



(12) 发明专利

(10) 授权公告号 CN 101685387 B

(45) 授权公告日 2015. 04. 29

(21) 申请号 200910174654. 5

(22) 申请日 2009. 09. 21

(30) 优先权数据

08305581. 4 2008. 09. 22 EP

08291125. 6 2008. 11. 28 EP

(73) 专利权人 汤姆森许可贸易公司

地址 法国布洛涅 - 比郎库尔

(72) 发明人 马克·乔伊

(74) 专利代理机构 中科专利商标代理有限责任

公司 11021

代理人 王波波

(51) Int. Cl.

G06F 7/72(2006. 01)

(56) 对比文件

CN 1750459 A, 2006. 03. 22,

US 2006282491 A1, 2006. 12. 14,

审查员 陈安安

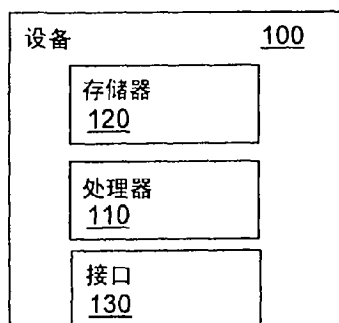
权利要求书1页 说明书8页 附图1页

(54) 发明名称

支持正整数的正则重新编码的方法、装置及
计算机程序

(57) 摘要

本发明提供一种用于对正整数 n 进行重新编码的正则方法, 其中, 选择小于 n 的整数 s 以定义整数 $n' = n - s$, 将 n' 的 m 进制表示逐数位与 s 的 m 进制表示相加, 以生成 n 的重新编码的表示, 其中 m 为整数。本发明还提供了一种设备 (100) 以及计算机程序产品 (140)。本方法的优点是它是正则的。



1. 一种用于对第一正整数 n 进行重新编码的正则方法, 所述第一正整数 n 是密码求幂算法的指数, 所述方法包括在处理器 (110) 中的以下步骤:

选择小于 n 的第二正整数 s ;

定义第三整数 $n' = n - s$;

对于第四整数 m , 将第三整数 n' 的 m 进制表示逐数位地与 s 的 m 进制表示相加, 以生成 n 的重新编码的表示。

2. 根据权利要求 1 所述的方法, 其中 $m = 2^k$, 其中 k 是正整数。

3. 根据权利要求 1 或 2 所述的方法, 其中 $s = \sum_{i=0}^{l-2} s_i m^i$, 其中 l 代表 n 的 m 进制长度。

4. 根据权利要求 3 所述的方法, 其中, 对于某个 $0 < \alpha < m$, $s_i = \alpha$ 。

5. 根据权利要求 4 所述的方法, 其中 $\alpha = 1$ 。

6. 根据权利要求 4 所述的方法, 其中 $\alpha = m - 1$ 。

7. 一种用于对第一正整数 n 进行正则重新编码的设备 (100), 所述第一正整数 n 是密码求幂算法的指数, 所述设备包括处理器 (110), 所述处理器 (110) 适于:

选择小于 n 的第二正整数 s ;

定义第三整数 $n' = n - s$;

对于第四整数 m , 将第三整数 n' 的 m 进制表示逐数位地与 s 的 m 进制表示相加, 以生成 n 的重新编码的表示。

支持正整数的正则重新编码的方法、装置及计算机程序

技术领域

[0001] 本发明一般地涉及数位重新编码,并且更具体地,涉及无符号数的重新编码。

背景技术

[0002] 本节意在向读者介绍可能与以下描述和 / 或要求保护的本发明的各方面相关的领域的各个方面。相信这种讨论有助于向读者提供背景信息,以便于更好地理解本发明的各个方面。相应地,应注意,应根据这一点来阅读这些陈述,而不应将其接纳为现有技术。

