(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4112373号
(P4112373)
(45)発行日　平成20年7月2日(2008.7.2)　(24)登録日　平成20年4月18日(2008.4.18)

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

請求項の数 11 (全 22 頁)

(21)出願番号	特願2002-582551 (P2002-582551)	(73)特許権者	503379380
(86)(22)出願日	平成14年4月15日 (2002.4.15)		ヤコブソン，ビヨルン，マーカス
(65)公表番号	特表2004-537062 (P2004-537062A)		アメリカ合衆国 ０７０３０ ニュージャージイ，ホボケン，ガーデン ストリート １２０３
(43)公表日	平成16年12月9日 (2004.12.9)	(74)代理人	100064447
(86)国際出願番号	PCT/US2002/011658		弁理士 岡部 正夫
(87)国際公開番号	WO2002/084944	(74)代理人	100085176
(87)国際公開日	平成14年10月24日 (2002.10.24)		弁理士 加藤 伸晃
審査請求日	平成17年1月27日 (2005.1.27)	(74)代理人	100106703
(31)優先権主張番号	60/284,001		弁理士 産形 和央
(32)優先日	平成13年4月16日 (2001.4.16)	(74)代理人	100096943
(33)優先権主張国	米国(US)		弁理士 臼井 伸一
(31)優先権主張番号	09/969,833	(74)代理人	100091889
(32)優先日	平成13年10月3日 (2001.10.3)		弁理士 藤野 育男
(33)優先権主張国	米国(US)		

最終頁に続く

(54)【発明の名称】暗号アプリケーションにおける一方向チェーンの効率的計算方法および装置

(57)【特許請求の範囲】

【請求項１】

１つのプロセッサによって実行される方法であって、該プロセッサは、メモリに接続されており、該メモリは、一方向チェーンの複数の値を格納するのに利用可能な指定された記憶量を有しており、該利用可能な指定された記憶量は、該一方向チェーンの全ての値を同時に記憶するのに要求される記憶量よりの小さく、該方法は、

該一方向チェーンの該値の部分集合をヘルパ値として該メモリ内に格納することにより、該部分集合には属していない該一方向チェーンの他の値の計算を容易にするステップと、

値の該部分集合に属する値のうちの１つを使用して、該部分集合には属していない該一方向チェーンの他の値のうちの１つを計算するステップと、

該部分集合には属していない該計算された値によって決定される１つの暗号化出力を生成するステップと、

該一方向チェーンの値の該格納された部分集合を更新して、該ヘルパ値の少なくとも１つを以前に部分集合の一部ではなかった新たなヘルパ値と置き換えるステップとを含む方法。

【請求項２】

請求項１に記載の方法において、前記一方向チェーンの前記値の生成に関連する記憶計算積は、ｓを前記チェーンの長さを示すものとした場合に、複雑度Ｏ（（ｌｏｇ（ｓ））^{2}）を有する方法。

【請求項３】
請求項１に記載の方法において、初期のヘルパ値として格納された該一方向チェーンの複数の値の初期の部分集合は、ｓを前記チェーンの長さとした場合に、
０≦ｊ≦ｌｏｇ$_2$ｓに対して、ｉ＝２^{j}
によって与えられるチェーン内の位置に存在する値からなる方法。
【請求項４】
請求項１に記載の方法において、該暗号化出力は、パスワード、暗号鍵、デジタル署名、及び認証コードのうちの少なくとも１つからなる方法。
【請求項５】
請求項１に記載の方法において、該ヘルパ値の各々は、関連するヘルパ値に加えて、該チェーンにおける目的地の位置、優先度指示子、及び状態を含む対応するペグの形態で格納される方法。
【請求項６】
請求項１に記載の方法において、１つの対応する値v_iが、該一方向チェーン内の所定の位置ｉについて計算され、１つ又は複数のヘルパ値が、指定された計算バジェット内で決定される方法。
【請求項７】
請求項５に記載の方法において、それぞれのヘルパ値を格納するのに使用されるペグの数は、

$$\sigma+\phi\log_2(\sigma+1)_k$$

によって与えられ、$s=2^{\sigma}$は、チェーンの長さである方法。
【請求項８】
請求項６に記載の方法において、該指定された計算バジェットは、

$$b=\lambda\sigma/2\mu$$

で与えられ、$s=2^{\sigma}$は、チェーンの長さである方法。
【請求項９】
装置であって、
１つのプロセッサと、
該プロセッサに接続され、１つの一方向チェーンの複数の値を格納するのに利用可能である指定された記憶量を有するメモリとを含み、該利用可能な指定された記憶量は、該一方向チェーンの全ての値を同時に記憶するのに要求される記憶量よりの小さく、
該プロセッサは、該一方向チェーンの該値の部分集合をヘルパ値として該メモリ内に格納することにより、該部分集合には属していない該一方向チェーンの他の値の計算を容易にし、値の該部分集合に属する値のうちの１つを使用して、該部分集合には属していない該一方向チェーンの他の値のうちの１つを計算し、該部分集合には属していない該計算された値によって決定される１つの暗号化出力を生成し、該一方向チェーンの値の該格納された部分集合を更新して、該ヘルパ値の少なくとも１つを以前に部分集合の一部ではなかった新たなヘルパ値と置き換えるように構成される装置。
【請求項１０】
１つの一方向チェーンの複数の値を生成する際に１つのプロセッサによって実行される１つ又は複数のプログラムを格納するコンピュータ可読媒体であって、該プロセッサは、メモリに接続されており、該メモリは、一方向チェーンの複数の値を格納するのに利用可能な指定された記憶量を有しており、該利用可能な指定された記憶量は、該一方向チェーンの全ての値を同時に記憶するのに要求される記憶量よりの小さく、実行されたときに、該１つ又は複数のプログラムは、該プロセッサに、
該一方向チェーンの該値の部分集合をヘルパ値として該メモリ内に格納することにより、該部分集合には属していない該一方向チェーンの他の値の計算を容易にするステップと、
値の該部分集合に属する値のうちの１つを使用して、該部分集合には属していない該一方向チェーンの他の値のうちの１つを計算するステップと、

該部分集合には属していない該計算された値によって決定される１つの暗号化出力を生成するステップと、

該一方向チェーンの値の該格納された部分集合を更新して、該ヘルパ値の少なくとも１つを以前に部分集合の一部ではなかった新たなヘルパ値と置き換えるステップとを実行させるコンピュータ可読媒体。

【請求項１１】

１つのプロセッサによって実行される方法であって、該プロセッサは、メモリに接続されており、該メモリは、一方向チェーンの複数の値を格納するのに利用可能な指定された記憶量を有しており、該利用可能な指定された記憶量は、該一方向チェーンの全ての値を同時に記憶するのに要求される記憶量よりの小さく、該方法は、

該一方向チェーンの該複数の値の第１の部分集合を初期ヘルパ値として該メモリ内に格納することにより、該第１の部分集合には属していない該一方向チェーンの他の値の計算を容易にするステップと、

値の該第１の部分集合に属する値のうちの１つを使用して、該第１の部分集合には属していない該一方向チェーンの他の値のうちの１つを計算するステップと、

該第１の部分集合には属していない該計算された値によって決定される１つの暗号化出力を生成するステップと、

該一方向チェーンの値の該第１の部分集合とは異なる、該一方向チェーンの第２の部分集合を更新されたヘルパ値として該メモリ内に格納することにより、該第２の部分集合には属しない該一方向チェーンの他の値の計算を容易にするステップとを含み、

該一方向チェーンの値の該第１の部分集合を格納するのに使用される該利用可能な指定された記憶量の少なくとも一部分は、該一方向チェーンの値の該第２の部分集合を格納するのに再利用される方法。

【発明の詳細な説明】

【技術分野】

【０００１】

［発明の分野］

本発明は、包括的には暗号の分野に関し、詳細には、例えば、暗号化、復号、デジタル署名、メッセージ認証、ユーザおよびデバイスの認証、マイクロペイメントなどの暗号アプリケーションにおける一方向チェーンおよび他のタイプの一方向グラフの連続値を計算する技法に関する。

【０００２】

［優先権主張］

本出願は、発明者Bjorn Markus Jakobssonの名前で２００１年４月１６日に出願された「Method and Apparatus for Efficiently Representing and Computing One-Way Chains」という発明の名称の米国仮出願第６０／２８４，００１号の優先権を主張する。この仮出願の開示は、参照により本明細書に援用される。

【０００３】

［発明の背景］

一方向関数は、値ｘを与えると、値ｙ＝ｆ（ｘ）を計算できるが、特定の一方向関数のみがいわゆる「トラップドア」を有する場合に、トラップドアが知られていないと、ｙを与えても、値ｘを計算することが計算上不可能な関数ｆである。上記状況において、値ｘは、関数ｆに関してｙのプリイメージ（pre-image）と呼ばれ、値ｙは、関数ｆに関してｘのイメージと呼ばれる。一方向関数の例として、周知のＳＨＡ－１ハッシュアルゴリズムおよびＭＤ５ハッシュアルゴリズムといったハッシュ関数、ならびにメッセージ認証コード（ＭＡＣ（message authentication code））を生成する関数が含まれる。これらの一方向関数および他の一方向関数のさらに詳細な内容については、A.J. Menezes等著の「Handbook of Applied Cryptography」CRC Press, 1997を参照されたい。この文献は、参照により本明細書に援用される。

