以采用一种或组合的非线性过程以形成输出函数来获得非线性输出。 可以选择递归关系和输出函数使之具有明显不同的数据对距离，从而当移位寄存器(52)移位时，无论是在递归关系中还是在输出函数中，都不会在移位寄存器(2)中两次使用相同的数据对元素。 在这些条件下，还可以选择递归关系和输出函数，以使密码保密性或计算效率为最佳。 另外，本发明另一要解决的问题是提供一种确保形成加密过程的延迟不会超过预定的界限的方法。 为此，测量加密延迟，如果估计的延迟超过了预定的阈值，则采用第二种加密方法，来限制加密操作的累计延迟。



ISSN1008-4274

知识产权出版社出版

1. 一种生成数据流密码的方法，其特征在于，它包含：

选择具有阶数大于 2 的有限域；

在所述有限域上选择一种递归关系；

选择一种输出函数；以及

按照所述递归关系和所述输出函数来计算所述数据流密码，其中，

所述递归关系和所述输出函数具有明显不同的数据对差，并且

根据密码保密性和计算效率，选择所述递归关系和所述输出函数，以使特性标准为最佳。

2. 如权利要求 1 所述的方法，其特征在于，所述有限域是根据用于计算所述数据流密码的处理器的字长来选择的。

3. 如权利要求 1 所述的方法，其特征在于，所述有限域是含有 256 个元素的伽罗瓦域。

4. 如权利要求 1 所述的方法，其特征在于，所述递归关系是最大长度。

5. 如权利要求 1 所述的方法，其特征在于，所述递归关系的阶数是 17。

6. 如权利要求 1 所述的方法，其特征在于，所述递归关系是用线性反馈移位寄存器来实现的。

7. 如权利要求 6 所述的方法，其特征在于，所述线性反馈移位寄存器是用循环缓存器来构成的。

8. 如权利要求 6 所述的方法，其特征在于，所述线性反馈移位寄存器是用滑动窗口构成的。

9. 如权利要求 1 所述的方法，其特征在于，所述性能标准是仅仅根据密码的保密性来实现的。

10. 如权利要求 1 所述的方法，其特征在于，所述性能标准是仅仅根据计算效率来实现的。

11. 如权利要求 1 所述的方法，其特征在于，所述输出函数包含计算所述发生器的状态的非线性函数。

12. 如权利要求 11 所述的方法，其特征在于，所述输出函数包含数据位的旋转。

13. 如权利要求 11 所述的方法，其特征在于，所述输出函数包含数据位的对

换。

14. 如权利要求 1 所述的方法，其特征在于，所述计算步骤包含域相乘和模相加。

15. 如权利要求 14 所述的方法，其特征在于，所述域相乘的结果是由素数的模来约减的。

16. 如权利要求 15 所述的方法，其特征在于，所述的素数是 257。

17. 如权利要求 14 所述的方法，其特征在于，所述的域相乘是用查询表来实现的。

18. 如权利要求 17 所述的方法，其特征在于，所述的查询表是预先计算并存储在存储单元内的。

19. 如权利要求 14 所述的方法，其特征在于，所述域相乘是通过下述步骤来实现的：

对两个运算对象中的每一个的对数值表格进行查询；

对所述两个运算对象的对数值进行模相加，以得到一个组合的对数值；以及

对所述组合的对数值的指数值表格进行查询。

20. 如权利要求 1 所述的方法，其特征在于，它还包含下述步骤：

用一个密钥来使所述发生器初始化。

21. 如权利要求 20 所述的方法，其特征在于，所述初始化步骤包含下述步骤：

将所述密钥的最小有效字节加到所述递归关系上；

使所述密钥移位一个字节；以及

重复所述相加步骤和所述移位步骤，直到所述密钥内的所有的字节都被加到了所述递归关系上。

22. 如权利要求 20 所述的方法，其特征在于，所述密钥的长度小于所述递归关系的阶数。

23. 如权利要求 20 所述的方法，其特征在于，它还包含下述步骤：

按照数据帧的密钥，对所述发生器进行初始化。

24. 如权利要求 23 所述的方法，其特征在于，按照数据帧的密钥对所述发生器进行所述初始化包含下述步骤：

将所述按照数据帧的密钥的最小有效字节加到所述递归关系上；

使所述按照数据帧的密钥移位三个数据位；

重复所述相加步骤和所述移位步骤，直到所述数据帧的密钥中的所有的字节

被加到了所述递归关系上。

25. 如权利要求 23 所述的方法，其特征在于，所述按照数据帧的密钥的长度是 4 个 8 位数据位长。

26. 如权利要求 23 所述的方法，其特征在于，用数据帧密钥使所述发生器进行所述初始化的步骤是针对每一数据帧进行的。

27. 一种生成数据流密码的装置，其特征在于，它包含：

接收用于执行递归关系和输出函数的指令的处理器，所述处理器按照所述指令执行对元素的运算，其中，

所述递归关系和所述输出函数具有明显不同的数据对差异，以及

根据密码的保密性和计算效率，选择所述递归关系和所述输出函数，以使性能标准为最佳。

28. 如权利要求 27 所述的装置，其特征在于，所述递归关系是定义在具有大于 1 的阶数的有限域上的。

29. 如权利要求 28 所述的装置，其特征在于，所述有限域是按照所述处理器的字长来选择的。

30. 如权利要求 28 所述的装置，其特征在于，所述有限域是含有 256 个元素的伽罗瓦域。

31. 如权利要求 27 所述的装置，其特征在于，所述递归关系是最大长度。

32. 如权利要求 27 所述的装置，其特征在于，所述递归关系的阶数是 17。

33. 如权利要求 27 所述的装置，其特征在于，所述递归关系是用线性反馈移位寄存器来实现的。

34. 如权利要求 33 所述的装置，其特征在于，所述线性反馈移位寄存器是用循环缓存器来构成的。

35. 如权利要求 33 所述的装置，其特征在于，所述线性反馈移位寄存器是用一个滑动窗口来实现的。

生成加密数据流密码的方法和装置

本申请是申请日为2000年2月4日申请号为第00803505.9号发明名称为“生成加密数据流密码的方法和装置”的中国专利申请的分案申请。

发明背景

I. 发明领域

本发明涉及加密。本发明尤其涉及一种生成加密数据流密码的方法和装置。

II. 相关技术领域的描述

加密是一种处理过程，通过该过程，采用随机处理对数据进行运算，从而使除特定用户外的所有的用户都不能理解这些数据。对数字化的数据进行加密的一种方法是通过采用数据流密码来完成的。数据流密码是采用要加密的数据以及由加密规则所产生的伪随机位数据流(或加密位数据流)并且通常采用异或(XOR)运算将它们合并来进行工作的。解密是这样一种简单的处理过程，它生成同样的加密位数据流，并采用相应的运算操作从加密的数据中去除加密位数据流。如果XOR运算是在加密一侧进行的，那么在解密侧也进行相同的XOR运算。对于保密的加密处理过程，加密位数据流必须是在计算上较难预测的。

用于生成伪随机数数据流的许多技术是基于阶数为 2 的伽罗瓦有限域上线性反馈移位寄存器(LFSR)的。这是 2^n 阶伽罗瓦有限域在 n 是正整数时的特殊情况。当 n＝1 时，伽罗瓦域元素包含 0 和 1 的数据位值。寄存器是通过使数据位移位一个数据位位置，并计算新的输出数据位来更新的。将新的数据位移位到寄存器内。对于斐波那契寄存器，输出数据位是寄存器中的数据位的线性函数。而对于伽罗瓦寄存器，许多数据位是按照刚刚从寄存器中移位出来的输出数据位来更新的。从数学上说，斐波那契寄存器和伽罗瓦寄存器的结构是等效的。

生成伪随机数数据流即移位和数据位获取的运算操作在硬件上没问题，但在软件或采用通用处理器或微处理器进行的其他实施方式上却存在一些难度。存在的难度在移位寄存器的长度超过用于生成数据流的处理器中寄存器的长度时而增加。另外，当 n＝0 时，每一组运算操作仅生成一个输出数据位，从而处理器的利用率很低。

采用数据流密码的典型的应用场合是无线电话。典型的无线电话通信系统是码分多址(CDMA)系统。CDMA 系统操作运行可参见已转让给本发明受让人的美国专利 4,901,307，该专利的标题是“SPREAD SPECTRUM MULTIPLE ACCESS COMMUNICATION SYSTEM USING SATELLITE OR TERRESTRIAL REPEATERS”。已转让给本发明的受让人的美国专利 5,103,459 中进一步揭示了一种 CDMA 系统，该专利的标题是“SYSTEM AND METHOD FOR GENERATING SIGNAL WAVEFORMS IN A CDMA CELLULAR TELEPHONE SYSTEM”。另一种 CDMA 系统包括 GLOBALSTAR 通信系统，用于采用低地轨道卫星进行的全球通信。其他的无线电话系统包括时分多址(TDMA)系统和频分多址(FDMA)系统。可以将 CDMA 系统设计成满足“双模式宽带扩谱蜂窝系统的 TIA/EIA/IS-95 移动站－基站兼容性标准”，该标准在本文中简称为 IS-95 标准。同样，可以将 TDMA 系统设计成满足 TIA/EIA/IS-54(TDMA)标准，或者对于欧洲全球系统来说满足移动通信(GSM)标准。

由于远端站中缺乏计算能力，而使无线电话中数字化话音数据加密受到了影响。这使得诸如 TDMA 标准中所使用的话音专用屏蔽加密处理能力较弱，并使生成如 GSM 标准中所使用的 A5 密码的数据流密码的硬件受到影响。硬件数据流密码的缺点是附加的硬件成本，以及当加密过程需要改变时，所花的时间及费用具大。由于无线电话系统和数字电话中许多的远端站都包含有微处理器和存储器，速度快并且采用很少存储器的数据流密码很适合于这些应用场合。

发明概述

本发明是一种生成加密数据流密码的新的、改进的方法和装置。按照本发明，设计循环关系(或称之为递归关系)，使之在比 GF(2)更大的有限域上进行运算操作。可以采用循环缓存器或使用滑动窗口来实现形成循环关系的线性反馈移位寄存器。在典型的实施例中，有限域元素的运算是用查询表来进行的。采用一个或组合的非线性处理过程，可以得到一个非线性输出。可以将数据流密码设计成支持多层密钥控，以满足数据流密码的使用要求。

本发明要解决的问题是，采用在处理器中实现起来较简单的结构来生成加密数据流密码。特别是，可以通过选择更适合于处理器的有限域，来实现更为有效的结构。可以选择循环关系的元素和系数，来配合处理器的字节或字的长度。这就使得可以由处理器对元素进行有效的运算。在典型的实施例中，所选择的有限域是具有 256 个元素(GF(2^8))的伽罗瓦域。这使得循环关系的元素和系数占用存储器的一

个字节，而使得可以进行有效的变换。另外，使用大的有限域减少了循环关系的阶数。对于有限域 GF(2^n)，对相同状态数进行编码的循环关系的阶数 k 减少了 n 倍(对于典型的 GF(2^n)，是 8 倍)。

本发明要解决的另一个问题是采用查询表来进行域的乘法。在该典型的实施例中，可以通过对两个运算对象中的每一个运算对象取对数、将对数值相加并对混合的对数值取指数，而在域中(对非零元素)进行乘法。可以用一种不可约多项式，来生成对数表和指数表。在本典型实施例中，表格是预先计算并存储在存储器中的。同样，可以用一种简单的查询表来进行常数系数的域计算。另外，可以用不可约多项式来预先计算表格，并将表格存储在的存储器内。

本发明再要解决的问题是，采用一个或混合的下述处理过程，去除线性反馈移位寄存器输出中的线性关系：不规则扰动(有时也称为抽取)、非线性函数、多个移位寄存器以及混合寄存器的输出、一个寄存器上的变量反馈多项式以及其他的非线性过程。在本典型实施例中，可以采用非线性输出来随机控制移位寄存器的扰动(stuttering)。另外，通过对移位寄存器内被选择的元素上进行非线性运算，可以得到非线性输出。此外，可以将非线性函数的结果与一组常数进行异或，而使非线性输出位出现不可预计的转化。

本发明的要解决的再一个问题是，用循环缓存器或滑动窗口来构成线性反馈移位寄存器。采用这种循环缓存器或滑动窗口结构，缓存器中的元素是不移位的。相反，采用指针或指示符来给出最近计算的元素的位置。指针随着进行新元素的计算而移动，并移位到循环缓存器或滑动窗口内。当指针到达边缘时，指针绕回。

本发明再要解决的问题是提供一种具有多层密钥控能力的数据流密码。在该典型实施例中用一个密钥首先使移位寄存器的状态初始化。对于某些以帧的方式进行数据传送的通信系统，可以为每一帧生成一个数据流密码，从而擦除的或溢出的序列帧不会打断加密过程。用帧密钥初始化过程，可以为每一帧，使第二层密钥控过程初始化。

本发明再要解决的问题是，使用最大长度的循环关系，而使得序列复盖了重复前最大数量的状态。

本发明还要解决的问题是，采用清晰配对差异的循环关系或输出等式。清晰配对差异是，当用来实现循环关系的移位寄存器出现移位时，无论是在循环关系或非线性输出等式中，移位寄存器中不会出现特定元素对使用过两次的情况。这种特性就从输出等式的结果中去掉了线性关系。

本发明还要解决的问题是，在具有清晰配对差异的同时，按照应用的要求，有选择地使密码保密性以及计算效率为最佳。

再有，本发明要解决的问题是，提供一种方法，使加密过程的延迟不会超过预定的界限。为此，对加密延迟进行测量，如果估计的延迟超过了预定的阈值，则采用第二种加密方法，来限制加密操作的累计延迟。

附图简述

在参照附图阅读了本发明的详细描述以后，读者将会更清楚地了解本发明的特征、目的和优点。图中，相同的标号所表示的意义相同。

图 1 是本发明典型实施例循环关系的方框图；

图 2 是采用处理器的数据流密码发生器的典型方框图；

图 3A 和 3B 分别表示时刻 n 和时刻 n+1 时循环缓存器内容的图；

图 3C 表示滑动窗口内容的图；

图 4 是本发明典型数据流密码发生器的方框图；

图 5 是本发明典型密钥初始化过程的流程图；

图 6A 是本发明典型每一帧初始化过程的流程图；

图 6B 是本发明第二种典型每一帧初始化过程的流程图；

图 7 是本发明第二种典型数据流密码发生器的方框图；

图 8 是本发明第三种典型数据流密码发生器的方框图；

图 9 是本发明用于限制加密延迟通常实施结构的方框图；

图 10 是本发明用于限制采用随机抽取或扰动的加密密码中的加密延迟的一种通用结构的方框图；

图 11 是本发明用于限制采用图 7 所示修改形式的随机抽取或扰动的加密密码中的加密延迟一种通用结构的方框图。

较佳实施例的详细描述

线性反馈移位寄存器(LFSR)是基于伽罗瓦域上的循环关系的，这里的输出序列由下面的循环关系定义：

$$S_{n+k} = C_{k-1}S_{n+k-1} + C_{k-2}S_{n+k-2} + \cdots + C_1S_{n+1} + C_0S_n \qquad (1)$$