[0003] 已经表明,密码求幂算法易受旁道攻击(side channel attacks)的攻击。在“Differential Power Analysis”(M. J. Wiener, editor, Advances in Cryptology-CRYPTO' 99, volume 1666 of Lecture Notes in Computer Science, pages 388-397, Springer Verlag 1999) 中,Paul Kocher、Joshua Jaffe 以及 Benjamin Jun 描述了使用对功率消耗的观察来进行的攻击,而 Karine Gandolffi、Christophe Mourtel 和 Francis Olivier 在“Electromagnetic Analysis :Concrete Results”(Ç.K. Koç, D. Naccache and C. Paar, editors, Cryptographic Hardware and Embedded Systems-CHES 2001, volume 2162 of Lecture Notes in Computer Science, pages 251-261, Springer Verlag 2001) 中以及 Jean-Jacques Quisquater 和 David Samyde 在“Electromagnetic Analysis (EMA) : Measures and Counter-Measures for Smart Cards”(I. Attali and T. P. Jensen, editors, Smart Card Programming and Security (E-Smart 2001), volume 2140 of Lecture Notes in Computer Science, pages 200-210, Springer Verlag 2001) 中已经描述了使用对电磁发射的观察来进行的攻击。

[0004] 被称作简单功率分析 (SPA) 以及简单电磁分析 (SEMA) 的这些攻击可以揭示求幂算法中简单实现的指数,这是由于所需运算依赖于指数的逐比特表示。

[0005] 为了减少计算求幂所要求的运算数量,已经开发了重新编码算法。最公知的示例是由 Ian Blake、Gadiel Seroussi 和 Nigel Smart 在“Elliptic Curves in Cryptography”(volume 265 of London Mathematical Society Lecture Note Series. Cambridge University Press. 1999) 中描述的非相邻形式 (Non-Adjacent Form) (NAF) 重新编码。NAF 重新编码使用 $\{-1, 0, 1\}$ 中的值来对指数的比特进行重新编码。这减少了后续求幂算法中要求的乘法数量,可以将其推广至 m 进制重新编码,如 Donald E. Knuth 在 The Art of Computer Programming (volume 2/Seminumerical Algorithms. Addison-Wesley, 2nd edition, 1981) 中所描述的。然而,设计这些重新编码算法是为了提高求幂算法的效率,而非增强对旁道攻击的抵御。

[0006] 已经提出了几种其它重新编码算法:

[0007] -Bodo Möller, “Parallelizable Elliptic Curve Point Multiplication Method with Resistance against Side-Channel Attacks”. In A. H. Chan and V. Gligor, editors, Information Security (ISC 2002), volume 2433 of Lecture Notes in Computer Science, pages 402-413, Springer Verlag 2002.

[0008] -Bodo Möller. “Fractional Windows Revisited: Improved Signed-Digit Representation for Efficient Exponentiation”. In C. Park and S. Chee, editors, Information Security and Cryptology-ICISC 2004, volume 3506 of Lecture Notes in Computer Science, pages 137-153, Springer Verlag 2004.

[0009] -Katsuyuki Okeya and Tsuyoshi Takagi. “A More Flexible Countermeasure against Side-Channel Attacks Using Window Method”. In C. D. Walter, Ç.K. Koç and C. Paar, editors, Cryptographic Hardware and Embedded Systems-CHES 2003, volume 2779 of Lecture Notes in Computer Science, pages 397-410, Springer Verlag 2003.

[0010] -Katsuyuki Okeya and Tsuyoshi Takagi. “The Width-w NAF method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side-Channel Attacks”. In M. Joye, editor, Topics in Cryptology-CT-RSA 2003, volume 2612 of Lecture Notes in Computer Science, pages 328-342, Springer Verlag 2003.

[0011] 然而, 如 Yasuyuki Sakai 和 Kouichi Sakurai 在 “A New Attack with Side Channel Leakage During Exponent Recoding Computations” (M. Joye and J.-J. Quisquater, editors, Cryptographic Hardware and Embedded Systems-CHES 2004, volume 3156 of Lecture Notes in Computer Science, pages 298-311, Springer Verlag 2004) 中所指出的, 为了实现正则求幂算法, 所使用的任何重新编码算法也必须是正则的。在正则重新编码算法中, 在算法的计算期间, 在主循环中不存在测试。

[0012] 尽管可以争论认为在生成指数时可以执行重新编码, 但在例如指数与随机值相结合的情况下这是不可能的, 这是由于必须在求幂之前执行重新编码。与随机值相结合是为了阻止特定旁道分析, 如 Jean-Sébastien Coron 在 “Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems” (Ç.K. Koç and C. Paar, editors, Cryptographic Hardware and Embedded Systems-CHES '99, volume 1717 of Lecture Notes in Computer Science, pages 292-302, Springer Verlag 1999) 中以及 Paul Kocher 在 “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems” (N. Koblitz, editor, Advances in Cryptology-CRYPTO' 96, volume 1109 of Lecture Notes in Computer Science, pages 104-113, Springer Verlag 1996) 中所述。