【０００４】

多くの暗号アプリケーションでは、いわゆる一方向チェーンを計算することが望ましい。これは、$\nu_{i-1}=f(\nu_i)$となるような値$\nu_1 \cdots \nu_s$の列である。より詳細には、ｇを、ハッシュチェーンまたは他の一方向関数ｈの出力のサイズの入力を関数ｈの入力のサイズに写像する関数とすると、$\nu_{i-1}=f(g(\nu_i))$となる。特に、当業者に周知のように、ｇは、適度な長さにするための情報の切り捨て、適度な長さにするための情報の付け足し、または他の同様の写像関数とすることができる。ハッシュ関数が受け付けるように、ｈが、任意の長さの入力を受け付ける関数である場合に、関数ｇを使用する必要はないことも知られている。あるいは、このような状況のｇは、恒等関数であると言うことができる。関数ｇは、必要ならば、一方向関数ｈの機能に適切に組み込まれていると理解することができるので、本明細書で使用される表記を簡単にするために、関数ｇの潜在的な使用は、明示的には示されない。

【０００５】

また、チェーンの異なるステップに対して異なる関数を使用できることにも留意すべきである。ここで、チェーンの「ステップ」とは、前のチェーン値からの所与のチェーンの出力値の生成をいう。同様に、さまざまなステップへの補助入力を認めることができる。説明を簡単にするために、これらの特定の変形は、本明細書では明示的に記述されないが、当業者は、一方向チェーンが、これらの変形または他の変形を使用して構成できることを認識するであろう。

【０００６】

上述したタイプの一方向チェーンは、値ν_sから開始し、その値から、ν_sに一方向関数を適用することによりν_{s-1}を計算し、次に、ν_{s-1}に一方向関数を適用することによりν_{s-2}を計算するということを繰り返すことにより計算することができる。値ｙは、チェーンの次の「リンク」で、一方向関数の入力として使用されるので、これは、上記値$y=f(x)$の計算の一般的な場合である。このようなチェーンを使用する１つの重要な理由は、時間を表すためである。例えば、一方向チェーン$\nu_1 \cdots \nu_s$が、第１の関係者によって終点の値ν_sから計算されて、チェーン値ν_1が、第２の関係者に与えられると、第１の関係者は、連続したプリイメージν_2、ν_3などを第２の関係者に示すことによって、「時間をインクリメント」することができる。第２の関係者は、単独では、ν_1からこれらの連続したプリイメージを計算することができないことに留意されたい。しかしながら、プリイメージν_2を与えると、第２の関係者は、$\nu_1=f(\nu_2)$であるかどうかをチェックすることによって、そのプリイメージの正確さを検証することができる。ν_3について、この検証は２つのステップを有する。第１のステップではν_2が計算され、第２のステップではν_1が計算されて既知の値ν_1と比較される。

【０００７】

一方向チェーンの別の既知の使用は、マイクロペイメントアプリケーションにおける金銭または他の支払いメカニズムを代表するものである。例えば、第１の関係者が、一方向チェーンを生成して、例えばチェーンの終点ν_sのデジタル署名を介して一方向チェーンを銀行によって認証させると、第１の関係者は、ν_2を明らかにすることによって１単位の支払いを第２の関係者に支払うことができる。第２の関係者は、この値を銀行に使って、その資金を得ることができる。第１の関係者は、ν_3を明らかにすることによって第２の単位を第２の関係者に支払うことができ、以下同様である。この簡単な例は、支払い先が単一の場合にのみ適用できることに留意されたい。

【０００８】

また、一方向チェーンは、ＭＡＣを有するメッセージの信憑性を検証する鍵の計算にも使用される。これについては、A. Perrig等著の「Efficient and Secure Source Authentication for Multicast」Proceedings of Network and Distributed System Security Symposium NDSS 2001, February 2001、A. Perrig等著の「Efficient Authentication and Signing of Multicast Streams over Lossy Channels」Proc. of IEEE Security and Privacy Symposium SP 2000, May 2000、およびA. Perrig等著の「TESLA: Multicast Source Authentication Transform」Proposed IRTF draft, http://paris.cs.berkeley.edu/~pe

rrigを参照されたい。これらの文献のすべては、参照により本明細書に援用される。このアプローチでは、デバイスは、メッセージおよびその対応するＭＡＣを第１の時間間隔の間に計算してブロードキャストし、次に、後の時間間隔の間に鍵を公開することになる。受信者は、鍵が公開された後にＭＡＣを検証することができ、送信者のみが、ＭＡＣがブロードキャストされた時点の鍵を知ることができた者であるという知識に基づいて、メッセージの信憑性を信頼することになる。鍵は、その鍵にハッシュ関数を適用して、その結果を、同じチェーン、したがってその送信者に関連付けられた最新でない方の鍵と比較することにより検証される。

【０００９】

これらのアプリケーションおよび他のアプリケーションにおける一方向チェーンの従来の使用は、連続したチェーン値を出力するために、それらの値を記憶しなければならないか、または計算しなければならないという点で、重大な欠点に苦しんでいる。それらの値が、計算を行うことなく記憶装置から直接出力される場合には、関係者は一般に、値のすべてを記憶しなければならない。あるいは、値が必要に応じて計算される場合には、関数ｆが、関係者がトラップドアを知っているかまたはトラップドアの使用を望むトラップドア関数でないと仮定して、関係者は、この関数ｆを繰り返し適用することにより、終点v_sから所与の値を計算しなければならない。計算を行うことなく記憶する場合には、関係者は、チェーンの長さｓに比例した記憶装置、換言するとＯ（ｓ）の記憶装置を使用する必要がある。ここで、Ｏ（）は、「オーダ（on the order of）」を示す。必要に応じて計算する場合には、第１の関係者は、Ｏ（ｓ）の計算を使用する必要がある。

【００１０】

従来の一方向チェーンの計算に関連するこの過度の記憶計算積は、例えば携帯電話、スマートカード、携帯情報端末（ＰＤＡ（personal digital assistant））または他のタイプの無線デバイスまたはポータブルデバイスといった、いわゆる「軽量」の処理デバイスにおいて特に問題である。これらのデバイスは、多くの場合、少なくとも部分的にこのようなデバイス内のバッテリ電力を使用することから、一般に、記憶リソースまたは計算リソースが制限されている。他の例として、いわゆる「スマートダスト（smart dust）」または「ダストコンピュータ（dust computer）」が含まれる。これは、地震や軍の調査の目的で大きなエリアを集合的にカバーするために使用することができる非常に小さな計算デバイスをいう。

【００１１】

これらの場合および多くの他の場合において、従来の一方向チェーンの計算に関連する記憶コストまたは計算コストは、標準的な暗号技法をこのようなデバイスで実施することを困難にするか、または不可能にする。

【００１２】

したがって、上記記憶コストおよび計算コストを十分に削減し、これにより、暗号技法の効率性を改善して、軽量の処理デバイスでこのような技法の実施を可能にするように、一方向チェーンおよび他の一方向グラフの連続したプリイメージの値を計算する改良された技法に対する要求が存在する。

【００１３】

［発明の概要］

本発明は、一方向チェーンおよび他の一方向グラフの連続値の効率的な計算の方法および装置を提供することにより、上述した要求を満たす。本発明の一態様によると、ヘルパ値が、チェーン値の生成に関連する記憶計算積を十分に削減する方法で配置され、利用される。より詳細には、記憶計算積は、本発明の技法を使用して、上述した従来のアプローチのようにチェーンの長さｓ自体のオーダ、すなわちＯ（ｓ）ではなく、チェーンの長さｓの対数の２乗のオーダ、すなわちＯ（（ｌｏｇ　ｓ）^{2}）となるように削減される。

【００１４】

本発明の例示の実施の形態では、一方向チェーンは、位置ｉ＝１，２，…ｓを有する長さｓのチェーンであり、各位置は、それに関連付けられた対応する値v_iを有し、値v_i

は、特定のハッシュ関数または他の一方向関数ｈに対して、$v_i = h(v_{i+1})$により与えられる。長さｓの一方向チェーンに対して、例えば、$0 \leqq j \leqq \log_2 s$についてｉ＝2^jによって与えられる位置のヘルパ値の初期分布を記憶することができる。一方向チェーンの現在位置における出力値v_iの所与の１つは、チェーンの現在位置と終点との間における一方向チェーンの別の位置についての事前に記憶された第１のヘルパ値を利用して計算することができる。所与の出力値の計算後、ヘルパ値の位置は、その後の出力値の計算を容易にするために調整される。

【００１５】

本発明の別の態様によると、一方向チェーン内のヘルパ値の初期分布におけるヘルパ値のそれぞれはペグ（peg）に対応し、このペグは、ヘルパ値に加えて、それに関連付けられた他の情報を有する。この他の情報には、チェーンにおける目的地の位置、優先度指示子、および状態が含まれ得る。出力値の所与の１つの計算において、ペグの１つまたは２つ以上のものは、一方向チェーンにおける新しい位置に再配置され、再配置されたペグのいずれに対しても、新しいヘルパ値が計算される。

【００１６】

ペグの再配置プロセスは、チェーンの各位置ｉに対して、対応する出力値v_iを計算できるように、かつ、あらゆるペグを、指定された計算バジェット内で再配置できるように構成されることが好ましい。例えば、例示の実施の形態で一方向チェーンに利用されるペグの個数は、

【数３】

$$\sigma + \lceil \log_2(\sigma+1) \rceil$$

によって近似的に与えることができる。ここで、$s = 2^\sigma$は、チェーンの長さである。指定された計算バジェットは、バジェット

【数４】

$$b = \lfloor \sigma/2 \rfloor$$

によって近似的に与えることができる。より具体的な数値例として、σ＝３１、$s = 2^{31} = 2.147 \times 10^9$である場合に、バジェットｂは１５になり、これは、各チェーン値を計算するのに、最大１５回のハッシュ関数の適用が必要となることを示す。この例では、ｎ＝３６個のペグが必要とされる。これらのペグのそれぞれは、周知のＳＨＡ－１ハッシュ関数の使用を仮定すると、ヘルパ値を記憶するための２０バイトに加えて、状態情報の記憶用に追加の８バイトを使用して実施することができる。この結果、全体で３６×（２０＋８）＝１００８バイトの記憶装置が必要とされる。この例の一方向チェーンが、毎秒１つのチェーン値の出力を必要とするアプリケーションで実施されると、そのチェーンは、６８年より長く持ちこたえることになる。