式中，S_{n+k}是输出元素，C_j是常系数，k 是递归关系的阶数，而 n 是时间指数。状态变量 S 和系数 C 是基础有限域的元素。等式(1)有时也用本说明书中忽略的常

数项来表示。

图 1 中示出了等式(1)中递归关系的典型实施方式的方框图。对于阶数为 k 的递归关系，寄存器 12 含有 k 个元素 S_n 至 S_{n+k-1}。将这些元素提供给伽罗瓦域乘法器 14，将这些元素与常数 C_j 相乘。从乘法器 14 得到的乘积提供给伽罗瓦域加法器 16，将乘积加和，给出输出元素。

对于 n＝1，GF(2)的元素包含一个数据位(0 值或 1 值)，而等式(1)的执行需要许多次按数据位进行的运算。这时，由于是采用设计用来对字节或字大小的对象进行计算的处理器来对一个数据位进行许多次运算的，所以采用通用处理器来构成递归关系是低效率的。

本发明中，线性反馈移位寄存器是设计用于对大于 GF(2)的有限域进行计算的。特别是，通过选择更适合于处理器的有限域，可以构成更有效的结构。本实施例中，选择的有限域是具有 256 个元素(GF(2^8))的伽罗瓦域，或具有 2^n 个元素的伽罗瓦域，这里，n 是处理器的字长。

在本较佳实施例中，采用的是具有 256 个元素(GF(2^8))的伽罗瓦域。这就使得递归关系中的每一个元素和系数占据存储器的一个字节。可以由处理器更有效地进行字节运算。此外，对相同数量的状态进行编码的递归关系的阶数 k 减少了 n 倍，对于 GF(2^8)来说是 8 倍。

本发明中，采用最大长度的递归关系可以得到最佳的结果。最大长度指的是在重复前输出序列的长度(寄存器的状态数)。对于阶数为 k 的递归关系，最大长度是 N^k-1，这里，N 是基础有限域中的元素数，并且在本较佳实施例中，N=256。状态全部为 0 是不允许的。

图 2 中示出的是采用处理器的数据流密码发生器的典型方框图。控制器 20 连接到处理器 22，并且包含一组指示处理器 22 进行操作的指令。所以，控制器 20 可以包含一个软件程序或一组微码。处理器 22 是一个执行发生器所需的运算的硬件。处理器 22 可以是一个微控制器、微处理器，或者是设计用于执行本文中的功能的数字信号处理器。存储器元件 24 连接到处理器 22，并用来构成线性反馈移位寄存器，并储存将在下文中描述的预计算表和指令。存储器元件 24 可以是用随机存取存储器或其他设计用于执行这里所描述的功能的存储器件构成的。指令和表格可以存储在只读存储器内，只有用于寄存器本身的存储器才需要在算法的执行期间进行修改。

I. 生成非线性输出数据流

采用线性反馈移位寄存器用于数据流密码是很难恰当执行的。这是因为可以利用输出数据流中剩余的线性关系得到某一时刻的寄存器状态。接着，按照需要，可以使寄存器向前或者向后，使输出数据流恢复。可以采用几种技术，用线性反馈移位寄存器来生成非线性数据流密码。在本典型实施例中，这些非线性技术包含：寄存器的扰动(无法预计的抽取)，对寄存器的状态采用非线性功能，采用多个寄存器和寄存器输出的非线性组合，在一个寄存器上使用变量反馈多项式，以及其他的非线性过程。这些技术中的每一种技术描述如下。某些技术是通过举例来说明的。也可以采用其他的技术来生成非线性数据流，这些技术也包括在本发明的范围内。

扰动是这样一种处理过程，即，以可变的和不可预计的方式对寄存器实行时钟控制。扰动实施简单，并且能给出良好的结果。扰动时，在数据流密码处是不提供与寄存器的某些状态相关的输出的，因此，就更难从数据流密码来重新构筑寄存器的状态。

对移位寄存器的状态采用非线性函数也可以给出良好的结果。如等式(1)所定义的那样，对于某一递归关系，从寄存器状态的线性函数和系数，产生输出元素。为了提供非线性关系，可以从寄存器状态的非线性函数中生成输出元素。特别是，可以采用对通用处理器上的字节或字的长度数据进行运算的非线性函数。

采用多个移位寄存器并以非线性方式将寄存器的输出组合起来，可以给出良好的结果。可以以硬件的方式容易地构成多个移位寄存器，这时所添加的费用是最少的，并且可以并行地操作移位寄存器，以保持相同的操作速度。为了能在通用处理器上实施，由于可以以固定的时间来对更大的移位寄存器进行更新，可以采用执行与多个移位寄存器的功能相似的功能的单个更大的移位寄存器(不降低整体速度)。

采用变量反馈多项式在一个寄存器上以不可预计的方式进行变更可以给出良好的结果。不同的多项式可以以随机的顺序互换，也可以以随机的方式来改变多项式。如果设计得当，这种技术的实现可能会很简单。

II. 更大阶数有限域元素的运算操作

伽罗瓦域 GF(2^n)含有 256 个元素。伽罗瓦域 GF(2^8)的元素可以以几种不同方式中的一种方式来表述。一种共同的并且是标准的表述是从具有阶数小于 8 的所有多项式的系数模 2 中形成所述数据域。即，数据域的元素可以用具有数据位($a_7, a_6, \cdots a_0$)的字节来表述，代表下面的多项式：

$$a_7x^7 + a_6x^6 + \cdots + a_1x + a_0 \qquad (2)$$

数据位也称为是多项式的系数。可以通过相应系数($a_7, a_6, \cdots a_0$)中每一个系数加法模 2 来执行由等式(2)所代表的两个多项式的加法运算。换句话说，通过对两个字节进行异或计算，可以实现对两个字节的加法运算。加法恒等式是多项式具有全 0 的系数(0，0,…,0)。

通过将普通的多项式与模 2 系数相乘，可以在数据域中进行乘法运算。然而，阶数为 n 的两个多项式相乘给出一个阶数为(2n-1)的多项式，该多项式需要减小为阶数为 n 的多项式。在典型的实施例中，通过将合成多项式除以一个不可约多项式，去掉商，并且保留余数作为约减的多项式，可以来实现多项式的约减。不可约多项式的选择改变了将编组内元素变换为存储器中的被编码的字节，但不会影响实际编组的运算操作。在典型的实施例中，选择的阶数为 8 的不可约多项式是：

$$x^8 + x^6 + x^3 + x^2 + 1 \qquad (3)$$

也可以采用其他阶数为 8 的不可约首一多项式，这也在本发明的范围内。乘法单位元是($a_7, a_6, \cdots a_0$)=(0,0，…，1)。

在通用处理器上进行多项式乘法和后续的约减运算是较复杂的运算。然而，对于具有中等数量元素的伽罗瓦域，这些运算操作可以通过查询表以及更简单的操作来完成。在本典型实施例中，数据域中(非 0 运算)的相乘可以通过对两个运算对象中的每一个运算对象取对数、加入对数值模 255 并对混合的对数值取指数来完成。可以将约减组合在查询表内。

可以象下面所描述的那样来生成指数表和对数表。首先，确定乘法子编组 GF(2^8)的生成符 g。这时，字节值 g＝2(代表多项式 x)是一个生成符。表 1 中的指数表是 i＝0,1,…，2^8-1 时，值 g^i 的 256 个字节的表。当 g^i(看作是整数)小于 256 时，指数值如表 1 中第一行开头的 8 个表值所示的那样。因为 g＝2，所以表中的每一个表值是相邻左边的表值的 2 倍(考虑到表 1 中绕回到下一行)。但是，当每一 g^i 大于 255 时，指数由等式(3)中所给出的不可约多项式约减。例如指数 x^8(第一行，第九列)由不可约多项式 $x^8+x^6+x^3+x^2+1$ 约减，而产生余数$-x^6-x^3-x^2-1$。该余数等效于模 2 运算的 $x^6+x^3+x^2$，在表 1 中表示为 77($2^6+2^3+2^2+1$)。重复该过程，直到当所有的指数 i=0 至 255 都计算了 g^i 为止。

在定义了指数表以后，可以计算对数表，作为指数表的逆。在表 1 中，对于采用不可约多项式得到的每一个指数 i，有一个唯一的指数值 g^i 的对应关系。对于表 1，映射是 i_2^i，所以第 i 个位置上储存的值是 2^i。对两边取 $\log_2$对数，得到下面的结果：$\log_2(i)\_i$。这两种变换表示，如果指数表中第 i 个位置上是用作对数

表的指数的，那么该指数的对数是指数表的索引。例如，对于 i＝254，如表 1 中最后一行、第 5 列所示的那样，指数值 $2^i=2^{254}=166$。两边取 $\log_2$ 对数，得到 $254=\log_2(166)$。所以，对数表中索引 i=166 中的值设置为 254。重复该过程，一直到变换了对数表中所有的值为止。0 的对数是一个未定义数。在本典型实施例中，0 用作占位符。

在定义了指数表以及对数表以后，数据域中(非零元素)的乘法可以通过查询对数表中的两个运算对象的对数、用模 255 将对数值相加以及通过查询指数表对混合的对数值取指数幂来进行。因此，数据域中的相乘可以通过三次查询操作和一次舍位相加来进行。在典型的伽罗瓦域 $GF(2^8)$ 中，每一表格是 255 个字节长，并且可以预先计算和存储在存储器内。在本典型实施例中，对数表的 0 位置是不使用的，从而无需从索引中减去 1。注意，当无论哪一个运算对象是 0 时，对数表中相应的值并不代表实际值。为了给出正确的结果，在按照所述的方法进行乘法之前，必须对每一运算对象进行测试看看是不是 0，如果是 0，则结果就是 0。

为了用递归关系从线性反馈移位寄存器中产生输出元素，由于系数 C_j 在等式(1)中是常数，那么情况就更简单了。为了进行有效的实施，只要有可能，就应当将系数选择为 0 或 1。当 C_j 是除 0 或 1 以外的一些值时，对于乘法 $t_i=C_j \cdot i$，可以预先计算一个表格，这时，$i=0,1,2,\cdots,2^8-1$。这时，乘法运算可以用一次表格查询来进行，而无需进行测试。这样的表格是固定的，并且可以存储在只读存储器内。

表1—指数表

i	xx0	xx1	xx2	xx3	xx4	xx5	xx6	xx7	xx8	xx9
00x	1	2	4	8	16	32	64	128	77	154
01x	121	242	169	31	62	124	248	189	55	110
02x	220	245	167	3	6	12	24	48	96	192
03x	205	215	227	139	91	182	33	66	132	69
04x	138	89	178	41	82	164	5	10	20	40
05x	80	160	13	26	52	104	208	237	151	99
06x	198	193	207	211	235	155	123	246	161	15
07x	30	60	120	240	173	23	46	92	184	61
08x	122	244	165	7	14	28	56	112	224	141
09x	87	174	17	34	68	136	93	186	57	114
10x	228	133	71	142	81	162	9	18	36	72
11x	144	109	218	249	191	51	102	204	213	231
12x	131	75	150	97	194	201	223	243	171	27
13x	54	108	216	253	183	35	70	140	85	170
14x	25	50	100	200	221	247	163	11	22	44
15x	88	176	45	90	180	37	74	148	101	202
16x	217	255	179	43	86	172	21	42	84	168
17x	29	58	116	232	157	119	238	145	111	222
18x	241	175	19	38	76	152	125	250	185	63
19x	126	252	181	39	78	156	117	234	153	127
20x	254	177	47	94	188	53	106	212	229	135
21x	67	134	65	130	73	146	105	210	233	159
22x	115	230	129	79	158	113	226	137	95	190
23x	49	98	196	197	199	195	203	219	251	187
24x	59	118	236	149	103	206	209	239	147	107
25x	214	225	143	83	166					

表 2—对数表

i	xx0	xx1	xx2	xx3	xx4	xx5	xx6	xx7	xx8	xx9
00x	0	0	1	23	2	46	24	83	3	106
01x	47	147	25	52	84	69	4	92	107	182
02x	48	166	148	75	26	140	53	129	85	170
03x	70	13	5	36	93	135	108	155	183	193
04x	49	43	167	163	149	152	76	202	27	230
05x	141	115	54	205	130	18	86	98	171	240
06x	71	79	14	189	6	212	37	210	94	39
07x	136	102	109	214	156	121	184	8	194	223
08x	50	104	44	253	168	138	164	90	150	41
09x	153	34	77	96	203	228	28	123	231	59
10x	142	158	116	244	55	216	206	249	131	111
11x	19	178	87	225	99	220	172	196	241	175
12x	72	10	80	66	15	186	190	199	7	222
13x	213	120	38	101	211	209	95	227	40	33
14x	137	89	103	252	110	177	215	248	157	243
15x	122	58	185	198	9	65	195	174	224	219
16x	51	68	105	146	45	82	254	22	169	12
17x	139	128	165	74	91	181	151	201	42	162
18x	154	192	35	134	78	188	97	239	204	17
19x	229	114	29	61	124	235	232	233	60	234
20x	143	125	159	236	117	30	245	62	56	246
21x	217	63	207	118	250	31	132	160	112	237
22x	20	144	179	126	88	251	226	32	100	208
23x	221	119	173	218	197	64	242	57	176	247
24x	73	180	11	127	81	21	67	145	16	113
25x	187	238	191	133	200	161				

III.存储器结构

在采用硬件来实施的时候，使数据位移位是一个简单而有效的操作过程。采用处理器和比起处理器的寄存器更大的移位寄存器使得使数据位移位是一个非常低效的递归过程。当要移位的单元是字节或字的时候，移位就更简单了，这是因为字节之间是没有进位的。但是，移位过程仍然是一个递归、低效的过程。

在典型的实施例中，线性反馈移位寄存器是用循环的缓存器或滑动窗口来实现的。图 3A 和 3B 中分别示出了在 n 时刻和 n+1 时刻循环缓存器 24a 中内容的图。对于循环缓存器 24a，移位寄存器中的每一个元件是存储在存储器中的相应的位置处的。一个索引或指示符 30 保留了图 3A 中存储器如 S_{k-1} 中储存的最新元素的储存位置。如图 3B 所示，在时刻 n+1，计算新的元素 S_K，并存储在存储器中最老的元素 S_0 处。因此，所进行的不是使存储器中的所有元素移位，而是将指示符 30 移位到新元素 S_k 的储存位置上。当指示符 30 到达循环缓存器 24a 的末端时，将其复位到开头的地方(如图 3A 和 3B 所示)。这样，循环缓存器 24 就象是一个圆，而不是一条直线。

