(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5436996号
(P5436996)
(45)発行日　平成26年3月5日(2014.3.5)　(24)登録日　平成25年12月20日(2013.12.20)

(51)Int.Cl.　F I
G09C 1/00 (2006.01)　G09C 1/00 650A

請求項の数 8　外国語出願　(全 14 頁)

(21)出願番号　特願2009-211965(P2009-211965)
(22)出願日　平成21年9月14日(2009.9.14)
(65)公開番号　特開2010-72644(P2010-72644A)
(43)公開日　平成22年4月2日(2010.4.2)
審査請求日　平成24年8月6日(2012.8.6)
(31)優先権主張番号　08305581.4
(32)優先日　平成20年9月22日(2008.9.22)
(33)優先権主張国　欧州特許庁(EP)
(31)優先権主張番号　08291125.6
(32)優先日　平成20年11月28日(2008.11.28)
(33)優先権主張国　欧州特許庁(EP)

(73)特許権者　501263810
トムソン　ライセンシング
Ｔｈｏｍｓｏｎ　Ｌｉｃｅｎｓｉｎｇ
フランス国，　９２１３０　イツシー　レ　ムーリノー，　ル　ジヤンヌ　ダルク，　１－５
１－５，　ｒｕｅ　Ｊｅａｎｎｅ　ｄ’Ａｒｃ，　９２１３０　ＩＳＳＹ　ＬＥＳ　ＭＯＵＬＩＮＥＡＵＸ，　Ｆｒａｎｃｅ
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介
(74)代理人　100107766
弁理士　伊東　忠重

最終頁に続く

(54)【発明の名称】正の整数の正則なリコーディングの方法、装置及びコンピュータプログラム

(57)【特許請求の範囲】
【請求項１】
暗号指数アルゴリズムの指数である第一の正の整数ｎを再コード化する正則な方法であって、
当該方法は、プロセッサにおいて、
減算手段が、インターフェースよって受け取られる前記第一の正の整数ｎから、ｎよりも小さい第二の整数ｓを減じて、第三の整数ｎ’＝ｎ－ｓを求め、
加算手段が、第四の整数ｍについて、前記第三の整数ｎ'のｍ変数の表現を桁毎にｓのｍ変数の表現に加算して、ｎの符号なしの再コード化された表現を生成する、
ステップを含むことを特徴とする方法。
【請求項２】
前記ｍは、ｍ＝２^{k}であり、
ｋは、第五の整数を表す、
請求項２記載の方法。
【請求項３】
【数９】

$$s = \sum_{i=0}^{l-2} s_i m^i$$

であり、前記Ｉはｎのｍ変数の長さを示す、
請求項１又は２記載の方法。
【請求項４】
０＜α＜ｍについて、$s_i = \alpha$である、
請求項３記載の方法。
【請求項５】
α＝１である、
請求項４記載の方法。
【請求項６】
α＝ｍ－１である、
請求項４記載の方法。
【請求項７】
暗号指数アルゴリズムの指数である第一の正の整数ｎを正則に再コード化する装置であって、
当該装置はプロセッサを備え、
前記プロセッサは、
インターフェースよって受け取られる前記第一の正の整数ｎから、ｎよりも小さい第二の整数ｓを減じて、第三の整数ｎ’＝ｎ－ｓを求める減算手段と、
第四の整数ｍについて、第三の整数ｎ'のｍ変数の表現を桁毎にｓのｍ変数の表現に加算して、ｎの符号なしの再コード化された表現を生成する加算手段と
を有する、ことを特徴とする装置。
【請求項８】
プロセッサにより実行されたとき、該プロセッサに請求項１乃至６の何れか記載の方法を実行させる命令を含むコンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、デジットリコーディング（digit recoding）に関し、より詳細には、符号なしのデジットリコーディングに関する。
【背景技術】
【０００２】
このセクションは、以下に記載及び／又は特許請求される本発明の様々な態様に関連する様々な従来の態様を読者に紹介することが意図される。この説明は、本発明の様々な態様の良好な理解を容易にするため、読者に背景技術を提供することに役立つと考えられる。したがって、これらの説明は、この観点から読まれるべきであり、従来技術の承認として読まれるべきではない。
【０００３】
暗号指数アルゴリズム（cryptographic exponentiation algorithms）は、サイドチャンネル攻撃（side channel attack）に対して脆弱であることが示されている。（M.J. Wiener, editor Advances in Cryptography － CRYPTO ’99, volume 1666 of Lecture Notes in Computer Science, pages 388－398, Springer Verlag 1999における） “Differential Power Analysis”では、Paul Kocher, Joshua Jaffe及びBenjamin Junは、電力消費の観察を使用した攻撃を記載しており、電磁気放射の観察を使用した攻撃は、Karine Gandolfi, Christophe Mourtel 及びFrancis Olivier による（C.K. Koc, D.Naccache及びC. Paar, editors, Cryptographic Hardware and Embedded Systems － CHES2001, volume 2162 of Lecture Notes in Computer Science, pages 251－261, Springer Verlag 2001における）“Electromagnetic Analysis: Concrete Results”、並びに、Jean－Jacques Quisquater及びDavid Samydeによる（I.Attai及びT.P. Jensen, editors, Smart Card Programming and Security (E－Smart 2001), volume 2140 of Lacture Notes in Com