【００１７】

［発明の詳細な説明］

本明細書では、本発明は、特定の例の一方向チェーンを使用して説明される。しかしながら、本発明の技法は、他のタイプの一方向チェーン、例えばトラップドアを有するまたは有さない他のタイプの一方向関数を使用して構成されるチェーンに、より広く適用できることが理解されるべきである。上記で示したように、チェーンの異なるステップに対して異なる関数を使用することもできるし、ステップの１つまたは２つ以上のものへの補助入力を認めることもできる。その上、本発明の技法は、直線状のチェーン値の列というよりもむしろ、木の形状または他のグラフの形状に配置された値の列にも適用することができる。さらに、図示されて説明される特定の例示の実施の形態は一例にすぎず、所与のアプリケーションの特定の必要性に適応するように再構成できるか、そうでなければ変更できる。

【００１８】

本明細書で使用される用語「一方向関数」は、一例として、プリイメージからイメージ

を計算する方が、イメージからプリイメージを計算することよりも十分に効率的である任意の関数を含むが、これに限定されるものでないことを意図している。例えば、イメージからプリイメージを計算する逆計算の関数は、計算的に高価であるか、不可能であるか、またはそうでなければ計算を行うのが困難である。

【００１９】

本明細書で使用される用語「チェーン」は、一般に、値の直線状の列だけでなく、複数の枝を有し、各枝がそれ自体、値の直線状の列に対応し得る木構造またはグラフ構造も含むように解釈されることを意図している。

【００２０】

用語「一方向チェーン」は、少なくとも１対の値が、一方向関数を介して相互に関連するチェーンをいう。

【００２１】

図１は、本発明の技法を利用した効率的な方法で連続値の列を計算することができる一方向チェーンの一例を示している。この例の一方向チェーンは、総数としてｓ個の要素を含む一方向チェーンを仮定している。各要素は、チェーンの対応する位置に関連付けられる。チェーンの最初の要素は、位置１の要素であり、本明細書ではチェーンの始点ともいう。チェーンの最後の要素は、本明細書ではチェーンの終点ともいい、位置ｓの要素である。ここで、ｓは、チェーンの範囲または長さを示す。位置ｉ＝１，２，…ｓのそれぞれには、図示するような対応する値v_iが関連付けられる。この例の一方向チェーンは、値v_iが、所与の関数ｈに対して次の式、

$$v_i = h(v_{i+1})$$

によって与えられるように構成される。ここでｈは、ハッシュ関数または別のタイプの一方向関数を意味することができると理解されるべきである。このように、一方向チェーンの所与の現在の値v_iは、関数ｈをチェーンの次の値に適用することにより計算される。ここで、この文脈における用語「次」は、値v_{i+1}をいう。上記に示したように、位置ｓの終点の値から開始してチェーンの各値を計算することは、過度に計算集約的（computation intensive）であり、チェーンに沿った各値の記憶は、過度に記憶集約的（storage intensive）である。この問題は、上記「軽量の」処理デバイス、すなわち、限られた計算リソースまたはメモリリソースを有するデバイスでは悪化する。

【００２２】

例示の実施の形態における本発明は、図１の一方向チェーンの所与の値v_iの計算効率を改善する技法を提供する。本発明によると、いわゆるヘルパ値（helper value）が、チェーン値の生成に関連する記憶計算積（storage-computation product）を十分に削減する方法で配置されて利用される。より詳細には、記憶計算積は、上述した従来のアプローチにおけるようなチェーンの長さそのもののオーダではなく、チェーンの長さの対数の２乗のオーダとなるように、本発明の技法を使用して削減される。

【００２３】

本発明の導入として、まず、図１のチェーンの先頭に近い要素の値を計算したい場合を仮定する。チェーンに沿った値が記憶されていない場合には、計算は、チェーンの長さｓに比例した作業量だけ必要となる。ヘルパ値が、値が計算される現在のチェーンの要素からある距離ｄに導入されると、その値の計算コストは、ｄ－１のハッシュ関数の値の評価になる。そして、計算される次の値のコストは、ｄ－２のハッシュ関数の値の評価になる。しかしながら、１つのヘルパ値のみが使用されると仮定すると、ヘルパ値に一旦到達すると、次の値のコストは、チェーンの終点に到達するコストとなる。この簡単な例では、ｄ＝ｓ／２、すなわち、ヘルパ値が、チェーンの中点に位置する場合に、総コストが最小になる。

【００２４】

１つではなく２つのヘルパ値が使用される場合には、全区間を３つの等しい長さの区間に分割することができる。この場合に、次の要素の計算コストの上限は、ｓ／３個のハッシュ関数の評価コストになる。記憶および計算の他のさまざまな線形結合も可能である。

例えば、ｓ≫１００の長さｓのハッシュチェーンの１００番目ごとの値を記憶する場合に、最悪の場合にも、Ｏ（ｓ／１００）の記憶装置しか必要とされず、Ｏ（１００）の計算しか必要とされない。ここで、本明細書で上述したように、Ｏ（）は、「オーダ」を示す。しかしながら、記憶計算積Ｓ×Ｃは、まだ、チェーンの長さｓに比例する。すなわち、Ｓ×ＣはＯ（ｓ）である。ここで、Ｓは、必要とされる記憶量を示し、Ｃは、１つの出力値につき必要とされる最悪の場合の計算量を示す。

【００２５】

全区間が、２つの等しい長さの区間に分割され、次に、これら２つのうちの最初のものが、それ自体２つに分割されると、初期計算コストの上限は、ｓ／４個のハッシュ関数の評価コストになる。この低い初期コストは、全区間の最初の半分に当てはまり、その後、次の要素すなわち終点への距離は、ｓ／２となる。しかしながら、再配置の操作を実行するのに十分な計算リソースが残っていると仮定して、最初のヘルパ値に到達するとすぐに、最初の中点または全体の中点と終点との間の新しい中点にそのヘルパ値を再配置した場合には、ｓ／４個の上限を維持することができる。ここで、３つのヘルパ値が存在する場合には、より多くの値を再配置することになるが、個数の増加により区間の長さが減少するので、計算の上限は、各要素について減少することに留意されたい。約ｌｏｇ　ｓ個のヘルパ値を使用すると、この場合、値が計算される要素ごとに、２つの位置からの最大距離にヘルパ値が存在するので、ヘルパ値の利点が最大になる。

【００２６】

図２Ａは、上述した本発明によるヘルパ値の初期分布を示している。これらのヘルパ値は、本明細書では、一般に「ペグ」とも呼ばれるが、所与のペグは、その特定のヘルパ値だけでなく、後にさらに詳述するように、一定の追加情報も含むことができる。したがって、各ペグは、それに関連付けられた単一のヘルパ値を有するとみなすことができ、そのヘルパ値は、ペグの位置におけるチェーン値v_iである。図２Ａの線図では、図１の長さｓのハッシュチェーンのヘルパ値の初期分布が示され、ヘルパ値、すなわちペグは、位置ｓ、ｓ／２、ｓ／４、ｓ／８などに決定される。より詳細には、ペグは、次の位置、

$0 \leqq j \leqq \log_2 s$に対して、$i = 2^j$

に最初に置かれる。位置2^jで開始するこれらのペグに加えて、一般に、少数の追加ペグを含めることが望ましい。これは、計算プロセスにおいて任意の特定のペグに到達し、その結果、そのペグが再配置に利用可能となる前に、このプロセス中、すぐにペグの移動を開始できるようにするためである。必要なペグの総数は、Ｏ（ｌｏｇ　ｓ）である。

【００２７】

図２Ａと共に説明し記述した特定の初期ペグ配置は、一例にすぎないことが理解されるであろう。上記で与えた例の初期配置と類似した他の初期の場所にペグを置くことが可能である。例えば、初期の位置$i = 2^j + 1$にペグを置くことができるし、それ以外に、記憶と計算との異なるトレードオフを提供するように、ペグを最初に置くことができる。例えば、上記例のように、区間を２つの等しい長さの部分に分割する代わりに、各区間を３つの等しい部分に分割し、次に、３つのうちの１つを３の部分にさらに分割するということを繰り返すことができる。この結果、記憶コストは追加されることになるが、計算コストは減少する。他のタイプの区間の分割も、本発明を実施する際に使用することができる。

【００２８】

動作中、図１の一方向チェーンのチェーン値v_iは、通常、一度に１つずつ計算されて出力される。１つの値は、各時間間隔ｉについて計算されて出力される。図２Ａと共に記述したペグは、この計算を容易にする。より詳細には、各時間間隔ｉの間、その間隔に対する値v_iが出力され、ペグの１つまたは２つ以上が、その後の計算を容易にするために再配置される。ペグの再配置プロセスは、各間隔ｉについて、必要な出力値v_iおよび適切な再配置を、Ｏ（ｌｏｇ　ｓ）でもある所定のバジェット（budget）内で求めることができることを確実にする。

【００２９】

図２Ｂは、本発明によるハッシュチェーンの計算プロセスの流れ図である。ステップ１０では、チェーンの終点および始点が決定される。このステップは、終点の値をランダムに選択し、所望の長さに達するまで、適切なハッシュ関数を繰り返し適用することによって始点の値を計算することにより実施されてもよい。ステップ１２では、初期のペグの位置が決定され、対応するヘルパ値が計算される。次に、ステップ１４では、このプロセスは、ペグのうちの１つの現在位置以外の位置の所与のチェーン値を出力する。この出力は、適切なヘルパ値、すなわち所与のチェーン値の位置にチェーンの終点方向において最も近いヘルパ値を使用して、そのチェーン値を計算することにより行われる。所与のチェーン値が出力された後、ステップ１６は、ペグが再配置されて、新しいヘルパ値が計算されることを示している。図２Ｂのプロセスのより詳細な例については後述する。