如图 3A 和 3B 所示，循环缓存器 24a 可以从左到右移位，或者从右到左移位。相应地，如图 3A 和 3B 所示，指示符 30 从左到右移动，或者从右到左移动。移位方向的选择是一种实施风格，并不会影响输出结果。

为了按照递归关系产生输出元素，通常要求从存储器中得到一个以上的元素。可以用一个单独的指示符来表示与每一所要求的元素相关的储存位置，而指示符在使寄存器移位时被更新。也可以根据需要，从指示符 30，计算与每一所需元素相关的储存单元。由于每一元素与储存单元有一个一一对应的变换关系，通过(按照递归关系)确定该元素与最新元素的偏差、在指示符 30 中加入该偏差，以及对被更新的指示符所表示的储存单元进行寻址，可以得到某一特定的元素。由于存储器的循环特性，被更新的指示符的计算是通过对指示符 30 的偏差的相加模 k 来确定的。当 k 是 2 的幂时否则在处理器上会是一种低效运算操作。

在本较佳实施例中，移位寄存器是用如图 3C 所示的滑动窗口来实现的。滑动窗口 24b 的长度至少是循环缓存器 24a 的两倍，并且包含相邻排列的两个循环缓存器 32a 和 32b。循环缓存器 32a 和 32b 中的每一个就象是上述循环缓存器 24a。循环缓存器 32b 完全与循环缓存器 32a 相同。在正常操作时，缓存器 32b 含有一些有意义的值。接着，根据缓存器 32b 中的值，计算缓存器 32a 中储存的值。因此，移位寄存器中的每一个元素存储在存储器中的两个相应的位置处，循环缓存器 32a

和 32b 各有一个。指示符 34 保留了循环缓存器 32a 中储存的最新元素的储存位置，即，图 3C 中的 S_{k-1}。在本典型实施例中，指示符 34 在滑动窗口 24b 的中间部分开始，从右移动到左，并且在到达左边结束部分的时候再恢复到中间位置。

从图 3C 可以看到，不管指示符 34 出现在循环缓存器 32a 的什么地方，前面的 k－1 个元素可以被寻址至指示符 34 的右边。因此，为了按照递归关系对移位寄存器中的某一元素进行寻址，将 k－1 或更小的偏差加到指示符 34 上。由于被更新的指示符总是指向指示符 34 的右侧的，并且具有一定的计算效率，所以无需进行加法模 k。对于这种结构，滑动窗口 24b 的长度任意，至少是循环缓存器 24a 的两倍，并且忽略所有多余的字节。另外，更新时间是固定的，并且较短。

IV. 基于 GF(2^n)上 LFSR 的典型数据流密码

可以通过基于 GF(2^8)上的线性反馈移位寄存器的典型数据流密码发生器来很好地描述本发明。下面描述的数据流密码采用上述在阶数为 8 的伽罗瓦域上的字节运算，在伽罗瓦域上，分别具有相加和相乘的_和_的表述。在本典型实施例中，表格查询是用于与常数 C 进行相乘的。在本典型实施例中，滑动窗口用来进行移位寄存器的快速更新。

典型发生器的方框图如图 4 所示。在本典型实施例中，线性反馈移位寄存器 52 是 17 个 8 位字节(或 136 个数据位)长，使得移位寄存器 52 能够有 $2^{136}-1$(或者约为 8.7×10^{40})个状态。整个寄存器为 0 的状态不是一种有效的状态，不会在任何一种状态中出现这种状态。以特定的非零元素数，按递归关系更新寄存器 52 的时间是常数，与寄存器 52 的长度无关。所以，可以在存储器中额外字节的标定成本下，来实现添加寄存器 52 的长度(对于更高阶的递归关系来说)。

在本典型实施例中，按照下面的递归关系来更新线性反馈移位寄存器 52：

$$S_{n+17} = (100S_{n+9}) S_{n+4} (141S_n) \qquad (4)$$

这里，运算是定义在 GF(2^8)上的，_是由伽罗瓦加法器 58 所表示的两个字节的异或运算，而_是伽罗瓦乘法器 54 所代表的多项式模乘法(见图 4)。在本典型实施例中，对系数 56 的模相乘是如上所述采用对预计算表进行字节表格查询来进行的。在本典型实施例中，用等式(3)所定义的不可约多项式来计算多项式模乘法表。将等式(4)中递归关系选择为最大的长度、具有极少的非零系数，从而所使用的移位寄存器元素不同于下述非线性函数所使用的移位寄存器元素。

在本典型实施例中，为了掩饰移位寄存器 52 的线性特征，采用上述两种技术，即扰动和采用非线性函数。采用的其他的非线性技术如下所述。在本典型实施例中，

通过在多个移位寄存器 52 元件上进行非线性运算，来引入非线性特征。在本典型实施例中，用非线性函数将移位寄存器 52 中的四个元件组合起来。典型的非线性函数如下所述：

$$V_n = (S_n + S_{n+5}) \leftrightarrow (S_{n+2} + S_{n+12}) \qquad (5)$$

这里，V_n是非线性输出(或发生器输出)，_是由算术加法器 60 所代表的加法舍位模 256，_是由模乘法器 62 所代表的乘法模 257，并且将在下文中描述。在本典型实施例中，所使用的四个字节是 S_n，S_{n+2}，S_{n+5}和 S_{n+12}，这里，S_n是按照等式(4)中的递归关系而在序列中最早计算的元素。选择这些元素，从而当寄存器移位时，在两个发生器输出的计算中不会使用两个元素。这些元素之间按对计算的距离是明显不同的值。例如，S_{n+12}是不与 S_{n+5}、S_{n+2}以及 S_n组合的，因为它是通过寄存器 52 移位的。这一特性称作是“全正差值组(full positive difference set)”。

简单的字节与结果舍位模 256 相加由数据位之间的进位而在 GF(2^8)中是非线性的。在本典型实施例中，寄存器中的两对元素{(S_n和 S_{n+5})和(S_{n+2}和 S_{n+12})}用加法模 256 组合起来，而产生两个中间结果。但是，由于最小有效位是没有进位输入的，并且仍然是线性组合在一起的，因而加法模 256 是不理想的。

另一种在处理器上可以方便地进行计算的非线性函数是乘法。但是，将普通的乘法舍位成只有一个字节不可能给出很好地结果，这是由于乘法模 256 因结果不是很好分布在数据域中的而不会形成一个编组。对素数 257 取整数模的一组相乘的数据域是可以使用的。这一数据组包含在 1 到 256 范围内的整数，带有分组运算是乘法约减模 257。注意，0 值是不会出现在该编组内的，但 256 值会出现在编组内。在本典型的实施例中，256 值可以用 0 的字节值来表示。

一般说来，处理器可以有效地执行乘法指令，但许多的处理器却不能执行或者说是有效地执行除法或取模运算。所以，模约减 257 可以代表一个性能颈瓶。但是，可以用计算模 2^n，来计算约减模 257，在 n＝8 时在公共处理器上有效的。可以看到，对于在 1 到 $2^{16}-1$ 范围内的值 X(这里，X 是两个第 8 阶运算对象相乘的结果)，约减模 257 可以计算为：

$$X_{257} = X_{256} - \frac{X}{256?_{257}}? \qquad (6)$$

式中，X_{257}是 X 的约减模 257，而 X_{256}是 X 的约减模 256。等式(6)表示，一个 16 位的数的约减模 257 可以通过从 8 个最小有效位(X_{256})中减去 8 个最大有效位(X/256)而得到。相减的结果在－255 和 255 之间，并且可以是负值。如果结果是

负值，则可以通过加上 257 而将其调节到正确的范围内。在另一种实施例中，采用包含 65,536 个元素而每一元素为 8 个数据位宽来执行约减模 257 运算。

两个中间结果的相乘是可以利用的许多非线性函数中的一个。采用查询表，也可以执行其他的非线性函数，如曲折函数或者在将它们组合起来之前对字节值进行置换。本发明指出使用各种非线性函数，以产生非线性输出。

在本典型实施例中，采用扰动来注入附加的非线性关系。可以采用从上述线性反馈移位寄存器的状态得到的非线性输出来重新构筑移位寄存器的状态。这种重新构筑由于不代表发生器输出处的某些状态并且是以不可预计的方式选择的而变得更加困难。在本典型实施例中，采用非线性输出来确定输出数据流中会出现的非线性输出的后续字节。当启动发生器时，第一输出字节用作扰动控制字节。在本典型实施例中，将每一个扰动控制字节分成四个数据位对，首先使用最小有效位。当已经使用了全部四个数据位对的时候，从发生器得到的下一个非线性输出字节用作下一个扰动控制字节，等等。

每一对扰动控制数据位可以取四个值中的一个值。在本典型实施例中，表 3 中列出了每一对值中执行的动作。

表 3

数据对值	发生器的动作
(0,0)	使寄存器循环，但不产生输出
(0,1)	使寄存器循环，并使得与常数$(01101001)_2$进行异或的非线性输出变成发生器的输出。使寄存器再次循环。
(1,0)	使寄存器循环两次，并使非线性输出变成发生器的输出。
(1,1)	使寄存器循环，并使与常数$(11000101)_2$进行异或的非线性输出变成发生器的输出。

如表 3 所示，在本典型实施例中，当数据对值是(0,0)时，使寄存器循环一次，但不产生输出。寄存器的循环表示按照等式(4)进行下一个序列输出的计算，并使新的元素移位到寄存器内。随后，用下一个扰动控制对确定下一步要进行的动作。

在本典型实施例中，当数据对值是(0,1)时，使寄存器循环，并按照等式(5)来产生非线性输出。使非线性输出与常数$(01101001)_2$异或，并提供结果作为发生器输出。接着，使寄存器再次循环。图 4 中，由异或门 66 执行异或功能处理，并由多路复用器(MUX)64 用来自缓存器 70 的扰动控制数据位对来选择常数。将异或

门 66 的输出提供到转换器(switch)68，由它按照扰动控制数据位对的值提供发生器输出和用于扰动控制的输出字节。扰动控制的输出字节被提供到缓存器 70。

在本典型实施例中，当数据位对值是(1,0)时，使寄存器循环两次，并给出按照等式(5)产生的非线性输出，作为发生器输出。

在本典型实施例中，当数据对值是(1,1)时，使寄存器循环，并按照等式(5)产生非线性输出。接着使非线性输出与常数$(11000101)_2$进行异或，给出结果作为发生器输出。

在本典型实施例中，选择用在上述步骤中的常数，使得当产生发生器输出时，使输出中有一半的数据位相对于其他扰动控制对产生的输出被交换。对于扰动控制对(1,0)，可以将非线性输出看作是与常数$(000000000)_2$的异或。所以，任意三个常数之间的海明 Hamming 距离是 4。数据位变换还屏蔽了发生器的线性关系，并阻碍了根据发生器的输出对状态进行的重新构筑。本发明支持多层密钥控制结构。支持多层密钥控制结构的数据流密码特别适用于数据是在帧中发送出去的无线通信系统，这些数据帧有可能在接收中出现错误或失序。下面将描述典型的二层密钥控制结构。

在本典型实施例中，采用一个密钥来使发生器初始化。使用密钥来使发生器在序列中有一个不可预计的跳跃。在本典型实施例中，密钥的长度是 4 到 k－1 个字节(或者对于阶数为 17 的典型递归关系为 32 到 128 个数据位)。最好不要采用小于 4 个字节的密钥，这是因为初始随机化是不合适的。也可以采用大于 k－1 个字节的密钥，但这样有些浪费，并且必须小心，使得密钥的值不会使寄存器状态被设置成全 0，这是一种不会随当前限制而出现的状态。

典型密钥初始化过程的流程图如图 5 中所示。过程在方框 110 处开始。在本典型实施例中，在方框 112 处，开始时，用斐波那契数模 256 来使移位寄存器的状态初始化。所以，分别用 1, 1, 2, 3, 5, 8 等等，使元素 $S_0, S_1, S_2, S_3, S_4, S_5$等等初始化。尽管采用的是斐波那契数，但可以采用在伽罗瓦域中不是线性相关的任何一组非零数来使寄存器初始化。这些数不应当再具有可以用来重新构筑寄存器的状态的线性关系。

接着，在方框 114 处，将循环数 n 设置成零。接着，密钥初始化进入一个循环。在该循环的第一个步骤中，即，在方框 116 处，将密钥材料中第一个不用的字节加到 S_n上。加上密钥材料使得发生器在序列中进行了一次不可预计的跳跃。接着，在方框 118 处，使密钥移位一个字节，从而删去方框 116 中所使用的字节。随

后，在方框 120 处，使寄存器循环。方框 116 和 120 的组合有效地进行下面的计算：

$$S_{n+17} = (100S_{n+9})S_{n+4}(141(S_nK)) \quad (7)$$

这里，K 是密钥材料第一个不用的字节。循环数 n 在方框 122 处递增。接着，在方框 124 处，判断是否使用了所有的密钥材料。如果回答是否，那么过程就回到方框 116。否则，过程继续进行到方框 126。

在本典型实施例中，在方框 126 处，将密钥的长度加到 S_n 上。密钥长度的添加使得发生器在序列中出现另外的跳跃。随后，过程进入第二个循环。在第二个循环中的第一个步骤中，在方框 128 处，使寄存器循环。循环指数 n 在方框 130 处递增，并且在方框 132 处，将其与发生器的阶数 k 比较。如果 n 不等于 k，那么过程就回到方框 128。否则，如果 n 等于 k，那么过程就继续进行到方框 134，将发生器的状态储存起来。接着，过程在方框 136 处终止。

除了密钥以外，本发明中还可以使用第二密钥。第二密钥不能被看作是保密的，但却是用在典型的无线电话系统中产生每一数据帧的唯一的密码。这就确保了擦除的帧或失序的帧不会使信息流中断。在本典型实施例中，数据流密码以 4 个 8 位码无符号的整数的形式来接受每一帧密钥，称作是帧密钥。每一帧密钥初始化与上述密钥初始化的相似的，但却是为每一数据帧而进行的。如果使用数据流密码使得没有必要使用每一帧密钥信息，例如，对于一可靠线路上的文件转发，那么每一帧初始化过程是可以略去的。