puter Science, pages 200－210, Springer Verlag 2001における）“Electromagnetic Analysis (EMA): Measures and Counter－Measures for Smart Cards”において記載されている。

【０００４】

これらの攻撃は、単純電力解析（SPA: Simple Power Analysis）及び単純電磁波解析（SEMA: Simple Electromagnetic Analysis）と呼ばれ、必要とされる演算が指数（exponent）のビット単位の表現に依存するので、素朴にインプリメントされる指数アルゴリズムにおいて使用される指数を示す。

【０００５】

リコーディングアルゴリズム（recording algorithm）は、指数を計算するために必要とされる演算数を減少するために開発されている。最も一般的に知られる例は、非隣接形式（NAF: Non－Adjacent Form）リコーディングであり、Ian Blake, Gadiel Seroussi及びNigel Smartによる“Elliptic Curves in Cryptography”（volume 265 of London Mathematical Society Lecture Note Series. Cambridge University Press. 1999）に記載される。NAFリコーディングは、{－1,0,1} における値を使用して指数のビットをリコーディングする。これにより、その後の指数アルゴリズムで必要とされる乗算の数が低減される。Donald E. Knuth in The Art of Computer Programming （volume 2/ Seminumerical Algorithms. Addison－Wesley, 2^{nd} edition, 1981）により記載されるｍ変数（ｍ－ary）リコーディングに一般化することができる。しかし、これらのリコーディングアルゴリズムは、指数アルゴリズムの効率を増加するために設計されており、サイドチャネル攻撃に対する耐性を高めるために設計されていない。

【０００６】

幾つかのリコーディングアルゴリズムが提案されている。

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

Bodo Moller. "Fractional Windows（登録商標） Revisited: Improved Signed－Digit Representation for Efficient Exponentiation”, C. Park及びS. Chee, editors, Information Security and Cryptography － ICISC 2004, volume 3506 of Lecture Notes in Computer Science, pages 137－153, Springer Verlag 2004.

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

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

【０００７】

しかし、Yasuyuki Sakai及びKouichi Sakuraiにより（M. Joye及び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における）“A New Attack with Side Channel Leakage During Exponent Recoding Computations”で述べられるように、正則の指数アルゴリズム（regular exponentiation algorithm）を達成するため、使用されるリコーディングア

ルゴリズムも正則である必要がある。正則なリコーディングアルゴリズムでは、アルゴリズムの評価の間にメインループにおけるテストが存在しない。

【０００８】

指数が生成されるときにリコーディングが実行されるべきであることが議論される場合、これは、たとえば指数がランダム値と結合される場合に不可能である。それは、リコーディングは指数の直前に実行される必要があるからである。ランダム値との結合は、Jean－Sebastien Coronにより（C.K. Koc及びC. Paar, editors, Cryptographic Hardware and Embedded Systems － CHES ’99, volume 1717 of Lecture Notes in Computer Science, pages 292－302, Springer Verlag 1999における）“Resistance against Differential power Analysis for Elliptic Curve Cryptosystems”、並びに、Paul Kocherによる（N. Koblitz, editor, Advances in Cryptology － CRYPTO ’96, volume 1109 of Lecture Notes in Computer Science, pages 104－113, Springer Verlag 1996における）“Timing Attacks on Implementations of Diffie－Hellman, RSA, DSS, and Other Systems”に記載されるように、所定のサイドチャネル解析を防止するために行われる。