[0013] 为了使求幂正则化, 已经提出了其它重新编码算法。Bodo Möller 在 “Securing Elliptic Curve Point Multiplication against Side-Channel Attacks” (G. Davida and Y. Frankel, editors, Information Security (ISC2001), volume 2200 of Lecture Notes in Computer Science, pages 324-334, Springer Verlag 2001) 中描述了一种针对 m 进制求幂的重新编码算法。用 $-m$ 来代替每个等于零的数位, 并且将次最高有效数位加 1。这得到了用集合 $\{1, \dots, m-1\} \cup \{-m\}$ 中包括的数来重新编码的指数。与 m 进制求幂算法相结合, 这表明应当对 x^m 进行预计算。尽管该计算在椭圆曲线上是“容易”的, 但是在有限环的乘法群的情况中则不是这样。

[0014] Camille Vuillaume 和 Katsuyuki Okeya 在 “Flexible Exponentiation With Resistance to Side Channel Attacks” (J. Zhou, M. Yung and F. Bao, editors, Applied Cryptography and Network Security-ACNS 2006, volume 3989 of Lecture Notes in Computer Science, pages 268-283, Springer Verlag 2006) 中描述了 Möller 的算法的无

符号版本。在集合 $\{1, \dots, m\}$ 中对数位进行重新编码：用 m 来代替每一个零数位并且将下一数位减 1。

[0015] Möller 算法的有符号和无符号版本的缺点是不能以正则方式来容易地实现它们。

[0016] 因此可以认识到，需要针对正则求幂的重新编码算法，其中以正则方式对指数进行简单的重新编码。本发明提供了针对该方案的多种变型。

发明内容

[0017] 在第一方面，本发明提供了一种用于对第一正整数 n 进行重新编码的正则方法，所述第一正整数 n 是密码求幂算法的指数。处理器选择小于 n 的第二整数 s ，定义第三整数 $n' = n - s$ ，以及对于第四整数 m ，将第三整数 n' 的 m 进制表示逐数位地与 s 的 m 进制表示相加，以生成 n 的重新编码的表示。

[0018] 在第一优选实施例中， $m = 2^k$ 。

[0019] 在第二优选实施例中， $s = \sum_{i=0}^{l-2} s_i m^i$ ，其中 l 代表 n 的 m 进制长度。有利地，对于某个 $0 < \alpha < m$ ， $s_i = \alpha$ ；其中，优选地， $\alpha = 1$ 或者 $\alpha = m-1$ 。

[0020] 在第二方面，本发明提供了一种用于对第一正整数 n 进行正则重新编码的设备。所述设备包括处理器，所述处理器适于：选择小于 n 的第二整数 s ，定义第三整数 $n' = n - s$ ，以及对于第四整数 m ，将第三整数 n' 的 m 进制表示逐数位地与 s 的 m 进制表示相加，以生成 n 的重新编码的表示。

[0021] 在第三方面，本发明了提供一种存储指令的计算机程序产品，当由处理器执行时，所述指令执行本发明的第一方面的方法。

附图说明

[0022] 现在将通过非限制性示例并参考附图来描述本发明的优选特征，附图中：

[0023] 图 1 示出了根据本发明的优选实施例的一种用于进行数位重新编码的设备。

[0024] 图中所示的方框是功能性实体，不必须与物理上分离的实体相对应。可以将这些功能实体实现为硬件、软件或者软件和硬件的组合；此外，可以在一个或者更多集成电路中实现这些功能实体。

具体实施方式

[0025] 图 1 示出了用于对数位，尤其是求幂算法中要使用的指数的数位进行重新编码的设备 100。设备 100 包括至少一个适于执行计算机程序的处理器 110（下文中称作“处理器”），该计算机程序执行下文描述的任意实施例的重新编码算法的计算。应当注意，还可以用硬件或者软件和硬件的组合来实现处理器 110。设备 100 还包括适于存储数据（例如来自处理器 110 的中间计算结果）的存储器 120。设备 100 还包括用于与其他设备（图中未示出）进行交互的至少一个接口 130（下文中称作“接口”）。图 1 还示出了存储计算机程序的计算机程序产品 140，例如 CD-ROM，当处理器 110 执行该计算机程序时，该计算机程序执行根据本发明的两个实施例中任一个的重新编码算法。