采用密钥的典型每一帧初始化过程的流程图如图 6A 所示。该过程在方框 210 处开始。在本典型的实施例中，在方框 212 处，用上述从密钥初始化过程中储存的状态，来使发生器的状态初始化。接着，在方框 214 处，将循环指数 n 设置成 0。随后，使每一帧初始化过程进入一个循环。在循环中的第一个步骤中，在方框 216 处，帧密钥的最小有效位被模 256 加到 S_n 上。接着，在方框 218 处，使帧密钥移位三个数据位，从而删除在方框 216 中使用的三个最小有效位。接着，在方框 220 处，使寄存器循环。在本典型实施例中，使循环指数 n 在方框 222 处递增，并在方框 224 处与 11 比较。方框 224 中所使用的 11 值对应于用作帧密钥的 32 个数据位以及帧密钥一次移位三个数据位这样一个事实。帧密钥的不同选择以及一次移位不同数量的数据位会产生方框 224 中使用的不同的比较值。如果 n 不等于 11，那么过程就回到方框 216。相反，如果 n 等于 11，那么过程就继续进行到方框 226，并且使寄存器继续循环。在方框 228 处使循环指数 n 递增，并在方框 230 处与 2k 比较。如果 n 不等于 2k，那么过程就回到方框 226。相反，如果 n 等于 2k，那么过

程就在方框 232 处中止。

上文中已经针对具有 256 个元素的典型伽罗瓦域描述了本发明。也可以采用不同的有限域，从而使元素的多少与用于计算元素的处理器以及/或者构成移位寄存器的存储器的字节或字的长短一致起来，或者还具有其他的优点。因此，可以采用具有两个以上元素的各种有限域，这也是属于本发明的范围的。

上述例子采用各种非线性处理器来屏蔽递归关系的线性关系的。可以采用不同的非线性处理过程，或上述非线性处理过程与其他的非线性处理过程的组合，来设计其他的一些发生器。因此，可以设计采用各种非线性处理过程来产生非线性的输出，这也属于本发明的范围。

上述例子采用的是具有阶数为 17 以及由等式(4)定义的递归关系。也可以产生具有其他阶数的递归关系，这也属于本发明的范围。此外，对于给定的阶数，可以产生各种不同的递归关系，这同样属于本发明的范围。本发明中，最长长度的递归关系将给出最佳结果。

V. 基于 GF(2^8)上 LFSR 的第二种典型数据流密码

递归关系和非线性函数都对移位寄存器的元素进行访问。仅选择访问哪一个元素，从而元素之间的距离形成“全正差值组”(“On Security of Nonlinear Filter Generators”, J. Dj. Golic, in Proceedings of Fast Software Encryption 1996 Cambridge Workshop, Springer-Variag 1996)。随后，在递归关系和非线性函数之间划分这些元素，以使每一个的扩展程度为最大。在这些限制下，可以进一步将本发明发展成增强密码的保密性和计算的有效性。与第一种典型的实施例相比，第二种典型实施例提供了一种改进的密码保密性。

在数学上，在 GF(2^8)上的 LFSR 等效于在长度为 136 的 GF(2)上的八个移位寄存器，每一个移位寄存器具有相同的递归关系。本发明的典型实施例包括在 GF(2^8)上的递归关系，它等效于一种特征多项式具有 51 个非零系数的二进制递归关系。递归关系中的三个抽头位置是由上文中给出的标准来决定的(即，“全正差值组”)。

在理想情况下，GF(2)上阶数为 136 的多项式，对于密码分析和最大值扩散的最佳长度应当近似具有其一半的系数为 1。在 GF(2^8)上有许多的多项式，有三个系数接近于这一目标，但所有三个系数都大于 1。这就是说，采用这些多项式需要三个查询表和参照对象(reference)，其效用与本发明的当前结构

相比较差。然而，这些多项式在理论保密性的基础上是完全可以接受的。

以获取可能的等效二进制多项式为目标，同时保留系数为1的当前结构(避免了乘法表和查询)，分析表明，使用65个非零二进制系数可以给出近乎实现68个非零系数目标的较佳实施例。在GF(2^8)上满足这些标准的多项式有16个。在GF(2^8)上总是按8个多项式编组，它们具有相同等效二进制多项式；这些多项式字节中的数据位发生了位置偏移。(可以找到每一等效的二进制多项式，例如，通过Berlekamp-Massey规则。)所以，如表4所示，有两种类型完全不同的多项式满足这一标准。对于本发明第二种典型的实施例，采用表4中第一组系数。

表4—递归系数

S_n	S_{n+4}	S_{n+15}	类型
99	1	206	1
106	1	201	1
142	1	126	1
148	1	214	1
203	1	146	1
210	1	19	1
213	1	195	1
222	1	136	1
40	1	109	2
45	1	38	2
46	1	159	2
57	1	129	2
110	1	209	2
117	1	63	2
32	1	219	2
140	1	97	2

第二种典型发生器的方框图如图7中所示。在本典型实施例中，线性反馈移位寄存器82是17个八数据位长。当然，也可以采用其他长度的寄存器82(用于不同阶数的递归关系)，这也属于本发明的保护范围。阶数为17的递归关系很适合于采用多达128个数据位密钥材料的应用场合。在本典型实施例中，按照下面的递归关系来更新线性反馈移位寄存器82：

$$S_{n+17}=(206\otimes S_{n+15})\oplus S_{n+4}\oplus(99\otimes S_n) \qquad (8)$$

⊕

式中的运算是定义在 GF(2^8)上的，⊕是由伽罗瓦加法器 88 所代表的两个字节上的异或运算，并且⊕是由伽罗瓦乘法器 84 所代表的多项式模乘法器(见图 7)。在本典型实施例中，对系数 86 的模乘法是通过对上述预先计算的表格进行字节表格查询来进行的。将等式(8)中的递归关系选择为最大长度。

在本典型实施例中，为了掩饰移位寄存器 82 的线性关系，采用上述两种技术，即，扰动和采用非线性函数。本说明书中还描述了其他的非线性技术。

在本典型实施例中，用相对于在 GF(2^8)的线性运算是非线性的函数(或输出等式)，通过将移位寄存器 82 的四个元素组合起来，引入非线性关系。在本典型实施例中，所使用的四个字节是 S_n, S_{n+2}, S_{n+5} 和 S_{n+12}，这里，S_n 是按照等式(8)中的递归关系而在序列中最早计算的元素。

本发明的许多密码保密性是由于采用了非线性函数而战胜了对扰动相位的攻击而得到，只要尽可能地非线性就可使该函数更强。

可以尝试各种可能的函数，用以将非线性函数与最接近的线性近似按位比较，并计算与 0.5 的平均绝对偏差和均方根偏差，这在理论上是一个完美的结果。研究表明，最优秀的技术方案是从旋转部分和得到的，这是一个在高阶数据位中带有影响的过程，从而将这些数据位与其他元素的最小有效位组合起来。

在微处理器上，加法函数通常一次仅接受两个运算操作，因而最好的策略是在一次中间加法以后进行旋转。将旋转运算操作记为 ROTL(x)，这意味着使 x 的数据位左移一个位置，因而更优越的非线性函数是：

$$V_n = ROTL(S_n + S_{n+2}) + S_{n+5} + S_{n+12} \qquad (9)$$

这里，V_n 是非线性输出，而＋是由算术加法器 90 所代表的加法舍位模 256(舍弃溢出)。ROTL 表示旋转算符 91。

加上 S_{n+5} 以后的附加旋转似乎不会产生更好的结果。正如本说明书中所讨论的那样，采用明显为非线性排列的查询表给出了另一种方法，但这种方法明显会有损本发明的计算效率。

在本典型实施例中，用于递归关系(8)的字节包含 S_n，S_{n+4} 和 S_{n+15}，而用于输出等式(9)的字节包含 S_n，S_{n+2}，S_{n+5} 和 S_{n+12}。在本典型实施例中，选择这些字节，使之具有特有的数据对距离。对于递归关系等式(8)，所使用的三个字节具有的数据对距离是 4(S_n 和 S_{n+4} 之间的距离)、11(S_{n+4} 和 S_{n+15} 之间的距离)和 15(S_n

和 S_{n+15}之间的距离)。同样，对于输出等式(9)，所使用的四个字节所具有的距离是2(S_n和S_{n+2}之间的距离)、3(S_{n+2}和S_{n+5}之间的距离)、5(S_n和S_{n+5}之间的距离)、7(S_{n+5}和 S_{n+12}之间的距离)、10(S_{n+2}和 S_{n+12}之间的距离)和 12(S_n和 S_{n+12}之间的距离)。递归关系(8)中的数据对距离(即，4,11 和 15)是在第一编组中唯一的，并且输出等式(9)中的数据对距离(即，2,3,5,7,10 和 12)在第二编组内也是特有的。另外，递归关系(8)中的数据对距离与输出等式(9)中的数据对距离也是明显不同的。明显不同的数据对距离确保了当移位寄存器 82 移位时，无论是在递归关系(8)中还是在非线性输出等式(9)中都不会两次使用移位寄存器 82 中特定的元素对。这一特性去掉了后续输出等式(9)中的线性关系。

在本典型实施例中，图 7 中所示的多路复用器(MUX)92、XOR 门 94、交换器(switch)96 以及缓存器 98 是以与图 4 中的 MUX64、XOR 门 66、转换器 68 以及缓存器 70 相同的方式工作的。

第二种典型的按帧初始化处理过程的流程图如图 6B 所示，它是图 6A 所示流程图的修改形式。

本实施例在第二密钥装载过程中采用非线性函数，从而以比以前更快的速度混合密钥信息，使得在产生输出之前进行的混合运行更短。这一特征防止了寄存器的状态形成寄存器总的状态组的线性子空间。

将密钥字节加到寄存器第 5 个字节上，而不是第 0 个字节上，而使扩展速度加快，这是递归关系元素之一。当加载“数据帧”时，一次放入 8 个数据位。除了进入来自“数据帧”的 8 位数据位以外，这一方法还在寄存器的第 8 个字节中加入了来自“nltap”的输出。在“数据帧”已被加载以后，这种方法继续使寄存器循环，并加入一些用于某些循环的输出。

因此，将图 6B 与图 6A 相比，方框 218 被修改，从而使该数据帧移位了 8 个数据位，去掉了 8 个最小有效数据位。新的方框 219 加入了来自非线性函数的输出。并且最终方框 224 中值的检验从 11 变到了 4。

VI. 基于 GF(2^8)上 LFSR 的第三种典型的数据流密码

如上所述，可以进一步将本发明发展以加强密码的保密性和计算效率，同时保持“全正差值组”。第三种典型实施例与第一种典型的实施例相比，使计算效率得到提高。

可以采用更简单的递归关系，其方法是采用更简单的二进制等效多项式，这样将使密码分析更为简单。首先，给定全正差值组的约束，使 S_{n+4}的系数都

是 1，就可以避免乘法表和相应的表格查询。有 8 种这样的递归，有相同的等效二进制多项式，含有 35 个非零系数。这些 S_n 的系数分别是：40,45,46,57,110,117,132 和 140。

如果允许某些内部系数是零，则可以有更简单的多项式。这时，去除的不仅可以是乘法，也可以是额外项的整个标记(reference)。有 32 个这样的递归关系；其中的 8 个是具有 11 个非零系数的等效二进制多项式，而其余的 24 项是具有 13 个非零系数的三个等效的二进制多项式。这些多项式中，8 个多项式具有与 S_{n+1} 项相关的 1 系数，而其余的 16 项使之与 S_{n+4} 项相关。前 8 个的等效二进制多项式看上去比其他的具有更加“扩展”的非零系数，因而对于本发明最短时间来说，将采用这样的递归关系。S_n 项的系数可以是 79,83,166,187,225,239,243 和 252 中任何数。对于本发明的第三种典型的实施例来说，采用第一个系数。因而递归关系变成：

$$S_{n+17} = 79S_n + S_{n+15} \tag{11}$$

在普通 8 位微处理器上，对移位寄存器元素的标记相当烦琐。似乎可以完全去掉这些标记之一，也不会对保密性有太多的影响。选择去掉元素 S_{n+2}，尽可能地使这些值“扩展”。然而，最好使中间和旋转，这是因为最小有效位的非线性关系不象所希望的那样好。事实上，这时的最佳旋转有四个位置。许多微处理器在实现这一操作时是执行“nybble－swap(半字节对换)”指令的。采用标记 SWAP()以表示使字节旋转 4 个位置，非线性函数变成：

$$V_n = SWAP(S_n + S_{n+5}) + S_{n+12} \tag{12}$$

第三种典型实施例发生器的方框图如图 8 所示。在本典型实施例中，线性反馈移位寄存器 102 的长度是 17 个 8 位数据位，当然，也可以采用其他长度的寄存器 102(用于不同阶数的递归关系)，并且这也属于本发明的范围内。阶数为 17 的递归关系很适合于采用多达 128 个数据位的密钥材料的应用场合。在本典型实施例中，线性反馈移位寄存器 102 是按照下面的递归关系(11)来更新的，这时的运算是定义在 $GF(2^8)$ 上的，⊕是对伽罗瓦加法器 108 所代表的两个字节进行的异或运算，而⊗是对伽罗瓦乘法器 104 所代表的两个字节进行的多项式模乘法运算。本典型实施例中，对系数 106 进行的模乘法运算是对上述预计算表格进行字节表格查询来进行的。可以将等式(11)中的递归关系选择成是最大长度。

这里，V_n 是非线性输出，而＋是算术加法器 110 所代表的加法舍位模 256

计算(舍弃溢出)。SWAP 表示对换算符 111。

在本典型实施例中，图 8 中的转换器 116 和缓存器 118 是以上述图 4 中的转换器 68 和缓存器 70 所描述的方式工作的。

在扰动期间，在两种情况下，非线性输出是与常数项进行异或的。(见表 3)在本例中，这些计算被取消。

VII. 具有实时条件的系统中采用抽取数据流密码发生的方法

上述对数据进行加密的方法和装置是许多数据流密码的一个子集，它在设计中含有“扰动”，用于保密。其他经常使用的项是“不规则抽取”或“时钟控制”。其意义是，以假定很难预计的方法，通过忽略某些更大密钥数据流的元素来生成输出密钥数据流。在上述算法中，平均基础移位寄存器有 7 次循环会产生 3 个 8 位数据位的输出。其他在本领域中众所周知的抽取发生器是收缩发生器(见 D. Coppersmith 等人的“收缩发生器”，*Proc.Crypto '93,Springer-Verlag, 1994*)以及“自收缩发生器”(W.Meier 和 O.Staffelbach 的“自收缩发生器”，*Communications and Cryptography: Two sides of one tapestry,* R.E.Blahut 等人，eds,Kluwer Academic Publishers, 1994)。

在移动电话应用中，经常采用数据流密码来对相对较小的数据帧进行加密。移动站(蜂窝电话)通常有一个计算能力相对较小的计算机，以降低成本和功耗。移动站在将数据帧发送出去或者收听者检测到声音的“丢失”的时机之前，有一个固定的时间进行加密或解密。

抽取上下文中的数据流密码的问题是，它们对数据进行解密所花费的时间(或者说是输出密钥数据流的生成速率)是可变的。当数据量较大时，生成相同数量的密钥数据流的总的时间统计变化不是很明显的。但是，当每一数据帧中的数据量相对较小的时候(即，与上述线性反馈移位寄存器的尺寸具有相同的数量级的时候)，输出中生成率的变化就很明显了。这就会产生这样的一种情况，即，移动站没有充分的时间进行加密或解密。