【００３０】

所与のペグの再配置は、新しい場所におけるそのペグの対応するヘルパ値ν_iの計算を必要とする。新しい場所におけるヘルパ値の計算コストは、関連付けられたペグが移動される距離に比例する。したがって、新しいペグが位置ｉの既存のペグの上に置かれ、その後、位置ｉ－ｊに移動されると、そのコストは距離ｊに比例する。同様に、ペグが位置ｉに位置する場合（別のペグがこの位置に存在しようがしまいが）、そのペグを位置ｉ－ｊに移動するコストも距離ｊに比例する。これらの双方の場合における距離ｊは、位置ｉの値から位置ｉ－ｊの値を計算するのに必要な一方向関数の評価数に対応する。

【００３１】

しかしながら、例示の実施の形態で最小化されるコストメトリックは、計算だけではなく、前述した記憶計算積Ｓ×Ｃがある。ここでＳは、必要とされる記憶量を示し、Ｃは、１つの出力値につき必要とされる最悪の場合の計算量を示す。これは、その新しい場所から出力される値が計算されるのと同様に、新しい場所におけるヘルパ値が、新しい場所から次の既知の値のハッシュ関数の評価を繰り返すことにより得られるからである。このような値の個数を増加させて、これにより、それらの値の間の距離を減少させることにより、ヘルパ値を再配置するコストを明らかに下げることができるが、本明細書で記述される結果は、一般に、上記記憶計算積の観点で評価されることになる。

【００３２】

バジェットは、計算されて出力される各値ν_iに対して割り当てられることが好ましい。バジェットは、要素ごとの計算の上限に対応する。ハッシュチェーンの各位置について、適切な値ν_iを計算して出力しなければならないので、現在のチェーンの要素の計算が、このバジェットにアクセスする最も高い優先度を有する。残りのあらゆるバジェットは、ヘルパ値の計算に割り当てられる。

【００３３】

好都合なことに、これらのヘルパ値は、高い優先度のヘルパ値と低い優先度のヘルパ値とに分割することができる。例示の実施の形態では、高い優先度のヘルパ値は、一般に、すでに相対的に小さな区間に再配置されて、現在の要素の近くに位置するのに対して、低い優先度のヘルパ値は、これとは逆に、一般に、より大きな距離をおいて現在の要素からより遠くに位置する。低い優先度のヘルパ値には、現在の要素が計算されて、高い優先度のヘルパ値がそれらの必要な分を使い果たした後、すなわちそれらのそれぞれの目的地に到達した後に残っている上記バジェットの低い優先度のヘルパ値の部分のみが割り当てられる。

【００３４】

３つ以上の優先レベルを有することも可能である。本明細書に記述される例示の実施の形態は２つの優先レベルを利用するが、現在の位置までの要素の距離に基づいて、所与のレベル内で要素にさらに優先度をつける。この優先度は、距離が増加するに伴い減少する優先度である。ここで、現在の位置は、値がちょうど出力されたばかりか、または、これからちょうど出力されるチェーンの位置に対応する。

【００３５】

図２Ｂと共に上記に示したように、チェーンの計算プロセスの初期化段階またはセット

アップ段階の間、チェーンの終点の値v_sはランダムに選択することができ、所望の始点は、繰り返されるハッシュ関数の評価により得ることができる。この機能は、ハッシュチェーンの要素の連続値v_iを後に計算して出力することになるデバイスよりも計算の制限が少ないデバイスによって実行することができる。あるいは、同じデバイスを、ハッシュチェーンの初期化と出力v_iの生成との双方に使用することができる。また、初期化は、図２Ａに示すような上記ペグの初期配置も含むことが好ましい。したがって、各ペグは、位置および位置に関連付けられたヘルパ値を有することになる。ここで、所与のペグの位置に対するペグの値は、終点の値v_sまたは終点の方向におけるヘルパ値のハッシュ関数の繰り返し適用により得られる。

【００３６】

次に、処理デバイスおよび対応するシステムの実施態様の一例について、図３および図４をそれぞれ参照して記述する。その後、例示の実施の形態の特定の一方向チェーンの計算プロトコルをより詳細に記述する。

【００３７】

図３は、本発明による一方向チェーンの計算プロセスの少なくとも一部を実施するのに使用することができる一例の処理デバイス１００を示している。この例における処理デバイス１００は、プロセッサ１０２、メモリ１０４、ネットワークインタフェース１０６、および１つまたは２つ以上の入出力（Ｉ／Ｏ）デバイス１０８を含む。デバイス１００は、軽量の処理デバイスに対応していてもよい。軽量の処理デバイスとしては、例えば、携帯電話、スマートカード、携帯情報端末（ＰＤＡ）、ダストコンピュータ、ビデオカメラ、調査デバイス、動き検出器（motion detector）、熱検出器、あるいは他のタイプの無線デバイスまたはポータブルデバイス、あるいはこれらのデバイスまたは他のデバイスの一部または組み合わせといったものである。プロセッサ１０２は、本発明による一方向チェーン計算プロトコルの少なくとも一部を実施するために、メモリ１０４に記憶された１つまたは２つ以上のソフトウェアプログラムを実行する。例えば、プロセッサ１０２は、１つまたは２つ以上の他の情報源から所与のチェーンの始点および終点を受信して、本明細書で記述された方法で連続したチェーン値v_iを計算するように構成することができる。また、プロセッサ１０２は、上記に示したように、終点の値v_sをランダムに選択して、この終点から始点を計算するように構成することもできる。

【００３８】

図４は、複数のクライアントデバイス１１０－１、１１０－２、…１１０－Ｎを組み込んだ一例のシステム構成を示している。各クライアントデバイスは、ネットワーク１１２を介して１つまたは２つ以上のサーバデバイス１１４－１、１１４－２、…１１４－Ｍに接続することができる。クライアント１１０またはサーバ１１４の１つまたは２つ以上のものは、図３のデバイス１００として実施することもできるし、同様の要素の配置を使用して実施することもできる。このタイプのシステム構成では、一方向チェーン値は、クライアント１１０の１つまたは２つ以上のものによって生成されて、サーバの１つまたは２つ以上のものに送信することもできるし、その逆も行うことができる。また、当業者に明らかなように、図４のさまざまな要素間の他のタイプの通信も、本発明の技法を使用して可能である。

【００３９】

図３および図４に示す要素の特定の配置は一例にすぎず、本発明の範囲を決して限定するものとして解釈されるべきでないことは、強調されるべきである。本発明は、処理デバイス（複数可）内のプロセッサおよび関連付けられたメモリといった要素の特定の実施態様に関係なく、プロセッサおよび関連付けられたメモリを有する任意のタイプの処理デバイス（複数可）を使用して、すべてまたは部分的に実施することができる。

【００４０】

次に、本発明による一方向チェーン計算プロトコルの第１の例示の実施の形態について記述する。このプロトコルの例は正確であり、チェーンの長さやチェーンの特定の値に関係なく、正確な値が出力されることを示すことができることを意味する。また、この例は

完全でもあり、これは、計算が、指定された計算の制限を越えることなく実行可能であることを意味する。

【００４１】

この実施の形態では、ハッシュチェーンＨは、例えば図１と共に上述した列のような値の列｛v_1，…，v_i，…v_s｝によって与えられる。ここでv_sは、｛０，１｝^{k}から一様にランダムに選択された値であり、$v_i = h(v_{i+1})$である。また、ここでｈ：｛０，１｝*→｛０，１｝^{k}は、例えば上記ＳＨＡ－１ハッシュ関数またはＭＤ５ハッシュ関数といったハッシュ関数であるか、または、プリイメージからイメージを計算する方が、イメージからプリイメージを計算するよりも十分に効率的な別のタイプの一方向関数もしくは他の関数である。この実施の形態では、鍵がかけられた関数を使用することも可能であるが、関数ｈは公に計算可能、すなわち公開情報のみにアクセスして計算可能であると仮定される。なお、鍵がかけられた関数を使用する場合には、入力の一部は、一般に公開されない。値v_1はチェーンの始点であり、値v_sはチェーンの終点であり、範囲ｓはチェーンの長さ、例えば、生成される連続した要素の個数を示す。この実施の形態では、$\sigma > 2$の整数に対して$s = 2^\sigma$と仮定される。他のチェーンの長さが所望される場合には、所望される長さより大きな上記形式の最小値としてｓを選択することができる。

【００４２】

バジェットｂは、出力されるハッシュチェーンの１つの要素につき許容される計算単位の個数として定義される。ここで、ハッシュ関数の評価のみがカウントされ、プロトコルの実行に関連付けられた他の計算ステップはカウントされない。これは、１つのハッシュ関数の評価を実行する計算労力は、１つの要素につき残っている作業をはるかに上回ることからして妥当である。

【００４３】

各ヘルパ値およびチェーンの終点は、ペグp_jに関連付けられる。各ペグp_jは、チェーン内の位置と、位置に関連付けられた値とを有する。ｐｏｓｉｔｉｏｎをペグの位置とすると、その値は、$v_{position}$である。さらに、各ペグは、目的地（そのペグが進んでいる位置）、優先度（高または低）、および活動ステータスまたは状態（フリー（free）、レディ（ready）、アクティブ、到着（arrived））に関連付けられる。

【００４４】

この実施の形態では、変数ｎが、使用されるペグの個数を示すために使用される。必要とされる記憶量は、１つのペグにつき必要とされる量のｎ倍に、プロトコルの実行に必要とされる量を加えたものである。