[0026] 在求幂中, 计算 $z = x^n$, 其中 n 为整数并且 x 是群 (以乘法形式来写的群) 中的元素。令 $n = \sum_{i=0}^{l-1} d_i m^i$ 代表以 m 为基数的 n 的展开式 (典型地 $m = 2^k$), 其中 l 是 n 的 m 进制长度。

取正整数 $s < n$ 并定义 $n' := n - s$ 。如果 $n' = \sum_{i=0}^{l-1} d'_i m^i$ 和 $s = \sum_{i=0}^{l-1} s_i m^i$ 分别代表 n' 和 s 的 m 展开式, 则 $x^n = x^{n'+s}$ 成立, 其中 $n' + s = \sum_{i=0}^{l-1} k_i m^i$, 其中继而 $k_i = d'_i + s_i$ 。

[0027] 如果我们定义以 m 为基数的 s 的最高有效数位为零, 则以 m 为基数的 n' 的最高有效数位 (即 k_{l-1}) 将保持大于或者等于零。如果不是这样, 则该重新编码将不是无符号的并且将因此不适用于计算逆 (inversion) 代价较高的群。

[0028] 第一优选实施例

[0029] 令 α 是满足 $0 < \alpha < m$ 的整数。

[0030] 选择 $s = \sum_{i=0}^{l-2} \alpha m^i = \alpha \frac{m^{l-1} - 1}{m - 1}$ 。这可以被视为是将 s 的所有数位设置为相同的值,

即 α 。由于 $n'_i \in \{0, \dots, m-1\}$, 则 $k_i \in \{\alpha, \dots, \alpha + (m-1)\}$ 。然后使用下述算法用于重新编码。

[0031] 输入: $n \geq 1, m = 2^k, l$ (n 的 m 进制长度)。

[0032] 输出: $n = \{k_{l-1}, \dots, k_0\}_m$, 其中 $k_i \in \{\alpha, \dots, \alpha + (m-1)\}, 0 \leq i \leq l-2$

[0033] 算法:

[0034] $s \leftarrow \alpha \frac{m^{l-1} - 1}{m - 1}$ 以及 $n \leftarrow n - s$

[0035] for $i = 0$ to $l-2$ do

[0036] $d \leftarrow n \bmod m$

[0037]

$$n \leftarrow \lfloor n/m \rfloor$$

[0038] $k_i \leftarrow d + \alpha$

[0039] end

[0040] $k_{l-1} \leftarrow n$

[0041] α 的第一优选选择是 1, 这是由于它针对重新编码的数位产生较小的值。 α 的第二优选选择是 $m-1$, 这是由于它给出 $s = m^{l-1} - 1$ (即设置为 1 的 $k(l-1)$ 的延续)。

[0042] 现在, 两个示例将说明第一优选实施例。对于这两个示例, 参数采用下列值:

[0043] $k = 2$

[0044] $m = 4$

[0045] $n = 73 = (1, 0, 2, 1)_4 = 1 \cdot 4^0 + 2 \cdot 4^1 + 0 \cdot 4^2 + 1 \cdot 4^3$

[0046] $l = 4$

[0047] 在第一示例中 $\alpha = 1$; 在第二示例中 $\alpha = m-1 = 3$ 。

[0048] 第一实施例的第一示例 ($\alpha = 1$)

[0049] $s := \alpha \frac{m^{l-1} - 1}{m - 1} = 1 \frac{4^{4-1} - 1}{4 - 1} = \frac{4^3 - 1}{3} = \frac{63}{3} = 21$

[0050] $n := n - s = 73 - 21 = 52$

[0051] 循环:for i = 0 to 1-2,即 for i = 0 to 2

[0052] i = 0 :

[0053] d := n mod m = 52 mod 4 = 0

[0054]

$$n := \lfloor n/m \rfloor = \lfloor 52/4 \rfloor = 13$$

[0055] k₀ := d + α = 0 + 1 = 1

[0056] i = 1 :