上述算法在最后阶段通过抽取(“扰动”)早先阶段产生的输出数据量来工作的。这是用输出中的一个 8 位数据位(扰动控制字节)确定以后四个阶段的行为来完成的。平均说来，每一这样的字节控制从早先阶段得到的 6 个值的消耗(最少 4 个，最多 8 个)，并产生 3 个输出值(最少 0 个，最多 4 个)。在长时间取平均的基础上对扰动控制字节本身进行计数，上述算法将生成基础寄存器 N 次循环的输出的 3/7N 个 8 位数据位。然而，也会有这样的情况，即，抽取所

引起的延迟会产生明显更大的延迟。

本发明提供了一种方法，即，通过在产生过度延迟之前使抽取过程中断，来限制对数据进行加密所需的时间。在本发明的第一种实施例中，加密系统识别加密时间是什么时候变得过大的，并用在固定速率下产生输出的保密性略弱的方法来产生其余的密钥数据量输出。在另一种实施例中，加密系统将抽取过程限制在预定个数的抽取。

由于这种情况出现的可能性较少，并且出现这些情况对于预计或检测加密的相反过程来说是不可能的，而且少量的输出是用保密性较弱的方法来产生的，所以，输出的加密分析仍然是不采用的。下面的例子给出，对于上面提供的算法，这是怎样来完成的，但这一原理是应用于钟控发生器，如，上面提到的收缩发生器。

该算法中，有两个扰动特征会引起密钥数据量输出产生率的变化。四种情况中有两种情况下，采用两个寄存器循环来产生一个 8 位数据位的输出。在其余的两种情况下，执行一次循环，但仅在一半的时间里产生一个 8 位数据位的输出。所以，在四分之一的情况中，是没有一点输出产生的；而在其余的时间里产生一个 8 位数据位的输出。另一方面，寄存器总循环数的变化控制整个抽取持续时间的变化。考虑到这些因素，有两种不同的方法可以用来检测发生器舍弃太多输出的情况，同时这样花费的时间太长。

1. 初始化以后，可以对发生器不产生任何输出的次数进行计数。当该计数达到某一阈值的时候，算法转向到采用保密性较弱并更可能预计的方法。

2. 初始化以后，计数器会对基础移位寄存器的总循环数进行计数。当该计数器达到一个(不同的)阈值的时候，算法将会相反回到保密性稍差并具有更多可预计性的方法。

上述方法在组合的时候采用了两种机制，以给出保密性。一种是移位寄存器的非线性函数功能。另一种是扰动机制。对于输出量相对较少的情况，即到达几百个 8 位数据位的情况，非线性函数本身将令给出合适的保密性。在本典型的实施例中，对数据进行加密的保密性较差的方法是简单地取消输出扰动，并从非线性函数中为基础寄存器的每一循环产生一个输出。对于其他的数据流密码，需要采用其他的方法。比如，在收缩发生器中生成数据位的保密性较差的方法是采用从时钟控制寄存器得到的输出数据位来选择来自其他寄存器的哪两个连续的数据位将用于输出。尽管该后退方案在由其自身使用时是不保密

的，但在本上下文中还是足够保密的。

上述第一种方法的变异形式是连续地监视没有输出生成的次数，并且无论什么时候，只要有预定次数的循环数没有输出产生，就强迫产生一个输出。例如，当有四次循环中没有输出产生时，就可以生成一个输出。这就保证有一个输出生成率的特别约束，因而在算法的每四次递归中永远不会落到一个输出以下。(可以将这种情形看作是对上文中提到的情形的略微复杂的扰动方法。)

每次结束时必须使密钥数据流发生器具有相同的配置参数，并且在同一时间上转到保密性较弱的方法。

应当选择用于转向的参数，从而使给定数据帧加密强度较小的几率会很小，比方说是 10^4 个中有一个或更少。采用数学方法，可以从给定密码的已知特征得到这些参数，或采用模拟的方法来确定。这同时确保了采用保密性较弱的方法加密的数据量是很小的。在本典型实施例中，在 104 个寄存器循环的情况下，加密系统转向到保密性较弱的密码。采用上述加密系统，将意味着，在 10,000 个数据帧中，只有一个以下的数据帧会转到保密性较弱的加密情况，并且其中，采用保密性较弱的方法，100 中只有约一个会产生 6 个或更多个 8 位的数据位输出(这是根据从 104 个循环到 116 个循环得到产生的 8 位数据位而估计的)。这种情况由加密分析检测到是极为不可能的，这是因为 6 个 8 位数据位中的任何一个数据流代表的是二分之一不到的寄存器状态，这在正常操作运行的随机情况下同样也会发生。

参见图 9，来描述加密过程中对延迟进行限制的加密系统。加密子系统 A 302 执行较佳的加密操作。即，加密子系统 A 302 比起通过加密子系统 B300 来说，具有更大的保密性。但是，与加密子系统 B302 所执行的加密操作相比，加密子系统 B 300 所执行的加密操作具有较少的延迟。

加密子系统 A 302 所提供的有关密钥数据流的信息被提供到超时检测器 306，由该检测器检测产生密钥数据流连续部分的时间，或者在另一些监视器中是产生对数据帧进行加密的累计时间。当从加密子系统 A 302 产生连续部分的密钥数据流的时间或者对一部分数据帧进行加密的累计时间超过了预定的门限值的时候，超时检测器 306 就向转换器 304 发送一个信号，指令转换器 300 应当采用加密子系统 B 300 所提供的密钥数据流，以替代加密子系统 A 320 所提供的密钥数据流。

将转换器 304 选择的密钥数据流提供到混合器 308 的第一输入。混合器 308

将要加密的数据与选择的密钥数据流混合，并输出经加密的数据。

图 10 描述的是图 9 中本发明的特定实施结构。在图 10 实施的特定实施结构中，较佳的加密操作按照线性运算操作、非线性运算操作以及随机抽取操作，产生密钥数据流。当抽取过程预示将会在加密过程引起过度的延迟的时候，那么将转变成退出该操作。

图 10 中，加密过程的线性部分是由线性反馈移位寄存器(LFSR)328 执行的，其结构在本领域中是众所周知的，其结构如上所述。线性反馈移位寄存器 328 将其输出和一组寄存器的内容提供至非线性运算处理器 324。非线性运算处理器 324 对线性反馈移位寄存器 328 的输出以及线性反馈移位寄存器 328 提供的寄存器内容进行非线性处理，并将非线性运算操作的结果输出到抽取器 322 和转换器 326。

抽取器 322 对非线性运算器 324 的输出进行随机化的抽取，由它按照伪随机选择过程，删除非线性运算器 324 提供的一部分数据，并将随机抽取的数据流提供到转换器 326。

在正常操作时，转换器 326 将抽取的密钥数据流从抽取器 322 输出到异或门 330。但是，当抽取控制器 320 检测到抽取过程预示着会在加密数据中引入过度的延迟的时候，抽取控制器 320 向转换器 326 发送一个信号，而由转换器 326 根据输出，输出从非线性运算处理器 324 提供的密钥数据流，而不进行抽取。在本典型实施例中，抽取控制器 320 通过对线性反馈移位寄存器 328 的移位数进行计数，或者通过对抽取器 322 提供的抽取量进行计数，来检测变得过长的抽取过程。由抽取器 322 抽取的线性反馈移位寄存器 328 的移位数或者从非线性运算器 324 的输出数由抽取控制器 320 用来确定何时抽取过程变得过长了。

本实施例的方框图如图 11 所示。图 11 是图 7 中所描述的发生器的修改形式，其中加入了对加密延迟进行限制的装置。在本典型实施例中，线性反馈移位寄存器 382 有 17 个 8 位数据位长。当然，寄存器 382 也可以采用其他的数据长度(用于不同阶数的递归关系)，这也属于本发明的范围。阶数为 17 的递归关系很适合于采用多达 128 个数据位的密钥材料应用场合。在本典型实施例中，线性反馈移位寄存器 382 是按照下面的递归关系来更新的：

$$S_{n+17} = (206 \otimes S_{n+15}) \oplus S_{n+4} \oplus (99 \otimes S_n) \qquad (13)$$

这里，运算是定义在 GF(2^8)上的，⊕是对伽罗瓦加法器 388 所代表的两个

字节进行异或运算，而⊗是对伽罗瓦乘法器 384 所代表的多项式模乘法运算。在本典型实施例中，对系数 386 进行的模乘法运算是通过对上述预先计算的表格进行查询来实行的。将等式(13)中的递归关系选择为最大长度。

在本典型实施例中，为了舍弃移位寄存器 382 的线性关系，采用上述两种技术，即，扰动和采用非线性函数。其他的非线性技术在本说明书的其他部分也有描述。

本典型实施例中，非线性关系是通过将移位寄存器 382 的四个元素采用相对于在 GF(2^8)上的线性运算是非线性的函数(或输出等式)进行合并而引入的。在本典型实施例中，所采用的四个字节是 S_n, S_{n+2}, S_{n+5} 和 S_{n+12}，这里，S_n 在按照等式(13)中的递归关系的序列中是最早计算的元素。

本发明的许多密码保密性是利用非线性函数来防止对扰动阶段产生攻击而得到的，因而要求该函数尽可能是足够非线性的。

发明人尝试了许许多多种可能的函数，从而将非线性函数与每一数据位位置中最接近的线性近似进行比较，并计算与 0.5 的平均绝对偏差和根均方偏差，而“0.5”是在理论上完美的结果。研究表明，优秀的技术方案是通过使部分和旋转而得到的，这是一个在高阶数据位上具有进位效果的过程，从而使这些数据位与其他元素的最小有效位混合起来。

在一个微处理器处，加法运算通常一次仅接受两个运算，从而最好的策略是在一次中间加法以后进行旋转。将旋转运算记为 ROTL(x)，这就是说使 x 的数据位左旋一个位置，那么更好的非线性函数是：

$$V_n = ROTL(S_n + S_{n+16}) + S_{n+1}S_{n+6} + S_{n+13} \qquad (14)$$

这里，V_n 是非线性输出，而＋是由运算加法器 390 进行的加法舍位模 256(舍弃了溢出)运算。ROTL 表示旋转运算 391。

在加上 S_{n+5} 以后进行的附加旋转似乎不会给出更好的结果。正如在本说明书的其他地方所讨论的那样，采用非线性置换的查询表给出了另一种方法，但明显会破坏本发明的计算效率。

在本典型实施例中，用于递归关系(13)的字节包含 S_n, S_{n+4} 和 S_{n+15}，而用于等式(14)的字节包含 S_n, S_{n+1}，S_{n+6}，S_{n+13} 和 S_{n+16}。在本典型实施例中，选择的这些字节具有明显不同的数据对距离。对于等式(13)的递归关系，所使用的这些字节具有数据对距离 4(S_n 和 S_{n+4} 之间的距离)、数据对距离 11(S_{n+4} 和 S_{n+15} 之间的距离)以及数据对距离 15(S_n 和 S_{n+15} 之间的距离)。同样，对于输出等式(14)，

所使用的5个字节具有数据对距离1(S_n和S_{n+1}之间的距离)、3(S_{n+13}和S_{n+16}之间的距离)、5(S_{n+1}和S_{n+6}之间的距离)、6(S_n和S_{n+6}之间的距离)、7(S_{n+6}和S_{n+13}之间的距离)、10(S_{n+6}和S_{n+16}之间的距离)、12(S_{n+1}和S_{n+13}之间的距离)、13(S_n和S_{n+13}之间的距离)、15(S_{n+1}和S_{n+16}之间的距离)和16(S_n和S_{n+16}之间的距离)。递归关系(13)之间的距离(即4,11和15)在第一编组内是唯一的(或特有的)，比起输出等式(14)中的数据对距离(即，1,3,5,6,7,10,12,13,15,和16)在第二编组内也是特有的。另外，递归关系(13)中的数据对距离除了一个数据对距离15以外，在输出等式(14)中是唯一的。特有的数据对距离确保了当移位寄存器382移位的时候，无论是在递归关系(13)中还是非线性输出等式(14)中都不会使用两次。这种特性去除了后续输出等式(14)中的线性关系。

在本典型实施例中，图11中的多路复用器(MUX)392、异或(XOR)门394、转换器396以及缓存器398是以上述参照图4中的MUX64、XOR门66、转换器68以及缓存器70的方式工作的。

来自转换器396的抽取密钥序列和加法器390b的输出被提供到转换器402。在正常操作下，转换器402输出来自转换器396的抽取的密钥序列。然而，如果抽取控制器400检测到加密过程由于抽取过程的可变性而引入了过度的延迟，则抽取控制器400向转换器402发送一个信号，使转换器402输出未抽取的序列。转换器402的输出与数据序列混合，提供经加密的数据。

在本典型实施例中,抽取控制器400通过对加法器390b输出的密钥数据流的某些部分进行选通而对来自缓存器398的输出数进行计数，或者通过对移位寄存器382的移位数进行计数，来检测加密延迟是否过长。也可以采用抽取扰动过程所引入的延迟的其他方法，这些方法同样属于本发明的范围。

加密数据的接收机必须知道什么时候在转换器396输出的抽取密钥序列和加法元件390b输出的未抽取的序列之间转换。在本典型实施例中，接收机还包括相同的抽取控制器400和转换器402，并将以与加密数据的发射机处相同的方式，在抽取和未抽取的序列之间转换。因此，就不必具有抽取的和未抽取的密钥序列之间的时间变换的明确提示了。

第二种典型的按数据帧初始化过程的流程图如图6B所示，它是图6A所示流程图的修改形式。

本实施例在第二密钥装载过程中采用非线性函数，以比以前更快的方式来混合密钥信息，从而使得在产生输出之前的混合时间更短。这一特征防止了寄

存器成为寄存器总状态数的线性子空间。

将密钥字节加到寄存器的第 15 个字节上，而不是加到第 0 个字节上，从而使扩展加速，这是递归关系元素中的一个。当装入“数据帧”的时候，一次输入 8 个数据位。除了加入来自“数据帧”的 8 位数据位以外，本方法还将来自“nltap”的输出加到了寄存器的第 8 个字节上。在已经装载了“数据帧”以后，本方法继续使寄存器循环，并加入某几个循环的输出。

因此，比较图 6B 和 6A，方框 218 被修改，从而数据帧移位了 8 个数据位，以去掉 8 个最小有效位。新的方框 219 加入了来自非线性函数的输出。并且最终在方框 224 中检验的值从 11 变到了 4。

前文中对较佳实施例的描述使得本领域中的普通技术人员能够制造和使用本发明。很明显，本领域中的普通技术人员还能对这些实施例作各种修改，可以无需发明人的帮助，将本文中所描述的基本原理应用于其他的实施例。所以，本发明并非仅限于这些实施例，应当从最宽的范围来理解本发明的原理和新特征。