【０００９】

他のリコーディングアルゴリズムは、指数を正則にするために提案されている。Bodo Mollerは、（G. Davida及びY. Frankel, editors, Information Security (ISC2001), volume 2200 of Lecture Notes in Computer Science, pages 324－334, Springer Verlag 2001における）“Securing Elliptic Curve Point Multiplication against Side－Channel Attacks”において、ｍ変数の指数のリコーディングアルゴリズムを記載している。ゼロに等しいそれぞれの桁は、－ｍで置き換えられ、次の最上位桁は、１だけインクリメントされる。これにより、セット{1,…,m－1}U{－m}に含まれる桁で再コード化される指数となる。ｍ変数の指数アルゴリズムと結合され、これは、x^{-m}が予め計算されるべきであることを意味する。この計算は楕円曲線に関して「容易」であるが、有限環の乗法群については容易ではない。

【００１０】

符号なしのバージョンのMollerのアルゴリズムは、camille Vuillaume及びKatsuyuki Okeyaによる（J. Zhou, M. Yung及びF. Bao, editors, Applied Cryptography and Network Security － ACNS 2006, volume 3989 of Lecture Notes in Computer Science, pages 268－283, Springer Verlag 2006における）“Flexible Exponentiation With Resistance to Side Channel Attacks”で記載されている。桁はセット{1,...,m}において再コード化され、それぞれのゼロの桁は、ｍで置き換えられ、次の桁は、１だけデクリメントされる。

【特許文献１】国際特許出願WO2008/069387A明細書

【発明の開示】

【発明が解決しようとする課題】

【００１１】

符号あり及び符号なしのバージョンのMollerアルゴリズムの問題点は、正則な様式で容易に実現することができないことである。

【００１２】

したがって、指数が正則の様式で簡単に再コード化される正則の指数のためのリコーディングアルゴリズムが必要とされる。

【課題を解決するための手段】

【００１３】

第一の態様では、本発明は、暗号指数アルゴリズムの指数である第一の正の整数ｎを再コード化する正則な方法に向けられる。プロセッサは、ｎよりも小さい第二の整数ｓを選択し、第三の整数ｎ’＝ｎ－ｓを定義し、第四の整数ｍについて、第三の整数ｎ'のｍ変数（ｍ－ａｒｙ）の表現を桁毎にｓのｍ変数の表現に加算して、ｎの再コード化された表現を生成する。

【００１４】
第一の好適な実施の形態では、ｍ＝２^{k}である。
【００１５】
第二の好適な実施の形態では、
【数１】

$$s = \sum_{i=0}^{l-2} s_i m^i$$

であり、この場合、Ｉはｎのｍ変数の長さを示す。０＜α＜ｍについて、ｓ$_{i}$＝αであり、この場合、好ましくはα＝１又はα＝ｍ－１である。
【００１６】
第二の態様では、本発明は、第一の正の整数ｎを正則に再コード化する装置に向けられる。本装置は、プロセッサを備えており、このプロセッサは、ｎよりも小さい第二の整数ｓを選択し、第三の整数ｎ’＝ｎ－ｓを定義し、第四の整数ｍについて、第三の整数ｎ'のｍ変数（ｍ－ａｒｙ）の表現を桁毎にｓのｍ変数の表現に加算して、ｎの再コード化された表現を生成する。
【００１７】
第三の態様では、本発明は、プロセッサにより実行されたとき、本発明の第一の態様の方法を実行する命令を記憶するコンピュータプログラムプロダクトに向けられる。
【図面の簡単な説明】
【００１８】
以下、添付図面を参照して、限定するものではない例を通して、本発明の好適な特徴が記載される。
【図１】本発明の好適な実施の形態に係るデジットリコーディングの装置を示すブロック図である。　図面において、表現されたブロックは機能的なエンティティであり、これらは物理的に個別のエンティティに必ずしも対応していない場合がある。これらの機能的なエンティティは、ハードウェア、ソフトウェア、又は、ソフトウェア及びハードウェアの組み合わせとして実現され、さらに、１以上の集積回路で実現される場合がある。
【発明を実施するための最良の形態】
【００１９】
図１は、桁、特に指数アルゴリズムで使用されるべき指数の桁を再コード化する装置１００を示す。本装置１００は、以下に記載される実施の形態のリコーディングアルゴリズムの計算を実施するコンピュータプログラムを実行する少なくとも１つのプロセッサ１１０（以下、プロセッサ）を有する。なお、プロセッサ１１０は、ハードウェア、又はソフトウェアとハードウェアとの組み合わせで実現される場合もある。本装置１００は、たとえばプロセッサ１１０からの中間の計算結果のようなデータを記憶するメモリ１２０を更に有する。また、装置１００は、他の装置（図示せず）とのインタラクション用の少なくとも１つのインタフェース１３０（以下、「インタフェース」）を有する。図１は、プロセッサ１１０により実行されたとき、本発明の方法の２つの実施の形態の何れかに係るリコーディングアルゴリズムを実行するコンピュータプログラムを記憶する、たとえばＣＤ－ＲＯＭのようなコンピュータプログラムプロダクト１４０を更に例示する。
【００２０】
指数において、（乗算で書かれた）群において、整数ｎ及びエレメントｘについてｚ＝ｘ^{n}が計算される。
【００２１】