[0057] d := n mod m = 13 mod 4 = 1

[0058]

$$n := \lfloor n/m \rfloor = \lfloor 13/4 \rfloor = 3$$

[0059] k₁ := d + α = 1 + 1 = 2

[0060] i = 2 :

[0061] d := n mod m = 3 mod 4 = 3

[0062]

$$n := \lfloor n/m \rfloor = \lfloor 3/4 \rfloor = 0$$

[0063] k₂ := d + α = 3 + 1 = 4

[0064] k₃ := n = 0

[0065] k = (k₃, k₂, k₁, k₀) = (0, 4, 2, 1)

[0066] $n = \sum_{i=0}^{l-1} k_i m^i = 1 \cdot 4^0 + 2 \cdot 4^1 + 4 \cdot 4^2 + 0 \cdot 4^3 = 1 \cdot 1 + 2 \cdot 4 + 4 \cdot 16 = 1 + 8 + 64 = 73$

[0067] 如所预期的,重新编码的 n 等于原始的 n。

[0068] 第一实施例的第二示例 (α = m-1 = 3)

[0069] $s := \alpha \frac{m^{l-1} - 1}{m - 1} = m^{l-1} - 1 = 4^3 - 1 = 63$

[0070] n := n - s = 73 - 63 = 10

[0071] 循环:for i = 0 to 1-2,即 for i = 0 to 2

[0072] i = 0 :

[0073] d := n mod m = 10 mod 4 = 2

[0074]

$$n := \lfloor n/m \rfloor = \lfloor 10/4 \rfloor = 2$$

[0075] k₀ := d + α = 2 + 3 = 5

[0076] i = 1 :

[0077] d := n mod m = 2 mod 4 = 2

[0078]

$$n := \lfloor n/m \rfloor = \lfloor 2/4 \rfloor = 0$$

[0079] k₁ := d + α = 2 + 3 = 5

[0080] i = 2 :

[0081] d := n mod m = 0 mod 4 = 0

[0082]

$$n := \lfloor n/m \rfloor = \lfloor 0/4 \rfloor = 0$$

[0083] $k_2 := d + \alpha = 0 + 3 = 3$

[0084] $k_3 := n = 0$

[0085] $k = (k_3, k_2, k_1, k_0) = (0, 3, 5, 5)$

[0086]

$$n = \sum_{i=0}^{l-1} k_i m^i = 5 \cdot 4^0 + 5 \cdot 4^1 + 3 \cdot 4^2 + 0 \cdot 4^3 = 5 \cdot 1 + 5 \cdot 4 + 3 \cdot 16 = 5 + 20 + 48 = 73$$

[0087] 如所预期的,重新编码的 n 再一次等于原始的 n 。

[0088] 应当注意,根据第一实施例的算法易于实现,但是要求预先知道 n 的 m 进制长度 (即 l)。由于这可能是一个缺点,因此第二优选实施例克服了这个问题,同时其实现稍微复杂一些。

[0089] 第二优选实施例

[0090] 如果更细致地查看减法步骤, $n' := n - s$, 可以建立下列等式 $d'_{i-1} = (d_i - s_i + \gamma_i)$

$\text{mod } m$ 以及 $\gamma_{i+1} = \left\lfloor \frac{d_i - s_i + \gamma_i}{m} \right\rfloor \in \{-1, 0\}$, 其中将“借位 (borrow)”初始化为 0, 即 $\gamma_0 = 0$ 。

这是在学校中学到的经典减法算法。由于 $d_i, s_i \in \{0, \dots, m-1\}$, 这给出 $k_i = d'_{i-1} + s_i$, 如果 $d_i + \gamma_i \geq s_i$, 则 $k_i = d'_{i-1} + s_i$ 等于 $d_i + \gamma_i$, 否则等于 $d_i + \gamma_i + m$ 。

[0091] 因此,对于 $s_i \neq 0$ 的任意选择,当 $d_i \in \{0, 1\}$ 时,得到 k_i 的非零值。如第一优选实施例中一样,对于某个 $0 < \alpha < m$, $s = \sum_{i=0}^{l-2} \alpha m^i$ 。此外,为了仅使用无符号算术, $\gamma'_{i-1} = \gamma_{i-1} + 1 \in \{0, 1\}$:

[0092]

$$\gamma'_i = \gamma_i + 1 = \left\lfloor \frac{d_i - s_i + \gamma_i}{m} \right\rfloor + 1 = \left\lfloor \frac{d_i - s_i + \gamma_i + m}{m} \right\rfloor = \left\lfloor \frac{d_i - s_i + \gamma'_{i-1} + m}{m} \right\rfloor$$

[0093] 输入 : $n \geq 1, m = 2^k, 0 < \alpha < m$

[0094] 输出 : $n = (k_{i-1}, \dots, k_0)_m$ 其中 $k_i \in \{\alpha, \dots, \alpha + (m-1)\}, 0 \leq i \leq l-2$

[0095] 算法 :

[0096] $i \leftarrow 0; \gamma' \leftarrow 1$

[0097] while $n \geq (m + \alpha)$ do

[0098] $d \leftarrow n \text{ mod } m$

[0099] $d' \leftarrow d + \gamma' + m - \alpha - 1$

[0100] $k_i \leftarrow (d' \text{ mod } m) + \alpha$

[0101]

$$\gamma' \leftarrow \lfloor d'/m \rfloor$$

[0102]

$$n \leftarrow \lfloor n/m \rfloor$$

[0103] $i \leftarrow i + 1$

[0104] end

[0105] $k_l \leftarrow n + \gamma' - 1$

[0106] 如第一优选实施例中一样, α 的优选选择是 1 和 $m-1$ 。

[0107] 现在,两个示例将说明第二优选实施例。对于这两个示例,参数采用下列值:

[0108] $k = 2$

[0109] $m = 4$

[0110] $n = 73 = (1, 0, 2, 1)_4 = 1 \cdot 4^0 + 2 \cdot 4^1 + 0 \cdot 4^2 + 1 \cdot 4^3$

[0111] 在第一示例中 $\alpha = 1$; 在第二示例中 $\alpha = m-1 = 3$ 。

[0112] 第二实施例的第一示例 ($\alpha = 1$)

[0113] $i := 0$

[0114] $\gamma' = 1$

[0115] $n = 73 \geq (m + \alpha) = 4 + 1 = 5$, 所以执行 while 循环

[0116] $d := n \bmod m = 73 \bmod 4 = 1$

[0117] $d' := d + \gamma' + m - \alpha - 1 = 1 + 1 + 4 - 1 - 1 = 4$

[0118] $k_0 := (d' \bmod m) + \alpha = (4 \bmod 4) + 1 = 0 + 1 = 1$

[0119]

$$\gamma' := \lfloor d'/m \rfloor = \lfloor 4/4 \rfloor = 1$$

[0120]

$$n := \lfloor n/m \rfloor = \lfloor 73/4 \rfloor = 18$$

[0121] $i := i + 1 = 0 + 1 = 1$

[0122] $n = 18 \geq (m + \alpha) = 4 + 1 = 5$, 所以再一次执行 while 循环

[0123] $d := n \bmod m = 18 \bmod 4 = 2$

[0124] $d' := d + \gamma' + m - \alpha - 1 = 2 + 1 + 4 - 1 = 5$

[0125] $k_1 := (d' \bmod m) + \alpha = (5 \bmod 4) + 1 = 1 + 1 = 2$

[0126]

$$\gamma' := \lfloor d'/m \rfloor = \lfloor 5/4 \rfloor = 1$$

[0127]

$$n := \lfloor n/m \rfloor = \lfloor 18/4 \rfloor = 4$$

[0128] $i := i + 1 = 1 + 1 = 2$

[0129] $n = 4 < (m + \alpha) = 4 + 1 = 5$, 所以不再一次执行 while 循环

[0130] $k_2 := n + \gamma' - 1 = 4 + 1 - 1 = 4$

[0131] $k = (k_3, k_2, k_1, k_0) = (0, 4, 2, 1)$

[0132] $n = \sum_{i=0}^{l-1} k_i m^i = 1 \cdot 4^0 + 2 \cdot 4^1 + 4 \cdot 4^2 + 0 \cdot 4^3 = 1 \cdot 1 + 2 \cdot 4 + 4 \cdot 16 = 1 + 8 + 64 = 73$