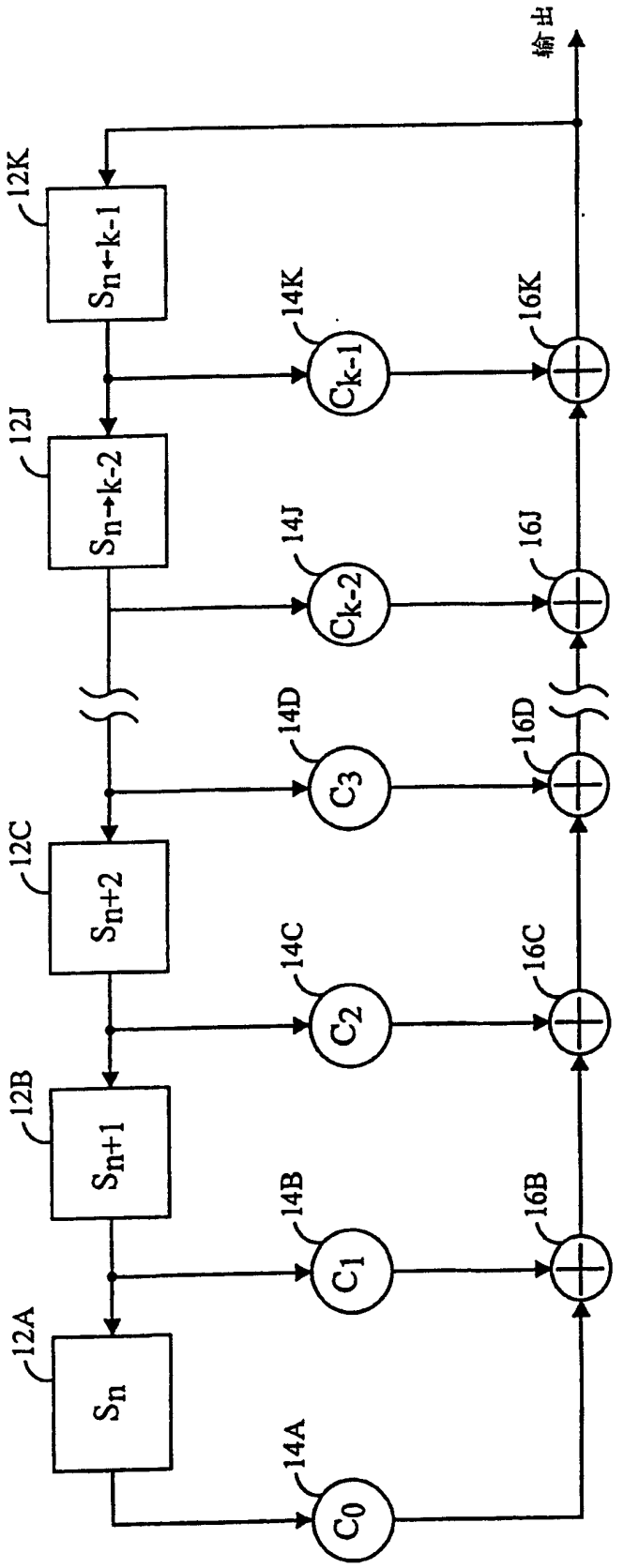

输出
12K
Sn←k-1
14K
Ck-1
16K
12J
Sn←k-2
14J
Ck-2
16J
14D
C3
16D
12C
Sn+2
14C
C2
16C
12B
Sn+1
14B
C1
16B
12A
Sn
14A
C0

图 1

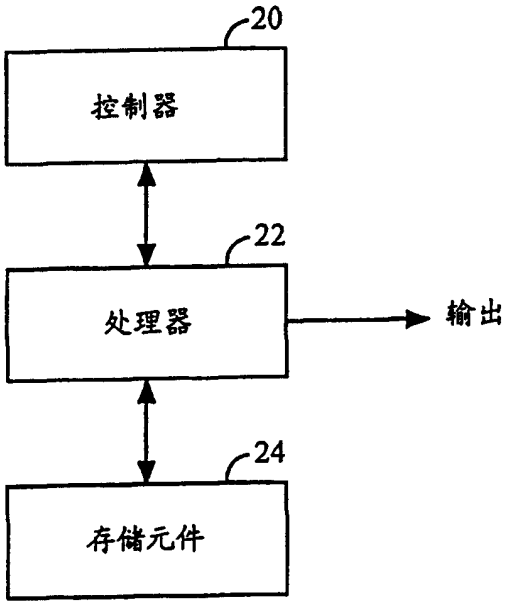


图 2

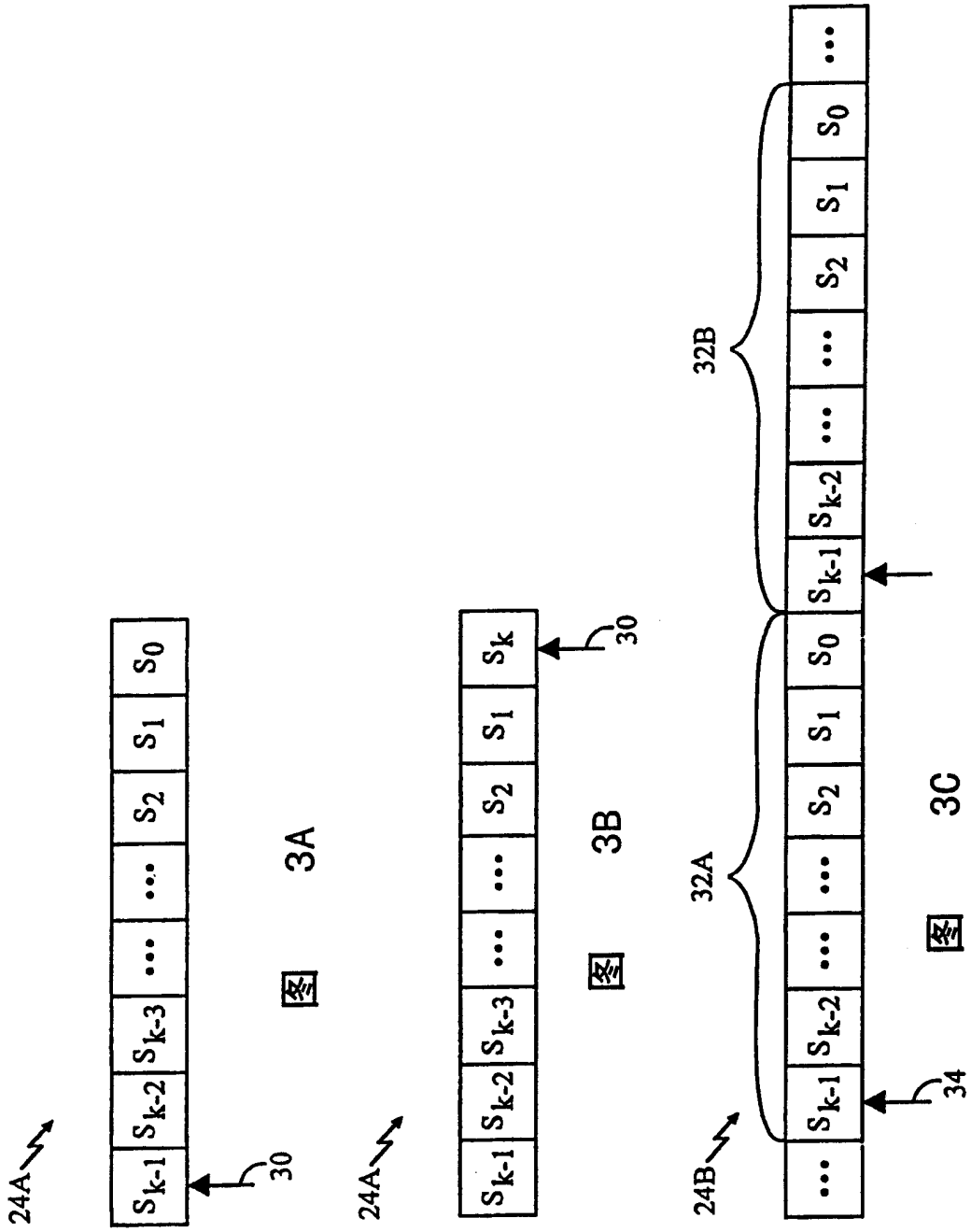

24A
S_{k-1}
S_{k-2}
S_{k-3}
...
...
S_2
S_1
S_0
30
24A
S_{k-1}
S_{k-2}
S_{k-3}
...
...
S_2
S_1
S_k
30
24B
32A
32B
...
S_{k-1}
S_{k-2}
...
...
S_2
S_1
S_0
S_{k-1}
S_{k-2}
...
...
S_2
S_1
S_0
...
34

图 3A 图 3B 图 3C

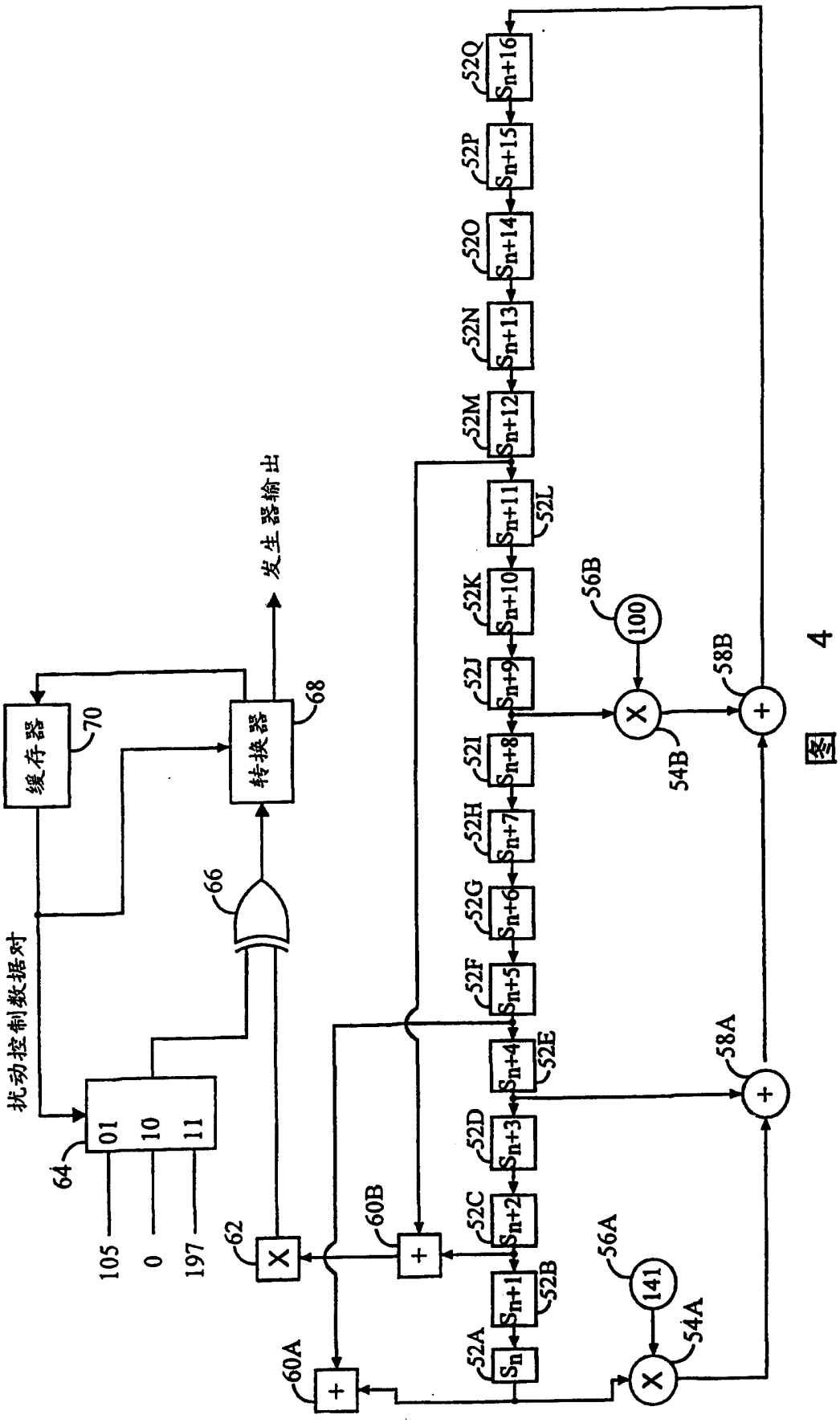

扰动控制数据对
发生器输出
缓存器
70
转换器
68
66
64
01
10
11
105
0
197
62
60A
60B
52A
Sn
52B
Sn+1
52C
Sn+2
52D
Sn+3
52E
Sn+4
52F
Sn+5
52G
Sn+6
52H
Sn+7
52I
Sn+8
52J
Sn+9
52K
Sn+10
52L
Sn+11
52M
Sn+12
52N
Sn+13
52O
Sn+14
52P
Sn+15
52Q
Sn+16
56A
141
54A
56B
100
54B
58A
58B