【００４５】

上述したタイプのセットアップ段階を実行した後、この実施の形態の計算プロトコルは、v_1から開始し、最小のバジェットおよび最小数のペグを使用して、要素ごとに、ハッシュチェーンＨを構成する値の列を生成する。プロトコルは、必要とされるバジェット

【数５】

$$b = \lfloor \sigma/2 \rfloor$$

を有し、

【数６】

$$n = \sigma + \lceil log_2(\sigma + 1) \rceil$$

個のペグを使用する。ここで、上述したように、$s = 2^\sigma$は、ハッシュチェーンＨの要素数である。

【００４６】

ペグが、現在の出力に対応する位置に置かれている場合には、このペグは、「到達された」と言われ、この時に、そのペグはフリー状態に入る。フリー状態にあるすべてのペグには、セットアップされてプロトコルの完全性を保証するガイドラインに従って、新しい位置、目的地、状態、および優先度が割り当てられる。これらのガイドラインについては

後述する。

【００４７】

各ステップでは、適切なハッシュチェーン値、すなわち現在のハッシュチェーン値が、計算されて出力される。次に、残っているあらゆるバジェットが、最も低い位置の値を有するペグ、すなわち始点に最も近いペグから開始して、アクティブな高優先度のペグに割り当てられる。まだ残っているあらゆるバジェットは、アクティブな低優先度のペグに割り当てられる。これにより、計算結果が、必要な時に利用可能であることが保証される。

【００４８】

いくつかのペグが同じ優先レベルを有する場合には、現在の位置に対して最小の距離を有するペグが選択される。この方法で、距離に基づいて異なる優先度を割り当てることができるが、簡単にするために、２つのレベルが使用され、所与のレベル内の優先順位付けは、現在の位置に対する距離に基づいて行われる。

【００４９】

高優先度のペグは、３つ以上の区間をカウントする場合にのみ、現在の値の後、最初の区間の「遠端」から開始される。高優先度のペグは、アクティブな低優先度のペグよりも低い位置においては、アクティブな状態のみが許容され、それ以外は、フリー状態に維持される。これは、高優先度のペグが、利用可能なバジェッドのあまりにも多くを取り過ぎないことを確実にするためである。すなわち、これは、高優先度のペグが、それらの目前のタスクを完了する場合に、高優先度のペグを、低優先度のペグの利益にまで遅らせる。

【００５０】

低優先度のペグは、次のペグが到達される前に完了できないような位置、すなわちそれら低優先度のペグの目的地に到着できないような位置から開始される。このように、１つのアクティブな低優先度のペグは、あらゆる残りの計算単位を取り込むのに十分であるので、一時に１つのアクティブな低優先度のペグのみが必要とされる。このアクティブな低優先度のペグに加えて、１つの「バックアップ」の（レディ状態にある）低優先度のペグが存在することが好ましい。このバックアップの低優先度のペグは、前にアクティブなペグがその目的地に到達するとすぐにアクティブにされる。ペグが到達されると、その優先度は、バックアップの低優先度のペグが現在存在しない場合には低に設定される。それ以外は高優先度のペグにされる。これらの規則により、アクティブなペグがその目的地に到達した直後であっても、残りの計算単位を消費する低優先度のペグが常に存在することが保証される。

【００５１】

この例示の実施の形態の初期化段階は、次の通りである。上記に示したように、終点ν_sが$\{0, 1\}^k$から一様にランダムに選択される。ここでｋは、この実施の形態では、１６０に設定されるものと仮定されるが、他の値を使用することもできる。列$H = \{\nu_1, \cdots, \nu_i, \cdots, \nu_s\}$は、ハッシュ関数ｈの繰り返し適用により計算される。ここで、$\nu_i = h(\nu_{i+1})$であり、$1 \leqq i < s$である。

【００５２】

$\sigma = \log_2 s$に対して$1 \leqq j \leqq \sigma$のペグp_jは、次のように初期化される。

ｐｏｓｉｔｉｏｎ（位置）を2^jに設定する。

ｄｅｓｔｉｎａｔｉｏｎ（目的地）を2^jに設定する。

ｖａｌｕｅ（値）をν_{2^j}に設定する。

ｓｔａｔｕｓ（ステータス）を到着（arrived）に設定する。

【００５３】

残りの$\sigma < j \leqq n$のペグp_jは、それらのステータスがフリーに設定される。すべてのペグの情報は、所望の出力列を生成するデバイス上に記憶される。このデバイスは、範囲または長さｓと共に、０に設定されたｃｕｒｒｅｎｔ（現在）および０に設定されたｂａｃｋｕｐ（バックアップ）の一対のカウンタも記憶する。初期化段階の終わりに、対（ｓｔａｒｔｐｏｉｎｔ（始点），ｃｕｒｒｅｎｔ）＝（ν_i，０）が出力され得る。

【００５４】

以下では、１≦ｊ≦ｎのペグｐ$_j$は、最も低い目的地の値が先頭にされて、それらペグの目的地によりソートされた状態で保持され、目的地を割り当てられていないペグは、ソートされたリストの最後に現れるものと仮定する。その結果、（現在の位置から）到達される次のペグは、常にｐ$_1$となる。ペグのステータスが任意の点において変更されると、その変更された項目は、このソートされたリストの適切な箇所に挿入される。表記ＬＰ（低優先度（low priority）の簡略表記）は、アクティブな低優先度のペグの別名として使用される。したがって、ＬＰ．ｐｏｓｉｔｉｏｎは、これがどのペグの番号に対応するかとは関係なく、アクティブな低優先度のペグの現在位置である。同様に、ＢＵは、バックアップの低優先度のペグをいい、ＢＵ．ｐｏｓｉｔｉｏｎは、バックアップの低優先度のペグの現在位置である。

【００５５】

以下のプロトコルは、上記に示した方法で、ハッシュチェーンを計算するために実行される。このプロトコルの各繰り返しにより、次のハッシュチェーン値が生成されて出力される。プロトコルは、ｐｌａｃｅＨＰおよびｐｌａｃｅＬＰの２つのルーチンを利用する。これらのルーチンは、それぞれ高優先度のペグと低優先度のペグとに値を割り当てる。これらのルーチンについては後にも記述する。

１．ａｖａｉｌｂｌｅをｂに設定する。　（残りのバジェットを設定する）
２．ｃｕｒｒｅｎｔを１つ増加する。　（ｃｕｒｒｅｎｔは出力値の位置である）
３．ｃｕｒｒｅｎｔが奇数の場合、以下の処理を実行する。　（この位置にペグは存在しない）
　　ｈ（ｐ$_1$．ｖａｌｕｅ）を出力する。　（計算および出力する）
　　ａｖａｉｌａｂｌｅを１つ減少させる。
　ｃｕｒｒｅｎｔが奇数でない場合、以下の処理を実行する。　（この位置にペグが存在する）
　　ｐ$_1$．ｖａｌｕｅを出力する。　（値を出力し、ペグの状態をフリーに設定する）
　　ｐ$_1$．ｓｔａｔｕｓをフリーに設定する。
　　ｃｕｒｒｅｎｔがｓに等しい場合には、停止する。　（列の最後の値）
４．すべてのフリーなペグｐ$_j$に対して、以下の処理を実行する。　（フリーなペグを再割り当てする）
　　ｂａｃｋｕｐが０である場合、以下の処理を実行する。　（必要とされる低優先度のｂａｃｋｕｐ）
　　ｐ$_j$．ｐｒｉｏｒｉｔｙを低に設定する。
　　ｐ$_j$．ｓｔａｔｕｓをレディに設定する。
　　ＢＵをｐ$_j$に設定する。
　　ｂａｃｋｕｐを１に設定する。
　　ｂａｃｋｕｐが０でない場合、以下の処理を実行する。
　　ｐｌａｃｅＨＰ（ｐ$_j$）を呼び出す。　（それを高優先度にする）
５．ペグをソートする。
６．ｊを１に設定する。　（最初のペグ）
７．ａｖａｉｌｂｌｅ＞０である間、以下の処理を実行する。
　　ａｖａｉｌａｂｌｅを１つ減少させる。　（残りのバジェットを１つ減少させる）
　　ｐ$_j$．ｐｏｓｉｔｉｏｎを１つ減少させる。　（ペグを移動する）
　　ｐ$_j$．ｖａｌｕｅをｈ（ｐ$_j$．ｖａｌｕｅ）に設定する。　（その値を計算する）
　　ｐ$_j$．ｐｏｓｉｔｉｏｎとｐ$_j$．ｄｅｓｔｉｎａｔｉｏｎが等しい場合、以下の処理を実行する。　（ペグに到着）
　　ｐ$_j$．ｓｔａｔｕｓを到着に設定する。
　　ｐ$_j$．ｐｒｉｏｒｉｔｙが低である場合、以下の処理を実行する。　（低優

先度のペグに到着）
ＬＰをＢＵに設定する。　（ｂａｃｋｕｐは低優先度になる）
ｂａｃｋｕｐを０に設定する。
ｐｌａｃｅＬＰを呼び出す。　（新しい低優先度のペグをアクティブにする）
ペグをソートする。
ｊを１つ増加する。　（次のペグ）
８．ペグをソートする。
９．１に進む。　（次の要素）

【００５６】
次に、ルーチンｐｌａｃｅＬＰについて記述する。まず、ｐｌａｃｅＬＰの呼び出し中、変数に割り当てられる値の列をどのように計算できるかについて記述する。列全体ではなく、１つだけのこのような割り当てがどのように計算されるかについては、後述する。スタック空間の使用を最小にするアプローチについても記述する。ｐｌａｃｅＬＰルーチンの望ましい機能は、低優先度のペグの次の始点を、関連付けられた目的地と共に計算することである。