[0133] 如所预期的,重新编码的 n 等于原始的 n 。

[0134] 第二实施例的第二示例 ($\alpha = m-1 = 3$)

[0135] $i := 0$

[0136] $\gamma' = 1$

[0137] $n = 73 \geq (m + \alpha) = 4 + 3 = 7$, 所以执行 while 循环

[0138] $d := n \bmod m = 73 \bmod 4 = 1$

[0139] $d' := d + \gamma' + m - \alpha - 1 = 1 + 1 + 4 - 3 - 1 = 2$

[0140] $k_0 := (d' \bmod m) + \alpha = (2 \bmod 4) + 3 = 2 + 3 = 5$

[0141]

$$\gamma' := \lfloor d'/m \rfloor = \lfloor 2/4 \rfloor = 0$$

[0142]

$$n := \lfloor n/m \rfloor = \lfloor 73/4 \rfloor = 18$$

[0143] $i := i+1 = 0+1 = 1$ [0144] $n = 18 \geq (m+\alpha) = 4+1 = 5$, 所以再一次执行 while 循环[0145] $d := n \bmod m = 18 \bmod 4 = 2$ [0146] $d' := d + \gamma' + m - \alpha - 1 = 2 + 0 + 4 - 3 - 1 = 2$ [0147] $k_1 := (d' \bmod m) + \alpha = (2 \bmod 4) + 3 = 2 + 3 = 5$

[0148]

$$\gamma' := \lfloor d'/m \rfloor = \lfloor 2/4 \rfloor = 0$$

[0149]

$$n := \lfloor n/m \rfloor = \lfloor 18/4 \rfloor = 4$$

[0150] $i := i+1 = 1+1 = 2$ [0151] $n = 4 < (m+\alpha) = 4+1 = 5$, 所以不再一次执行 while 循环[0152] $k_2 := n + \gamma' - 1 = 4 + 0 - 1 = 3$ [0153] $k = (k_3, k_2, k_1, k_0) = (0, 3, 5, 5)$

[0154]

$$n = \sum_{i=0}^{l-1} k_i m^i = 5 \cdot 4^0 + 5 \cdot 4^1 + 3 \cdot 4^2 + 0 \cdot 4^3 = 5 \cdot 1 + 5 \cdot 4 + 3 \cdot 16 = 5 + 20 + 48 = 73$$

[0155] 如所预期的, 重新编码的 n 等于原始的 n 。

[0156] 可以认识到, 对相同的输入, 这两个实施例如所预期的那样给出了相同的重新编码的数位。例如, 对于这两个实施例, 第一示例给出 $(4, 2, 1)$, 同时对于这两个实施例, 第二示例给出 $(3, 5, 5)$ 。

[0157] 还可以认识到, 这两个实施例是正则的, 这是由于在主循环内不存在测试; 在第一实施例中, 在 for 循环内不存在测试, 并且在第二实施例中, 在 while 循环中不存在测试。

[0158] 因此可以认识到, 本发明实现了正整数的正则重新编码。

[0159] 可以独立地或以任何合适组合来提供说明书和 (在合适时) 权利要求和附图中公开的每个特征。描述为以硬件实现的特征也可以以软件实现, 反之亦然。在适用时, 连接可以被实现为无线连接或有线 (不必须是直接或专用的) 连接。

[0160] 权利要求中出现的参考标号仅作为示意, 不对权利要求的范围具有限制作用。

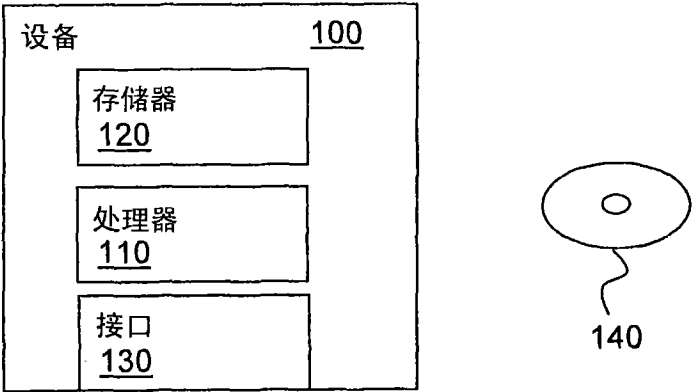


图 1