图 4

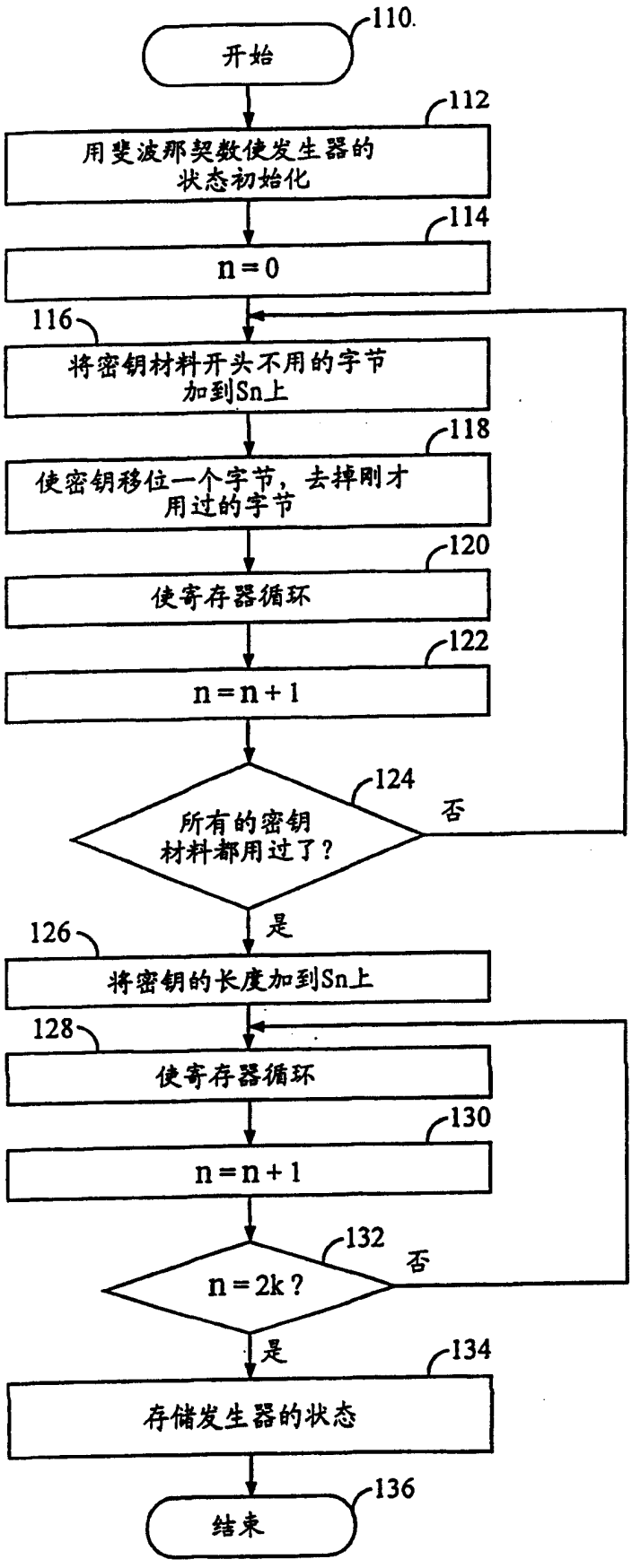

110
开始
112
用斐波那契数使发生器的状态初始化
114
n=0
116
将密钥材料开头不用的字节加到Sn上
118
使密钥移位一个字节，去掉刚才用过的字节
120
使寄存器循环
122
n=n+1
124
所有的密钥材料都用过了？
否
是
126
将密钥的长度加到Sn上
128
使寄存器循环
130
n=n+1
132
n=2k？
否
是
134
存储发生器的状态
136
结束

图 5

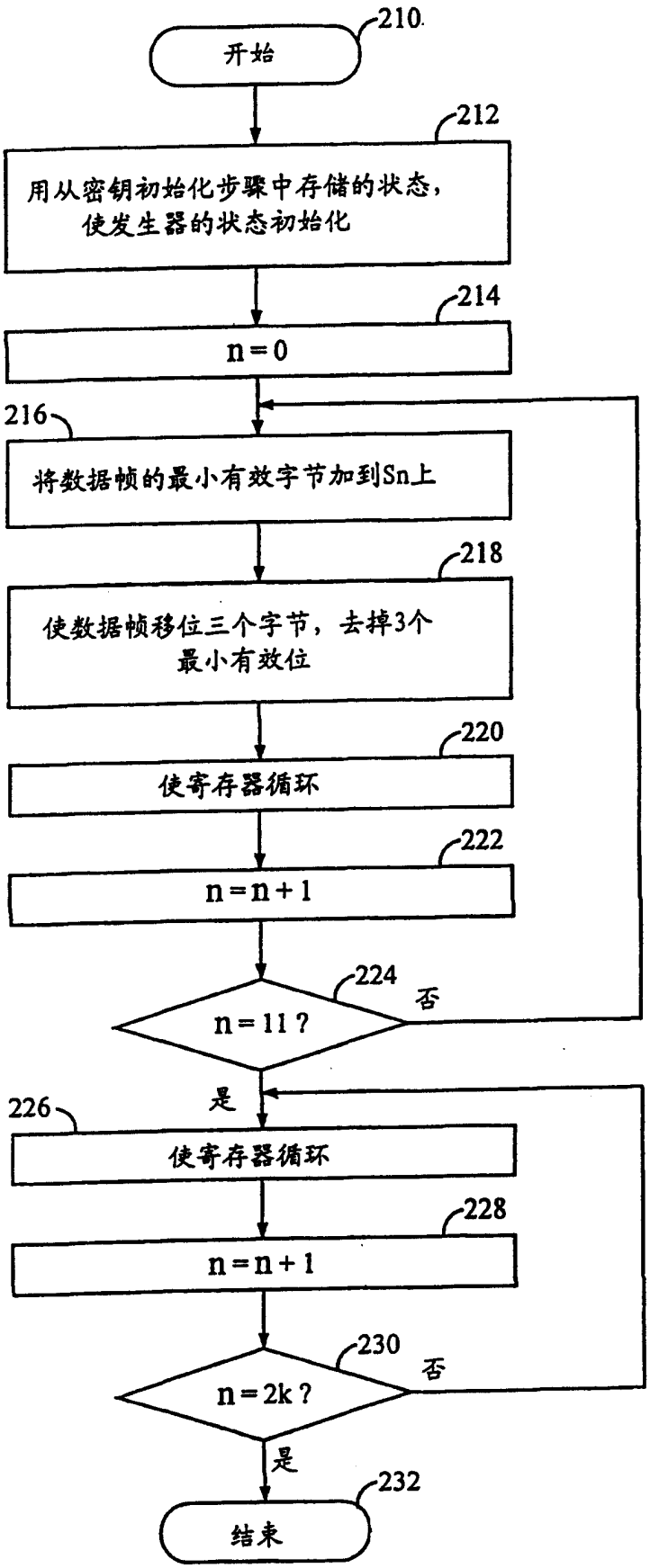

210
开始
212
用从密钥初始化步骤中存储的状态，使发生器的状态初始化
214
n=0
216
将数据帧的最小有效字节加到Sn上
218
使数据帧移位三个字节，去掉3个最小有效位
220
使寄存器循环
222
n=n+1
224
n=11？
否
是
226
使寄存器循环
228
n=n+1
230
n=2k？
否
是
232
结束

图 6A

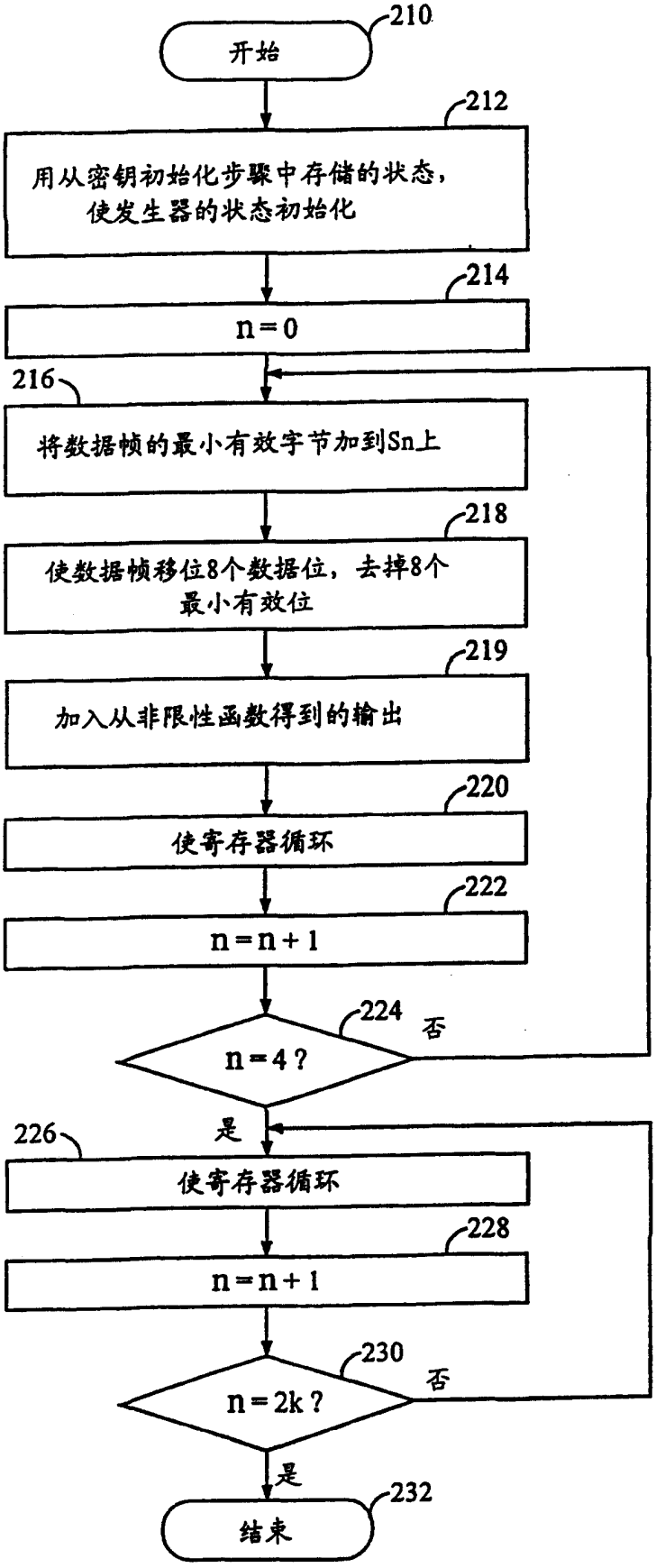

210
开始
212
用从密钥初始化步骤中存储的状态，使发生器的状态初始化
214
n=0
216
将数据帧的最小有效字节加到Sn上
218
使数据帧移位8个数据位，去掉8个最小有效位
219
加入从非限性函数得到的输出
220
使寄存器循环
222
n=n+1
224
n=4？
否
是
226
使寄存器循环
228
n=n+1
230
n=2k？
否
是
232
结束

图 6B

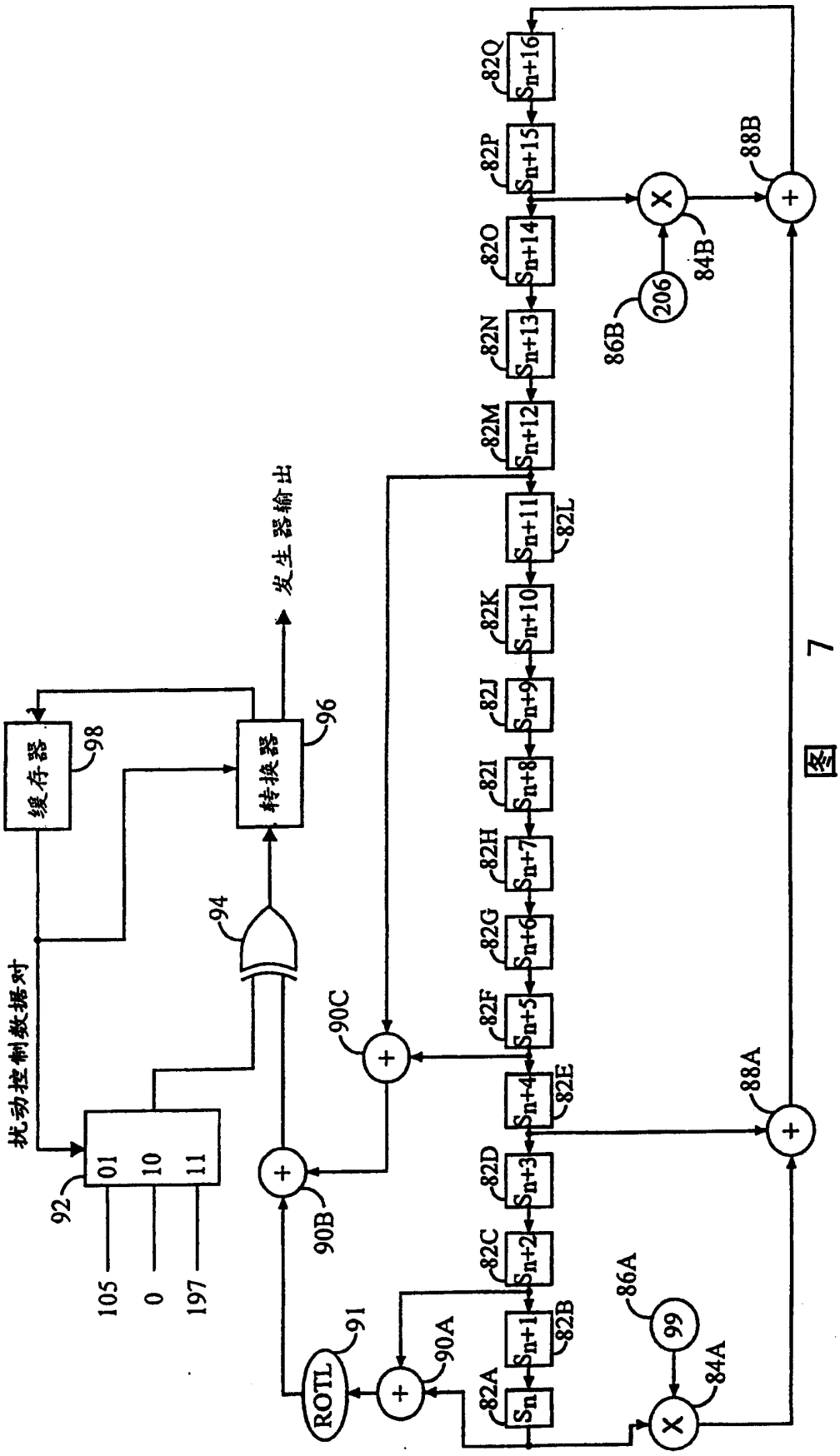

扰动控制数据对
发生器输出
缓存器
98
转换器
96
94
92
01
10
11
105
0
197
91
ROTL
90A
90B
90C
86A
99
84A
86B
206
84B
88A
88B
82A
82B
82C
82D
82E
82F
82G
82H
82I
82J
82K
82L
82M
82N
82O
82P
82Q
Sn
Sn+1
Sn+2
Sn+3
Sn+4
Sn+5
Sn+6
Sn+7
Sn+8
Sn+9
Sn+10
Sn+11
Sn+12
Sn+13
Sn+14
Sn+15
Sn+16