【数２】

$$n=\sum_{i=0}^{l-1}d_i m^i$$

を基数ｍ（典型的に、ｍ＝２^{k}）におけるｎの展開を示し、この場合、Ｉはｎのｍ変数の長さである。正の整数ｓ＜ｎを取り、ｎ’＝ｎ－ｓを定義する。

【００２２】

【数３】

$$n'=\sum_{i=0}^{l-1}d'_i m^i \text{ 及び } s=\sum_{i=0}^{l-1}s_i m^i$$

がそれぞれｎ’及びｓのｍの展開を示す場合、ｘ^{n}＝ｘ$^{n'+s}$に従い、この場合、

【数４】

$$n'+s=\sum_{i=0}^{l-1}k_i m^i$$

であり、この場合、ｋ$_{i}$＝ｄ’$_{i}$＋ｓ$_{i}$である。

【００２３】

基数ｍにおけるｓの最上位桁をゼロと定義する場合、基数ｍにおけるｎ’の最上位桁（すあなわちｋ$_{Ｉ－１}$）は、ゼロよりも大きいか又はゼロに等しい。これが当てはまらない場合、再コード化は、符号なしではなく、従って、計算の反転が高価な群にとって適切ではない。

【００２４】

［第一の好適な実施の形態］

αを０＜α＜ｍを満たす整数とする。

【００２５】

【数５】

$$s=\sum_{i=0}^{l-2}\alpha m^i=\alpha\frac{m^{l-1}-1}{m-1}$$

を選択する。これは、ｓの全ての桁を同じ値、すなわちαに設定することとして見られる場合がある。ｎ'$_{i}$∈｛０，．．．，ｍ－１｝であるので、ｋ$_{i}$∈｛α，．．．，α＋（ｍ－１）｝に従う。次いで、以下のアルゴリズムが再コード化のために使用される。

【００２６】

入力：ｎ≧１，ｍ＝２^{k}，Ｉ（ｎのｍ変数の長さ）。

出力：ｎ＝（ｋ$_{Ｉ－１}$，．．．，ｋ$_{０}$）$_{ｍ}$，ｋ$_{i}$∈｛α，．．．，α＋（ｍ－１）｝、０≦ｉ≦Ｉ－２．

アルゴリズム：

【００２７】

【表１】

$$s \leftarrow \alpha \frac{m^{l-1}-1}{m-1} \text{ and } n \leftarrow n-s$$

for $i = 0$ **to** $l-2$ **do**

$d \leftarrow n \bmod m$

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

$k_i \leftarrow d + \alpha$

end

$k_{l-1} \leftarrow n$

αについて第一の好適な選択は１である。これは、再コード化された桁について小さな値となるからである。αの第二の好適な値はｍ－１である。これは、ｓ＝ｍ$^{Ｉ－１}$－１を与えるからである（すなわち、１に設定される一連のｋ（Ｉ－１））。

【００２８】

以下、２つの例は、第一の好適な実施の形態を説明する。２つの例について、パラメータは、以下の値をとる。

ｋ＝２

ｍ＝４

ｎ＝７３＝（1,0,2,1）$_4$＝１・４^0＋２・４^1＋０・４^2＋１・４^3

Ｉ＝４。

【００２９】

第一の例ではα＝１であり、第二の例ではα＝ｍ－１＝３である。

【００３０】

［第一の実施の形態の第一の例（α＝１）］

【００３１】

【表２】

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

$n := n - s = 73 - 21 = 52$

loop: for $i = 0$ to $l-2$, i.e. for $i = 0$ to 2

$i = 0$:

$d := n \bmod m = 52 \bmod 4 = 0$

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

$k_0 := d + \alpha = 0 + 1 = 1$

$i = 1$:

$d := n \bmod m = 13 \bmod 4 = 1$

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

$k_1 := d + \alpha = 1 + 1 = 2$

$i = 2$:

$d := n \bmod m = 3 \bmod 4 = 3$

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

$k_2 := d + \alpha = 3 + 1 = 4$

$k_3 := n = 0$

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

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

期待されるように、再コード化されたｎは、オリジナルのｎに等しい。

【００３２】

［第一の実施の形態の第一の例（α＝ｍ－１＝３）］

【００３３】

【表３】

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

$n := n - s = 73 - 63 = 10$

loop: for $i = 0$ to $l-2$, i.e. for $i = 0$ to 2

$i = 0$:

$d := n \bmod m = 10 \bmod 4 = 2$

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

$k_0 := d + \alpha = 2 + 3 = 5$

$i = 1$:

$d := n \bmod m = 2 \bmod 4 = 2$

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

$k_1 := d + \alpha = 2 + 3 = 5$

$i = 2$:

$d := n \bmod m = 0 \bmod 4 = 0$

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

$k_2 := d + \alpha = 0 + 3 = 3$

$k_3 := n = 0$

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

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

期待されるように、再コード化されたｎは、オリジナルのｎに等しい。

【００３４】

なお、第一の実施の形態に係るアルゴリズムは実現するのが容易であるが、前もって（Ｉの）ｎのｍ変数の情報を必要とする。これは問題となる場合があるので、実現するのが僅かに複雑になるが、第二の好適な実施の形態はこの問題を克服する。

【００３５】

［第二の好適な実施の形態］

減算ステップｎ’＝ｎ－ｓを更に詳細に見る場合、以下の式を設定する。

【００３６】

【数６】

$$d'_i = (d_i - s_i + \gamma_i) \bmod m \quad \text{及び} \quad \gamma_{i+1} = \left\lfloor \frac{d_i - s_i + \gamma_i}{m} \right\rfloor \in \{-1,0\}$$

この場合、“borrow”は０に初期化され、すなわち$\gamma_0 = 0$である。これは、学校で習得

する古典的な減算アルゴリズムである。d_i，$s_i \in \{0, \ldots, m-1\}$ であるので、これにより、$k_i = d'_i + s_i$ が与えられ、これは $d_i + \gamma_i \geqq s_i$ の場合に $d_i + \gamma_i$ に等しく、さもなければ $d_i + \gamma_i + m$ に等しい。

【００３７】

したがって、$s_i \neq 0$ の任意の選択について、$d_i \in \{0, 1\}$ のとき、k_i について非ゼロの値となる。第一の好適な実施の形態におけるように、$0 < \alpha < m$ について

【数７】

$$s = \sum_{i=0}^{I-2} \alpha m^i$$

である。さらに、符号なしの計算のみを使用するため、$\gamma'_i = \gamma_i + 1 \in \{0, 1\}$ である。

【００３８】

【数８】

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

入力：$n \geqq 1$、$m = 2^k$、$0 < \alpha < m$

出力：$n = (k_{l-1}, \ldots, k_0)_m$，$k_i \in \{\alpha, \ldots, \alpha + (m-1)\}$、$0 \leqq i \leqq I-2$

アルゴリズム：

【００３９】

【表４】

$i \leftarrow 0;\ \gamma' \leftarrow 1$

while $n \geq (m + \alpha)$ **do**

$d \leftarrow n \bmod m$

$d' \leftarrow d + \gamma' + m - \alpha - 1$

$k_i \leftarrow (d' \bmod m) + \alpha$

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

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

$i \leftarrow i + 1$

end

$k_i \leftarrow n + \gamma' - 1$

第一の好適な実施の形態におけるように、α の好適な選択は１及び $m-1$ である。

【００４０】

以下、２つの例は、第二の好適な実施の形態を示す。２つの例について、パラメータは以下の値をとる。

【００４１】

k＝2
m＝4
n＝７３＝（1,0,2,1）$_4$＝１・4^0＋２・4^1＋０・4^2＋１・4^3
第一の例ではα＝１であり、第二の例ではα＝m－１＝３である。
【００４２】
［第二の実施の形態の第一の例（α＝１）］
【００４３】
【表５】