【００５７】
以下では、説明を簡略化し、さまざまなチェーンの長さに均一性を提供するために、「正規化された」位置が使用される。正規化された位置から実際の位置を得るには、正規化された位置に値λを乗算すればよい。ここで、λは、ｂをバジェットとすると、２ｂより小さくない２の最小の累乗である。したがって、ｂ＝４、λ＝８に対して、（４，８，６，８，１６，１２，１０，１２，１６，１４，１６）から開始する正規化された始点の列は、（３２，６４，４８，６４，１２８，９６，８０，９６，１２８，１１２，１２８）の列に対応する。同様に、目的地の点およびペグの始点とペグの目的地との間の距離は、正規化された表現で記述される。

【００５８】
時間の経過につれて、ペグはチェーンの適切な位置に置かれる。このような位置のそれぞれは、その終点が他のペグおよび／または現在位置に対応する区間内に存在する。上記例示の実施の形態では、所与のペグは、区間を、２つの等しい大きさの区間に分割する。その後、その結果の部分が再び分割される。チェーンの始点の最も近く、すなわち現在位置に最も近く、かつ、この現在位置とチェーンの終点との間に位置する区間を選択することにより、常に最も大きな部分が分割され、等しい大きさの区間の間の関係が崩れる。

【００５９】
区間の最初の分割は、木のルートと関連付けて見ることができる。ルートの子は、最初の分割の結果生じた２つの区間の分割、および、それらのそれぞれの分割による子に対応する。リーフは、区間の最小の分割に対応する。木の各ノードによって、始点、距離、および目的地が関連付けられる。ここで、目的地は、始点と距離との差である。

【００６０】
高さｊの木のルートの始点は、$start = 2^{j+1}$である。木における高さｉのノードの距離は、$dist = 2^{i-1}$である。したがって、ルートの距離は2^{j-1}であり、木のリーフはすべて、正規化された距離ｄｉｓｔ＝１を有する。任意のノード（および特にルート）の目的地ｄｅｓｔは、その始点と距離との差、すなわちｄｅｓｔ＝ｓｔａｒｔ－ｄｉｓｔである。最後に、左の子の始点は、その親の目的地の値ｐａｒｅｎｔ．ｄｅｓｔであるのに対して、右の子については、その親の始点の値ｐａｒｅｎｔ．ｓｔａｒｔである。

【００６１】
深さ優先探索（左の子は、常に右の子より先にトラバースされる）を実行することから得られるｓｔａｒｔおよびｄｅｓｔの割り当てのシーケンスについて考察する。そのシーケンスは、関連付けられた初期区間、すなわち分割前の区間に対応する割り当てのシーケンスである。さらに、このような木からなる「森」をトラバースすることから得られるこ

のような割り当てのシーケンスを考察する。ここで、最初の木は高さ１を有し、各木は、その先行する木よりもレベルが１つだけ高くなっている。そのシーケンスは、正規化された割り当てに必要とされるシーケンスである。

【００６２】

ｐｌａｃｅＬＰの各呼び出しにより、まず、上述した列からのすべてについて、このような正規化された値の対が計算され、次に、これらの値がλと乗算されて、その結果が返される。したがって、各呼び出しでは、次のような設定が行われる。

ＬＰ．ｐｒｉｏｒｉｔｙを低に設定する。

ＬＰ．ｓｔａｔｕｓをアクティブに設定する。

ＬＰ．ｐｏｓｉｔｉｏｎをλ　ｓｔａｒｔに設定する。

ＬＰ．ｄｅｓｔｉｎａｔｉｏｎをλ　ｄｅｓｔに設定する。

【００６３】

λ　ｓｔａｒｔ＞ｓになるとすぐに、低優先度のペグはもはや必要なくなるので、割り当ては実行されない。ｐｌａｃｅＬＰのあらゆる呼び出しは、呼び出し後、割り当てを行うことなくリターンする。

【００６４】

ルーチンｐｌａｃｅＬＰは、その列の第ｉ番目の要素が、アクティブにされる第ｉ番目の低優先度のペグの、ハッシュチェーンＨ上の開始位置および目的地の対に対応する要素の列を生成する。その始点は、ハッシュチェーン上にすでに置かれているペグの始点に対応し、その目的地は、この同じペグと、現在のポインタの方向における最も近いペグとの中点に対応する。始点と目的地との間の距離は、移動するペグに利用可能な、１つの要素あたりのバジェットの少なくとも２倍である。これにより、低優先度のペグは、別のペグがアクティブにされる前に、その目的地に到達しないことが保証される。

【００６５】

高優先度のペグの次の場所を計算するルーチンは、上記と類似しており、その主な相違は、（１）実際の値と正規化された値とが一致すること、および、（２）森の中の木が、高さ$\log_2 \lambda$に到達した後、それらの木が成長を停止することである。

【００６６】

このように、第ｉ番目の木の開始位置は、$i \leqq \log_2 \lambda$についてはｓｔａｒｔ＝2^{i+1}であり、$i > \log_2 \lambda$についてはｓｔａｒｔ＝λ（ｉ－$\log_2 \lambda$）である。すでに述べたように、左の子の始点は、その親の目的地の値ｐａｒｅｎｔ．ｄｅｓｔであるのに対して、右の子については、その親の始点の値ｐａｒｅｎｔ．ｓｔａｒｔである。木における高さｉのノードの距離は、ｄｉｓｔ＝2^{i-1}である。すでに述べたように、目的地は、その始点と距離との差、すなわちｄｅｓｔ＝ｓｔａｒｔ－ｄｉｓｔである。

【００６７】

どの割り当てが行われる前にも、ｓｔａｒｔ≦ＬＰ．ｐｏｓｉｔｉｏｎ、すなわち、当該割り当てが、アクティブな低優先度のペグの前方の位置に対するものであることが検証される。これが、そうではない場合には、この点に対する割り当ては行われず、この比較は、ルーチンの次の呼び出しで再度実行される。そうでない場合には、このルーチンで、以下の割り当てがペグp_jに対して行われる。

p_j．ｐｒｉｏｒｉｔｙを高に設定する。

p_j．ｓｔａｔｕｓをアクティブに設定する。

p_j．ｐｏｓｉｔｉｏｎをｓｔａｒｔに設定する。

p_j．ｄｅｓｔｉｎａｔｉｏｎをｄｅｓｔに設定する。

【００６８】

次に、ルーチンｐｌａｃｅＨＰについて記述する。このルーチンは、その列の第ｉ番目の要素が、アクティブにされる第ｉ番目の高優先度のペグの対（ｓｔａｒｔ，ｄｅｓｔ）に対応する要素の列を生成する。その始点ｓｔａｒｔは、ハッシュチェーン上にすでに置かれているペグの始点に対応し、目的地ｄｅｓｔは、この同じペグと、現在のポインタの方向における最も近いペグとの間の中点に対応する。始点は、現在のポインタとアクティ

ブな低優先度のペグとの間の点であって、可能な限り現在のポインタに近い点として選択される。その結果、始点と目的地との間の距離は、少なくとも２となる。

【００６９】

次に、上述したｐｌａｃｅＬＰルーチンおよびｐｌａｃｅＨＰルーチンの記憶の複雑度に関連したいくつかの問題を検討する。当業者に容易に理解されるように、上述した処理を容易にするには、メモリスタックを使用することができる。あるいは、作業空間を保存するために、スタックを使用せずに、必要なときに最初から状態を再計算する解決法を選ぶこともできる。ある変数は木の高さを記憶し、別の変数はルートからの（深さ優先探索を使用した）ステップ数を記憶する。さらに、変数は、ｓｔａｒｔ、ｄｉｓｔ、およびｄｅｓｔ用に必要とされる。これらの値を計算するために、処理を行う者は、（それぞれの時に）それらの始点の割り当てから開始して、次に、必要とされるルートからのステップ数を導き出す木のトラバースに従ってそれらの割り当てを変更する。これは、上記のように、それぞれの割り当てに従って行われる。これらの操作のいずれも、いかなるハッシュ関数の評価も必要としないことに留意すべきである。

【００７０】

ｐｌａｃｅＬＰの最大の木の高さは、$2^{\sigma-\lambda}$の正規化された始点および2^{σ}の始点に対応するので、この高さはσ－$\log_2\lambda-1$となる。したがって、この変数は、

【数７】

$$\lfloor \log_2 \sigma \rfloor$$

ビットを必要とする。ｐｌａｃｅＨＰでは、最大の高さは$\log_2\lambda$となり、

【数８】

$$\lceil \log_2 \lceil \lambda \rceil \rceil$$

ビットの記憶装置を必要とする。最大の高さの木は、ｐｌａｃｅＬＰについては$2^{\sigma-\log 2\lambda-1}-1$個のノードを有し、ｐｌａｃｅＨＰについては$2^{\lambda}-1$個のノードを有する。したがって、そのルートからの距離は、σ－$\log_2\lambda-1$で表すことができ、それぞれλビットで表すことができる。最後に、最後の３つの変数に割り当てることができる最大値は2^{σ}であるので、それら３つの変数の最大値は、それらのそれぞれ１つにつきσビットである。これらの変数は、計算における状態のみを保持するので、各ルーチンにつき一組を有する必要はない。

【００７１】

したがって、ｐｌａｃｅＬＰおよびｐｌａｃｅＨＰの記憶の必要量は、４σ＋$\log_2$σ＋λ－１ビット未満である。これは、ペグの記憶の必要量に比べると小さく見える。ペグは、値を記憶するのに１６０ビットを必要とし、残りの状態情報、例えばカウンタなどに（チェーンの長さに応じて）ある少ないビット数を必要とする。特に、σビットは、位置および目的地のそれぞれに必要とされる。この実施の形態では、全体で約