图 7

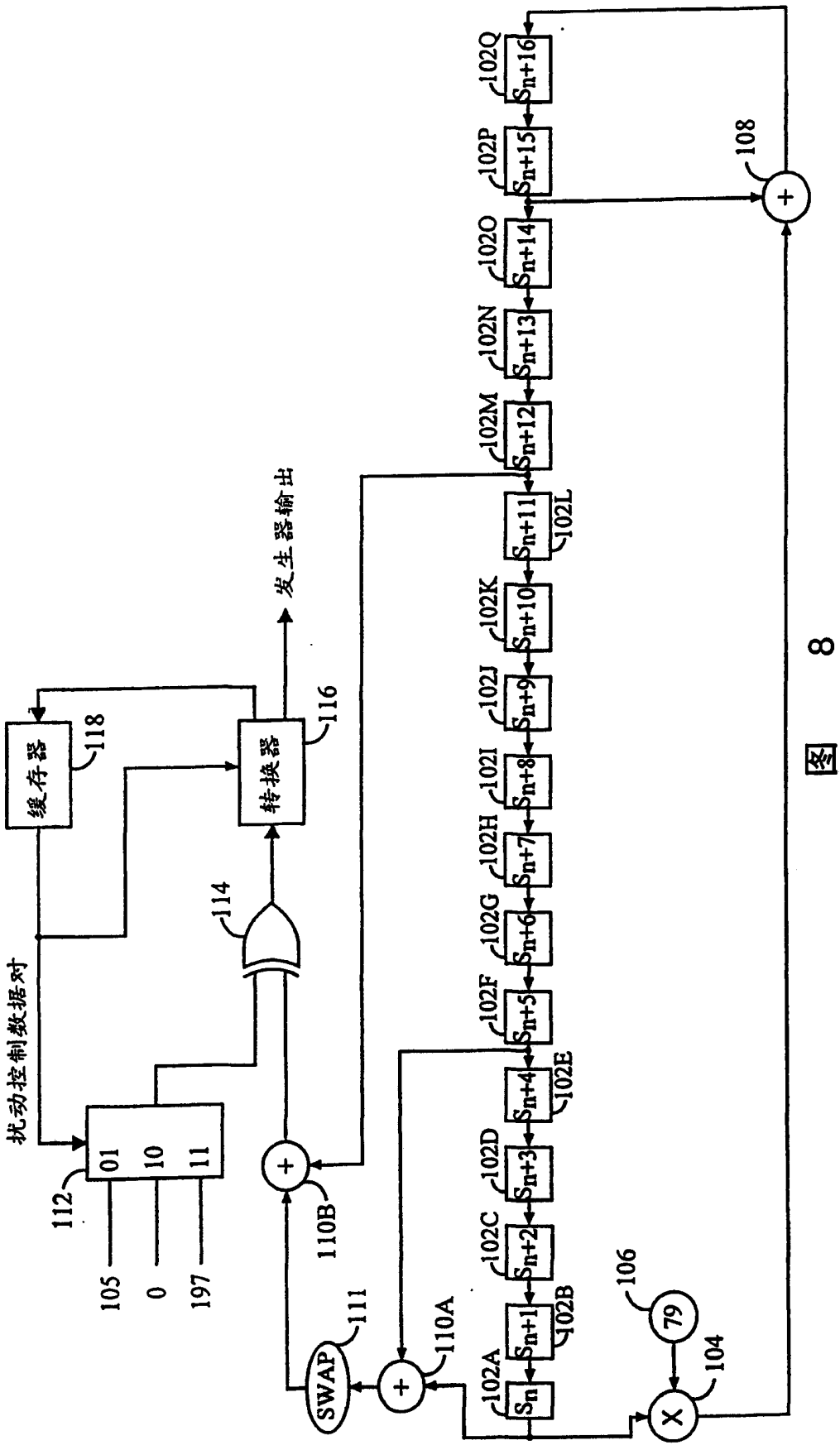

扰动控制数据对
缓存器
118
转换器
116
发生器输出
114
112
01
10
11
105
0
197
110B
SWAP
111
110A
102A
Sn
102B
Sn+1
102C
Sn+2
102D
Sn+3
102E
Sn+4
102F
Sn+5
102G
Sn+6
102H
Sn+7
102I
Sn+8
102J
Sn+9
102K
Sn+10
102L
Sn+11
102M
Sn+12
102N
Sn+13
102O
Sn+14
102P
Sn+15
102Q
Sn+16
106
79
104
108

图 8

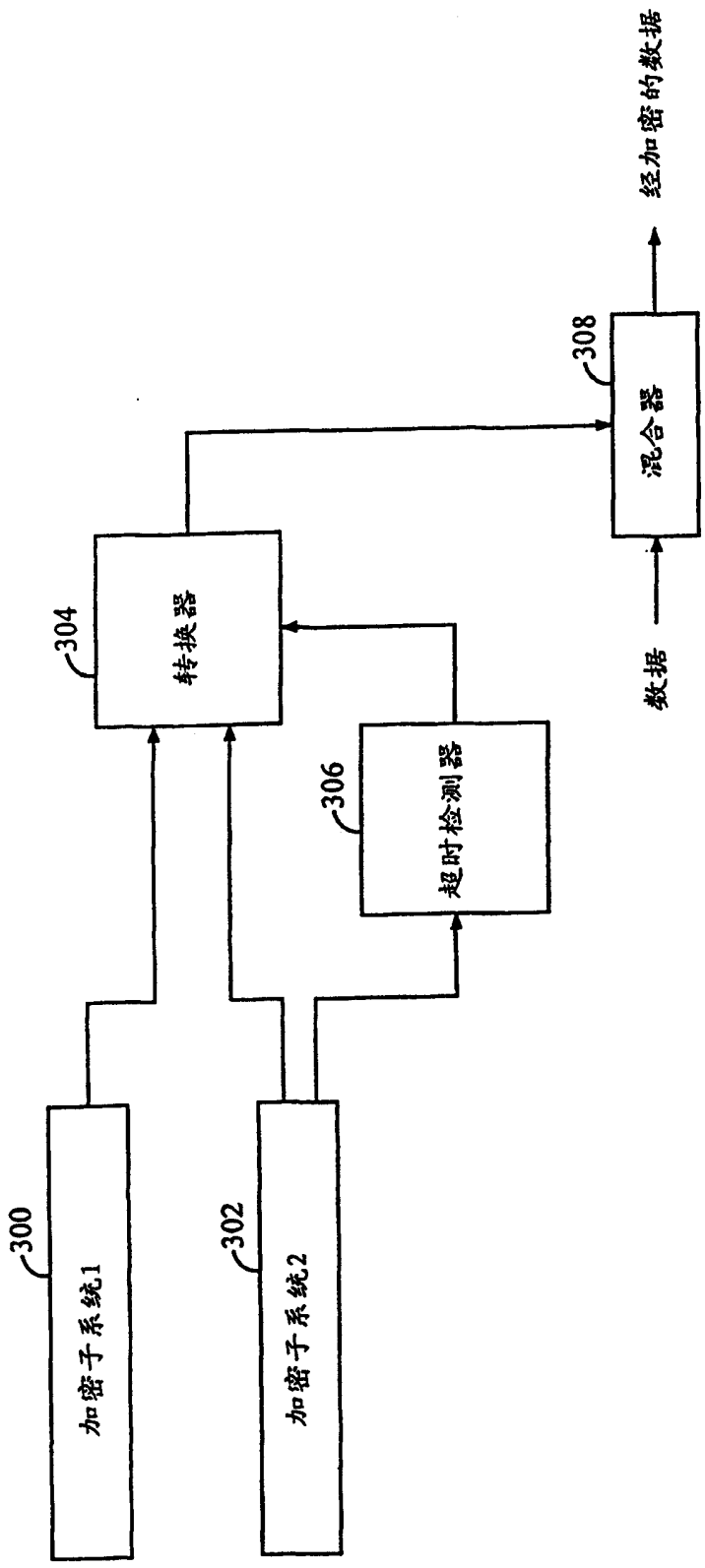


图 9

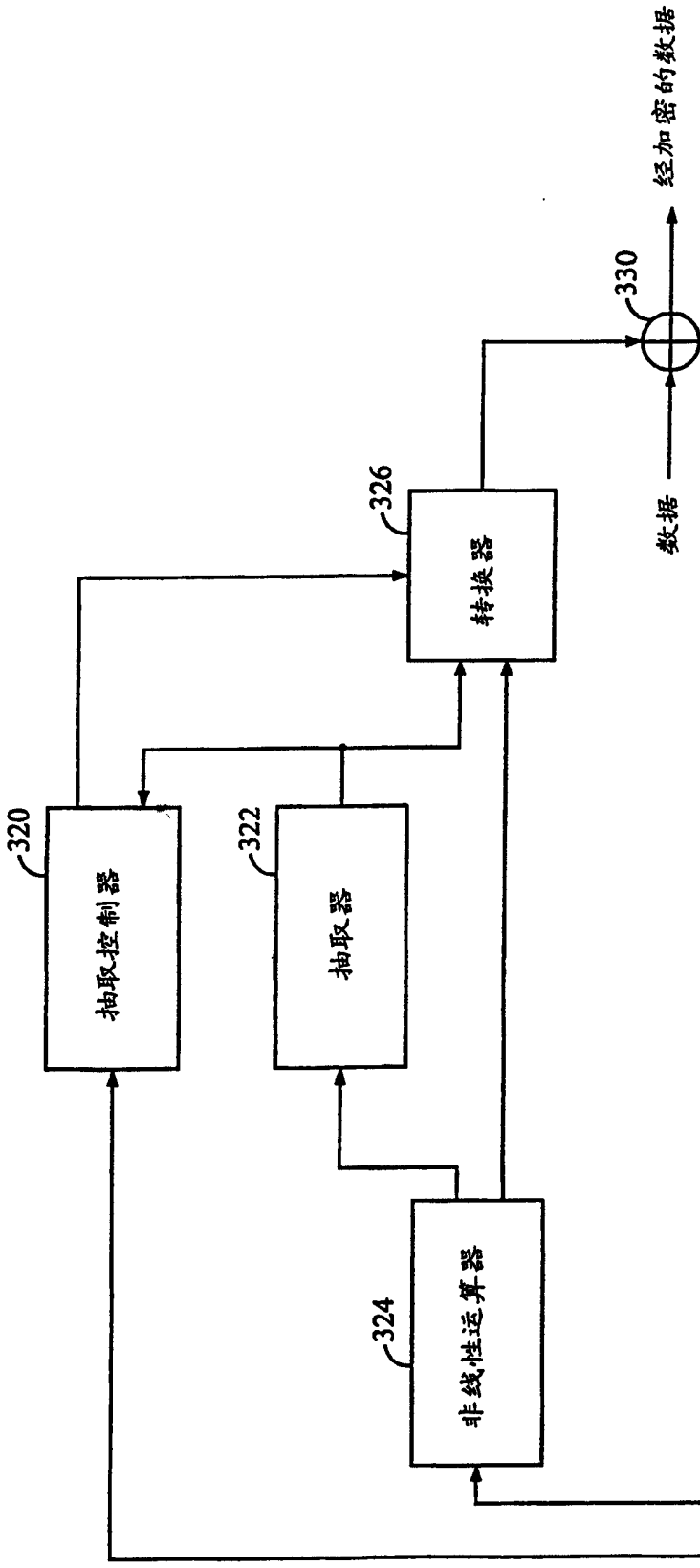


图 10

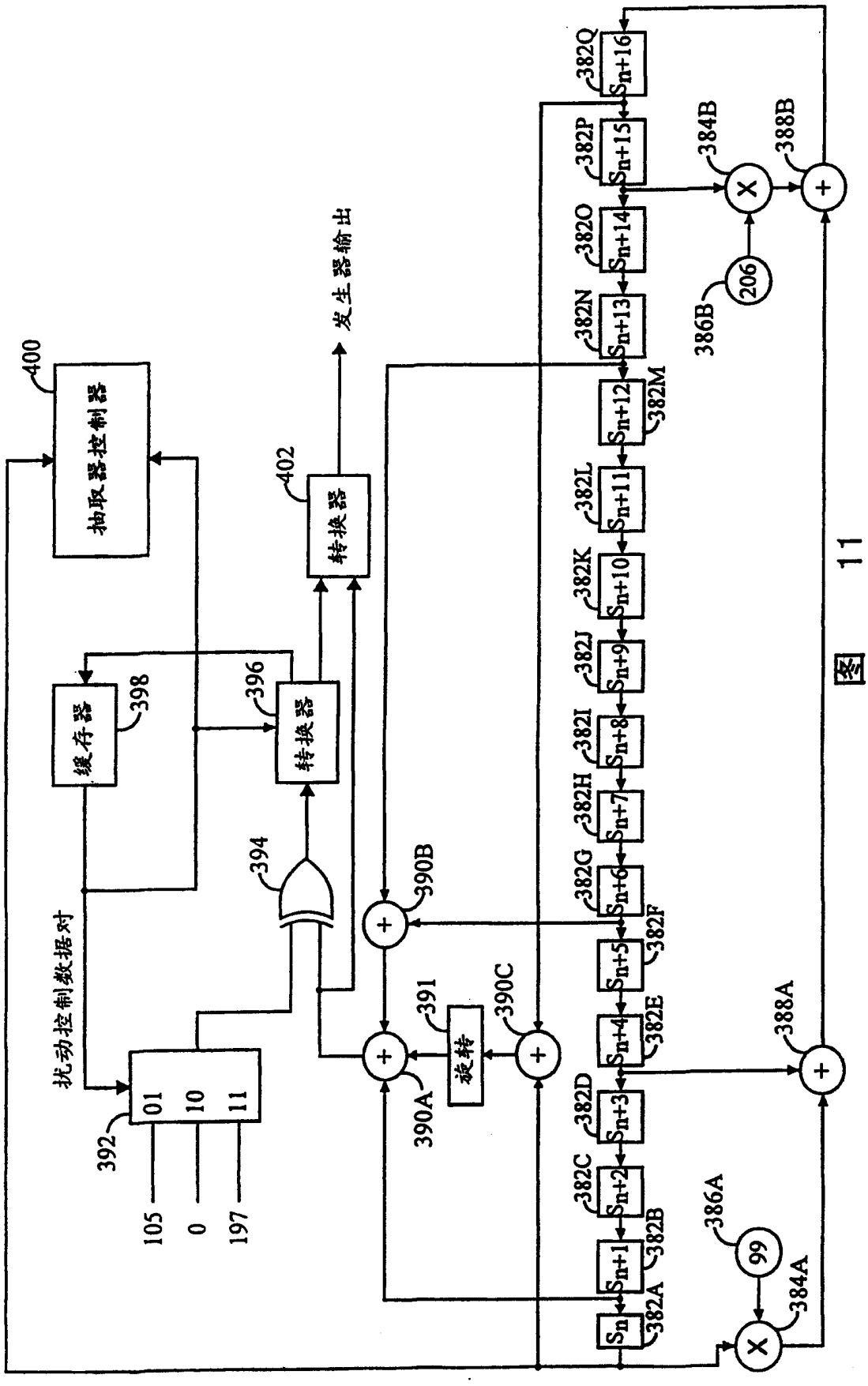

400
抽取器控制器
发生器输出
402
转换器
缓存器
398
396
转换器
394
扰动控制数据对
392
01
10
11
105
0
197
390B
390A
391
旋转
390C
382Q
Sn+16
382P
Sn+15
382O
Sn+14
382N
Sn+13
Sn+12
382M
382L
Sn+11
382K
Sn+10
382J
Sn+9
382I
Sn+8
382H
Sn+7
382G
Sn+6
Sn+5
382F
Sn+4
382E
382D
Sn+3
382C
Sn+2
Sn+1
382B
Sn
382A
384B
388B
206
386B
386A
99
384A
388A

图 11