$i := 0$

$\gamma' = 1$

$n = 73 \geq (m + \alpha) = 4 + 1 = 5$, so the while-loop is executed

$d := n \bmod m = 73 \bmod 4 = 1$

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

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

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

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

$i := i + 1 = 0 + 1 = 1$

$n = 18 \geq (m + \alpha) = 4 + 1 = 5$, so the while-loop is executed again

$d := n \bmod m = 18 \bmod 4 = 2$

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

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

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

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

$i := i + 1 = 1 + 1 = 2$

$n = 4 < (m + \alpha) = 4 + 1 = 5$, so the while-loop is NOT executed again

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

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

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

期待されるように、再コード化されたｎは、オリジナルのｎに等しい。
【００４４】
［第二の実施の形態の第二の例（α＝m－１＝３）］
【００４５】

【表６】

$i := 0$

$\gamma' = 1$

$n = 73 \geq (m + \alpha) = 4 + 3 = 7$, so the while-loop is executed

$d := n \bmod m = 73 \bmod 4 = 1$

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

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

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

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

$i := i + 1 = 0 + 1 = 1$

$n = 18 \geq (m + \alpha) = 4 + 1 = 5$, so the while-loop is executed again

$d := n \bmod m = 18 \bmod 4 = 2$

$d' := d + \gamma' + m - \alpha - 1 = 2 + 0 + 4 - 3 - 1 = 2$

$k_1 := (d' \bmod m) + \alpha = (2 \bmod 4) + 3 = 2 + 3 = 5$

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

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

$i := i + 1 = 1 + 1 = 2$

$n = 4 < (m + \alpha) = 4 + 1 = 5$, so the while-loop is NOT executed again

$k_2 := n + \gamma' - 1 = 4 + 0 - 1 = 3$

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

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

期待されるように、再コード化されたｎは、オリジナルのｎに等しい。

【００４６】

両方の実施の形態は、予想通りに、同じ入力について同じ再コード化された数字を与える。たとえば、第一の例は、両方の実施の形態について（４，２，１）を与え、第二の例は、両方の実施の形態について（３，５，５）を与える。

【００４７】

また、両方の実施の形態は、メインループ内でテストが存在しないので正則であり、第一の実施の形態ではｆｏｒループ内でテストが存在せず、第二の実施の形態ではｗｈｉｌｅループ内でテストが存在しない。

【００４８】

本発明は、正の整数の正則なリコーディングを可能にすることを理解されたい。

【００４９】

明細書及び（必要に応じて）特許請求の範囲及び図面で開示されるそれぞれの特徴は、

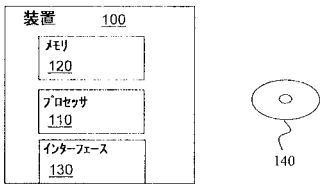
独立に提供されるか、又は任意の適切な組み合わせで提供される場合がある。ハードウェアで実現されるように記載される特徴は、ソフトウェアで実現される場合もあり、逆も然りである。コネクションは、必要に応じて、ワイヤレスコネクション又はワイヤドコネクションとして実現される場合があり、必ずしもダイレクトコネクション又は専用コネクションである必要はない。

【符号の説明】

【００５０】

１００：装置
１１０：プロセッサ
１２０：メモリ
１３０：インタフェース
１４０：コンピュータプログラムプロダクト

【図１】



フロントページの続き

(72)発明者　マルク　ジョイエ
フランス国　３５５１０　セソン・セヴィニェ　アヴェニュ・ド・カラデュ　５５

審査官　金沢　史明

(56)参考文献　特開２００７－１８７９０８（ＪＰ，Ａ）
特開２００５－０２０７３５（ＪＰ，Ａ）
特表２００８－５２５８３４（ＪＰ，Ａ）
米国特許出願公開第２００４／０１６２８６８（ＵＳ，Ａ１）
米国特許出願公開第２００４／０２１５６８４（ＵＳ，Ａ１）
米国特許出願公開第２００６／０１３３６０３（ＵＳ，Ａ１）
K. Okeya et al., Signed Binary Representations Revisited, Cryptology ePrint Archive, International Association for Cryptologic Research (IACR), ２００４年　８月, Report 2004/195, ［２０１３年８月３０日検索］, インターネット, ＵＲＬ, http://eprint.iacr.org/2004/195.pdf
C. Clavier et al., Universal Exponentiation Algorithm, Lecture Notes in Computer Science (LNCS), Springer-Verlag, ２００１年, Vol. 2162, pp. 300-308
M. Joye et al., Exponent Recoding and Regular Exponentiation Algorithms, Lecture Notes in Computer Science (LNCS), Springer-Verlag, ２００９年, Vol. 5580, pp. 334-349

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０９Ｃ　　１／００
Ｈ０４Ｌ　　９／００－　９／３８
Ｇ０６Ｆ　　７／７２