【数９】

$$(160+2\sigma)(\sigma+\lceil \log_2(\sigma+1)\rceil)$$

ビットが必要とされる。

【００７２】

長さｓ＝2^{σ}、バジェット

【数１０】

$$b=\lfloor \sigma/2 \rfloor$$

および

【数１１】

$$n=\sigma+\lceil \log_2(\sigma+1)\rceil$$

のｎ個のペグの一方向チェーンについて検討する。セットアップ段階から特定の要素までのバジェットの合計は、当該要素における累積バジェットと呼ばれる。

【００７３】

より具体的な数値の例として、σ＝３１、ｓ＝2^{31}＝２．１４７×10^{9}である場合に、バジェットｂは１５になり、これは、各チェーン値を計算するのに、最大１５回のハッシュ関数の適用が必要となることを示す。この例では、ｎ＝３６個のペグが必要とされる。これらのペグのそれぞれは、周知のＳＨＡ－１ハッシュ関数の使用を仮定すると、ヘルパ値を記憶するための２０バイトに加えて、状態情報の記憶用に追加の８バイトを使用して実施することができる。なお、ＳＨＡ－１ハッシュ関数は、FIPS PUB 180-1、「Secure Hash Standard, SHA-1」、www.itl.nist.gov/fipspubs/fip180-1.htmに記述されており、これは、参照により本明細書に援用される。この結果、全体で３６×（２０＋８）＝１００８バイトの記憶装置が必要とされる。この一方向チェーンが、毎秒１つのチェーン値の出力を必要とするアプリケーションで実施されると、そのチェーンは、６８年より長く持ちこたえることになる。

【００７４】

上記に示したように、この例示の実施の形態における一方向チェーンの計算プロトコルは、完全性の性質を示す。ｂのバジェット制約およびｎの記憶制約を有するプロトコルがハッシュ列の第ｊ番目の値v_jを出力する場合に限り、かつ、そのプロトコルが、要素ｊ－１で成功した場合に、そのプロトコルは要素ｊで成功すると言うことができる。そのプロトコルは、要素１では自動的に成功すると言われる。要素１は、セットアップ段階に対応し、例示の実施の形態では、セットアップ段階において、計算および記憶の点に関して厳しい制約が加えられていない。

【００７５】

プロトコルが、チェーンの範囲または長さｓ＝２σ、バジェット

【数１２】

$$b=\lfloor \sigma/2 \rfloor$$

および

【数１３】

$$n=\sigma+\lceil \log_2(\sigma+1) \rceil$$

個のペグについて、２≦ｊ≦ｓの要素ｊで成功することを示すことができる。

【００７６】

上述したプロトコルは、本発明の現在の好ましい実施の形態である。このプロトコルの１つの可能な変形の例を以下に記述する。当業者は、他の数多くの変形も可能であることを認識するであろう。

Ａ．開始

１．v_sを選択し、ｔｅｍｐをv_sに設定する。

２．上述した方法でv_sから$v_1 \cdots v_{s-1}$を計算する。

３．カウンタｋをｓに設定し、カウンタｍをｎに設定する。

４．j_mをｋとし、ＦＩＸ（j_m）をv_kとする。

５．ｋをｔｒｕｎｃ（ｋ／２）に設定し（すなわち、２分して端数を切り捨てる）、ｍをｍ－１に設定する。

６．ｋ回繰り返す。ｔｅｍｐをｈ（ｔｅｍｐ）に設定する。

７．ｋ＞１の場合には、４に進む。

８．ｃｕｒｒｅｎｔを１に設定する。

【００７７】

上記開始プロセスは、場所ｓ，ｓ／２，ｓ／４，…２にペグを配置する。したがって、一般に、この開始プロセスは、好ましい実施の形態と共に記述した図２Ａの初期ペグセットアップに対応する。上記に示したように、ペグは所与のデバイス用に計算されて記憶さ

れる。この計算は、補助デバイスによって実行されてもよいが、必ずしもそうする必要はない。所与のデバイスが、上述した方法で初期化されるとすぐに、一方向チェーンの計算プロセスが実施される。このプロセスは次のようになる。

Ｂ．演算

１．ｃｕｒｒｅｎｔを１つ増加する。ａｖａｉｌａｂｌｅをｂに設定する。

２．p_j．ｃｕｒｒｅｎｔとp_j．ｐｏｓｉｔｉｏｎとが等しい場合には、p_j．ｖａｌｕｅを出力する。次に、この値p_jを、ｐｌａｃｅＨＰルーチンまたはｐｌａｃｅＬＰルーチンのうち、必要とされるいずれか一方に従って再配置する。

３．そうでない場合には、最小のp_j．ｐｏｓｉｔｉｏｎを有するp_jについて、ｔｅｍｐをp_jに設定する（好ましい実施の形態では、ソートによりこのp_jは常にp_1である）。次に、以下のものを、ｔｅｍｐ．ｐｏｓｉｔｉｏｎがｃｕｒｒｅｎｔと等しくなるまで繰り返す。

ａｖａｉｌａｂｌｅを１つ減少させる。

ｔｅｍｐ．ｐｏｓｉｔｉｏｎを１つ減少させる。

ｔｅｍｐ．ｖａｌｕｅをｈ（ｔｅｍｐ．ｖａｌｕｅ）に設定する。

４．移動させる次のペグを選択する。好ましい実施の形態では、すでに２つの優先度が存在する。ここで優先度は、（１）最も近い値が次の回にあるべき場所にない場合（すなわちp_j．ｄｅｓｔｉｎａｔｉｏｎとp_j．ｐｏｓｉｔｉｏｎとの差がｂより大きい場合）には当該最も近い値に与えられ、そうでない場合には（２）最も大きなｐｏｓｉｔｉｏｎを有する値に与えられる。これは、優先度の割り当ての最適な戦略ではないが、多くのチェーンで受け入れることができることに留意すべきである。

５．選択されたペグp_jを次のように移動させる。

p_j．ｐｏｓｉｔｉｏｎを１つ減少させる。

p_j．ｖａｌｕｅをｈ（p_j．ｐｏｓｉｔｉｏｎ）に設定する。

６．p_j．ｐｏｓｉｔｉｏｎとp_j．ｄｅｓｔｉｎａｔｉｏｎとが等しい場合には、p_j．ｓｔａｔｕｓを到着に設定する。

７．ａｖａｉｌａｂｌｅを１つ減少させ、ａｖａｉｌａｂｌｅ＞１である場合には、４に進む。

【００７８】

上記計算プロセスは、好ましい実施の形態の計算プロセスのステップ１～８に対応し、生成される各チェーン出力値に対して繰り返される。上記に示したように、例示のプロトコルの他の数多くの変形を使用して、本発明の技法を実施することができる。

【００７９】

上述した例示の一方向チェーンの計算プロトコルは、従来の技法を上回る大きな利点を提供する。より詳細には、上記プロトコルは、従来の技法を使用している場合のＯ（ｓ）ではなく、チェーンの長さの対数の２乗のオーダ、すなわちＯ（（ｌｏｇ　ｓ）^{2}）の記憶計算積を有する。本発明の一方向チェーンの計算は、例えばデジタル署名、メッセージ認証、ユーザおよびデバイスの認証、ならびにマイクロペイメントといった数多くの暗号アプリケーションでの使用に適している。そして、その記憶および計算の効率の良さにより、これらおよび他の暗号アプリケーションを軽量デバイス上で実施することが可能となる。

【００８０】

一例として、公に検証可能な署名を生成するのに、本発明でなければ一方向チェーン計算に基づく従来の技法が使用されることになるが、本発明によれば、この本発明を使用して、特に効率的な方法で生成することができる。その結果、署名は、従来のMerkleの署名およびLamportの署名よりもはるかに効率的に生成される。Merkleの署名およびLamportの署名は、一般に、従来の署名の中で最も効率的なものに含まれると考えられており、L. Lamport著の「Constructing Digital Signatures from a One Way Function」、SRI International Technical Report CSL-98 (Octorber 1979)、およびR. Merkle著の「A digital signature based on a conventional encryption function」、Proceedings of Crypto

'87にそれぞれ記述されている。これらの文献の双方とも、参照により本明細書に援用される。

【００８１】

より詳細には、本発明のデジタル署名のアプリケーションは、上記引用したA. Perrig等の参考文献に記述されたものと類似の技法を利用することができる。この場合、受信側の関係者は、メッセージ認証コード（ＭＡＣ）が別の関係者から受信されるとすぐに、従来のタイムスタンプ技法を使用して、そのＭＡＣにタイムスタンプを付け、その後、対応する鍵が受信されるとすぐに、ＭＡＣを検証する。ここで、この鍵は、ハッシュチェーンにより計算された値である。その後、第３者は、鍵およびタイムスタンプが、メッセージおよびＭＡＣと共に与えられると、ＭＡＣの正確さと、鍵が公開された時刻より前にＭＡＣが受信された事実とを検証することができる。これは、バインドしたデジタル署名（binding digital signature）を構成する。

【００８２】

また、本発明は、安全なログインアプリケーション、例えば、パスワードは１度しか使用されないので、そのパスワードがログインセッション中またはログインセッション後に危険にさらされるかどうかは問題にならないアプリケーションにも使用することができる。特に、パスワードとして、v_1から開始するハッシュチェーンの連続した値を使用することができる。

【００８３】

図４のシステムを参照して、このような安全なログインプロセスは、次のようにして実施することができる。クライアント１１０－１が、パスワードを使用してサーバ１１４の組にログインしたいものと仮定する。このパスワードは、ネットワーク１１２上を、暗号化されるか、または暗号化されないで送信される。クライアント１１０－１は、攻撃者が、送信された所与のパスワードを再利用することを防止したい。したがって、そのパスワードは、本明細書で記述した方法で、ハッシュチェーン上の値として生成される。その結果、チェーン値v_1が最初のパスワードとして送信され、２番目のパスワードとしてv_2が送信され、以下同様である。別のクライアント１１０またはサーバ１１４のうちの特定の１つであり得る別の関係者は、クライアント１１０－１を、サーバ１１４により登録し、次に、クライアント１１０－１による最初のログインの試みが行われる前に、クライアント１１０－１にハッシュチェーンの終点値v_sを、任意の付加的な登録情報を加えて送信する。所与のログインの試みにおいて、クライアント１１０－１は、識別情報、登録情報、適切なチェーン値v_1、v_2など、および多くのログインの試みがどのように実行されたかを識別するカウンタ、または、そうでなければどのチェーン値が送信されたかを指定するカウンタをサーバ１１４に送信する。ログインの試みが、本来的にタイムベースである場合には、そのカウンタは、暗黙的なものとすることができる。この特定の適用のさらに詳細な内容については、L. Lamport著の「Password authentication with insecure communication」、Communications of the ACM, 24(11):770-772, November 1981を参照されたい。この文献は、参照により本明細書に援用される。

【００８４】

本発明は一般に、一方向チェーンの計算を利用するあらゆるアプリケーションに役立つ。本発明が一方向チェーンの計算プロセスの記憶および計算の効率を十分に改善するのと同程度に、本発明は、一方向チェーンの計算を必要とするあらゆるアプリケーションの効率を改善する。本発明は、１つまたは２つ以上の一方向チェーン値を計算できるあらゆるタイプの処理デバイスに実施することができる。

【００８５】

上記に示したように、本発明は、木の形状または他のタイプのグラフの形状の要素の配置に適用することができる。このような配置は、上記で与えた「チェーン」の一般的な定義の範囲内に含まれる。次に、本発明の技法の木構造またはグラフ構造への適用を説明する２つの例を提供する。これは、アップデート戦略をこの適用および特定の木トポロジーまたはグラフトポロジーに適するように変更する問題にすぎず、それ以外の原理は、上述

したものと同じ状態で維持されることに留意されたい。

【００８６】

例１：木構造。このような構造では、終点の値v_sは、木のルートノードに対応する。表記を簡単にするために、このノードをＲと呼ぶことにする。Ｖ１およびＶ２をＲの２つの子とし、Ｖ１１およびＶ１２をＶ１の２つの子とする。同様にして、木のすべてのノードに名前を付けることができる。本発明が上記のような２分木だけでなく、任意のファンアウトの頂点を有する木にも関することは、当業者に明らかであろう。ｓを木の高さとする。ルートに関連付けられた値はＲ．ｖａｌであり、ランダムに選択されてもよい。Ｖｘ．ｖａｌをノードＶｘに関連付けられた値とする。ここで、ｘは例えば「１１１２」などの特定のパスを示す。次に、Ｖｘの第１の子の値はＶｘ１．ｖａｌ＝ｈ（Ｖｘ．ｖａｌ，１）であり、Ｖｘの第２の子の値はＶｘ２．ｖａｌ＝ｈ（Ｖｘ．ｖａｌ，２）である。当業者によって理解されるように、他の割り当てが同様に選択されてもよい。これから、ノードＶｘ１またはＶｘ２に関連付けられた値を与えると、その親Ｖｘに関連付けられた値を決定することは、その祖先の１つに関連付けられた値の知識がなければ実行不可能であることが分かる。ここで、木のリーフのある部分を既知であるとし、ペグを木のいくつかの頂点に配置する。例えば、１つのペグが、位置Ｖ１に配置されると仮定する。これにより、Ｖ１を祖先とするノードのすべての高速化した計算が可能となる。木の他方の半分にあるノードの値を計算するには、Ｒに上って行かなければならない。その代わりに、ペグを「オールワン」パス上において、ルートからリーフへの中間に配置してもよい。これにより、木のより小さな部分ではあるが、より速い速度で、その部分の高速化した計算が可能となる。第２のペグは、木の別の部分に配置することができる。例えば、第２のペグは「オールワン」パス上において、第１のペグとリーフとの間の半分の距離に配置することができる。これにより、この第２のペグより下のノードの計算がさらに高速化され、第１のペグより下であるが、第２のペグより下ではないパス上の他のノードを多少高速に計算することが可能になる。さらに別のノードをさらに計算することができるが、ルートからパスを完全にたどる必要がある。次に、ノードが計算されるとすぐに、次のアクセスの確率分布に従ってペグを再配置することができる。したがって、木のある部分のアクセスが、木の第２の部分のアクセスの可能性をより高くする場合には、１つまたは２つ以上のペグをこの部分に再配置することができる。これは、上述した例示の実施の形態について言えば、指定された優先順位付けに従って任意の利用可能なバジェットを使い、祖先の値からなる連続した一方向イメージとして新しい値を計算することにより行われる。

【００８７】

例２：２つの結合したチェーン。２つの結合したチェーンが存在する状況について考察する。各チェーンは、それ自身の終点を有し、チェーン値のうちのあるわずかな一部は、それらの対応するチェーン値の関数であるだけでなく、他方のチェーン値の関数でもある。異なるチェーンは異なる一方向関数を使用してもよく、入力の異なる部分で動作してもよいことに留意されたい。したがって、この状況は、非循環有向グラフ（ＤＡＧ（directed acyclic graph））の特定の形状を記述し、他のこのような例を構成できることが当業者によって理解されるであろう。単純な場合には、一方のチェーンの終点から距離ｉにあるそのチェーンのすべての要素を、その祖先（そのチェーンの要素ｉ－１）と、他方のチェーンの終点からその距離にある他方の要素（他方のチェーンの要素ｉ－１）との双方の関数とすることができる。これらのチェーンのそれぞれの長さをｓとし、ペグが、チェーンのうちの一方の始点から距離ｓ／２、ｓ／４、ｓ／８などに置かれ、他方のチェーンにはペグが置かれていないと仮定する。これらのチェーンは連結されているので、一方のチェーンのすべての値の効率的な計算を可能にすることにより、他方のチェーンのすべての値も計算することができる。当業者によって理解されるように、任意の個数のチェーンをこのようにして連結することができ、本発明の技法を利用して、出力値の効率的な計算を可能にすることができる。

【００８８】

本明細書で記述された例示的な一方向チェーンの計算技法は、本発明の動作を説明する

ことを意図したものであり、したがって、本発明を任意の特定の実施の形態または複数の実施の形態の群に限定するものと解釈されるべきではないことは、再度強調されるべきである。例えば、特定のタイプの一方向チェーンを使用して説明したが、上記に示したように、本発明は他のタイプの一方向関数に適用でき、木や他のタイプのグラフなどの他の形状の要素の配置に適用できる。さらに、使用された特定の個数のヘルパ値、それらの初期分布、ならびに、対応するペグを計算および所与のチェーン値の出力に基づいて再配置する方法は、代替的な実施の形態では変更することができる。添付の特許請求の範囲の範囲内のこれらおよび他の数多くの代替的な実施の形態は、当業者に明らかであろう。

【図面の簡単な説明】

【００８９】

【図１】連続的な出力値が本発明の技法を利用した効率的な方法で計算され得る一例の一方向チェーンを示す図である。

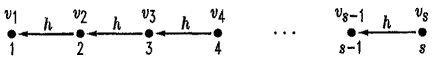
【図２Ａ】本発明の例示の実施の形態の記憶されるヘルパ値の初期セットアップの例を示す図である。

【図２Ｂ】本発明による一例の一方向チェーンの計算プロセスの流れ図である。

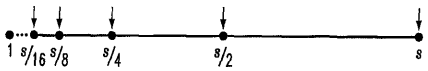
【図３】本発明が全体的にまたは部分的に実施され得る例示の処理デバイスの簡略したブロック図である。

【図４】本発明の一例のネットワークベースの実施の形態を示す図である。

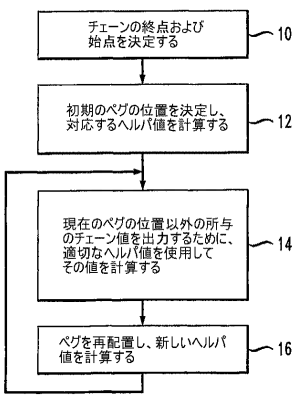
【図１】



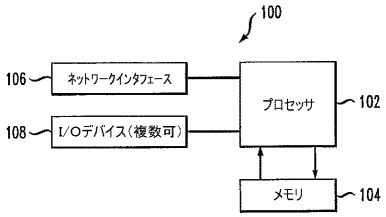
【図２Ａ】



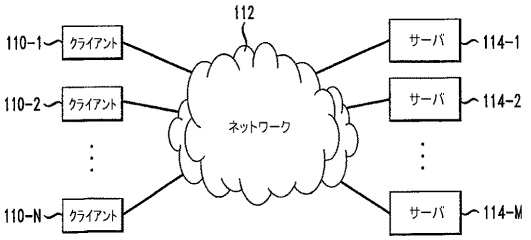
【図２Ｂ】



【図３】



【図４】



フロントページの続き

(74)代理人 100101498
弁理士 越智 隆夫
(74)代理人 100096688
弁理士 本宮 照久
(74)代理人 100102808
弁理士 高梨 憲通
(74)代理人 100104352
弁理士 朝日 伸光
(74)代理人 100107401
弁理士 高橋 誠一郎
(74)代理人 100106183
弁理士 吉澤 弘司
(74)代理人 100120064
弁理士 松井 孝夫
(72)発明者 ヤコブソン，ビヨルン，マーカス
アメリカ合衆国 ０７０３０ ニュージャージイ，ホボケン，ガーデン ストリート １２０３

審査官 石田 信行

(56)参考文献 米国特許第０５４３４９１９（ＵＳ，Ａ）
米国特許第０６０９７８１１（ＵＳ，Ａ）
特開２０００－２５９６１１（ＪＰ，Ａ）
特表２００５－５２５７３１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G09C 1/00