

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2007-57811

(P2007-57811A)

(43) 公開日 平成19年3月8日(2007.3.8)

(51) Int. Cl. F I テーマコード (参考)
G09C 1/00 (2006.01) G09C 1/00 650A 5J104

審査請求 未請求 請求項の数 5 O L (全 14 頁)

<p>(21) 出願番号 特願2005-242956 (P2005-242956)</p> <p>(22) 出願日 平成17年8月24日 (2005.8.24)</p> <p>特許法第30条第1項適用申請有り 2005年3月3日 社団法人電子情報通信学会発行の「電子情報通信学会技術研究報告 信学技報Vol. 104 No. 708」に発表</p>	<p>(71) 出願人 504139662 国立大学法人名古屋大学 愛知県名古屋市千種区不老町1番</p> <p>(74) 代理人 100082500 弁理士 足立 勉</p> <p>(72) 発明者 ▲高▼木 直史 名古屋市千種区不老町1番 国立大学法人名古屋大学内</p> <p>(72) 発明者 カイハラ、マルセロ エミリオ 名古屋市千種区不老町1番 国立大学法人名古屋大学内</p> <p>Fターム(参考) 5J104 AA18 AA22 JA28 NA02 NA18 NA39</p>
--	--

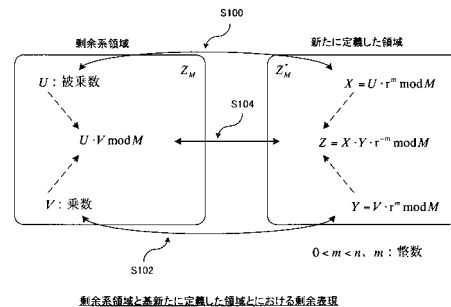
(54) 【発明の名称】 剰余系の計算方法及び装置並びにプログラム

(57) 【要約】

【課題】 剰余系における乗算剰余算の繰返し計算を高速化するための計算方法を提供する。

【解決手段】 剰余系における変数 U , V を新たに定義する領域において、 $X = U \cdot R \text{ mod } M$ 、 $Y = V \cdot R \text{ mod } M$ に変換し、乗算剰余算 $U \cdot V \text{ mod } M$ を $X \cdot Y \cdot R^{-1} \text{ mod } M = U \cdot V \cdot R \text{ mod } M$ に置き換える。ここで、 $R = r^m$ (ただし、 $0 < m < n$ 、 m : 整数) とすると、乗数 U は $X = U \cdot r^m \text{ mod } M$ に、被乗数 V は $Y = V \cdot r^m \text{ mod } M$ に変換される (S101, S102)。つまり、 U , V が新たに定義する領域の X , Y に変換される。そして、乗算剰余算は、 $U \cdot V \text{ mod } M$ から、新たに定義する領域における $X \cdot Y \cdot r^{-m} \text{ mod } M$ に置換される (S104)。このようにして、パラメータ m を導入することにより、乗数 Y を上位部分 Y_H と下位部分 Y_L との二つの部分に分割し、これらを並列に処理することが可能となる。

【選択図】 図1



【特許請求の範囲】

【請求項 1】

整数 M を法とする剰余系において、変数 U 、 V を前記法 M と互いに素で M より小さな定数 R を用いて、変数 $X = U \cdot R \bmod M$ 、変数 $Y = V \cdot R \bmod M$ 、
に変換し、

剰余系における乗算剰余算、 $U \cdot V \bmod M$ を演算

$$X \cdot Y \cdot R^{-1} \bmod M \cdots \text{式 1}$$

に置換し、

前記剰余系における計算と同じ計算を行い、

その計算結果 Z を演算

$$Z \cdot R^{-1} \bmod M \cdots \text{式 2}$$

にて逆変換して剰余系における計算結果を得ることを特徴とする剰余系の計算方法。

10

【請求項 2】

請求項 1 に記載の剰余系の計算方法において、

前記変数 Y が r 進で n 桁であるとき、前記定数 R を、

$$R = r^m \quad (m \text{ は、} m < n \text{ を満たす整数})$$

とし、

前記変数 Y を

$$Y = Y_H \cdot r^m + Y_L \cdots \text{式 3}$$

によって、上位 $(n - m)$ 桁の Y_H と下位 m 桁の Y_L とに分割し、

20

前記式 1 を、

$$(X \cdot Y_H \bmod M + X \cdot Y_L \cdot r^{-m} \bmod M) \bmod M \cdots \text{式 4}$$

に変換して、

前記式 4 の

$$X \cdot Y_H \bmod M \cdots \text{式 4 a}$$

と、

$$X \cdot Y_L \cdot r^{-m} \bmod M \cdots \text{式 4 b}$$

と、

を並列処理で実行することを特徴とする剰余系の計算方法。

【請求項 3】

30

r 進の整数 M を法とする剰余系において (ただし、 M と r とは互いに素)、前記法 M 、 r 進で n 桁の変数 Y 及び変数 X が入力されたときに前記変数 Y を上位 $(n - m)$ 桁の Y_H 及び下位 m 桁の Y_L に分割する分割手段と、

前記変数 Y を分割した上位 $(n - m)$ 桁の Y_H 、前記変数 X 及び前記法 M で

$$X \cdot Y_H \bmod M$$

を計算して出力する第 1 乗算剰余算器と、

前記変数 Y を分割した下位 m 桁の Y_L 、前記変数 X 及び前記法 M で

$$X \cdot Y_L \cdot r^{-m} \bmod M$$

を計算して出力する第 2 乗算剰余算器と、

前記第 1 乗算剰余算器の出力及び前記第 2 乗算剰余算器の出力を加算し、その加算結果
を出力する加算器と、

40

を備えたことを特徴とする乗算剰余演算装置。

【請求項 4】

請求項 3 に記載の乗算剰余演算装置において、

前記加算器の加算結果を入力し、前記法 M による剰余算を行って出力する剰余算器を備えたことを特徴とする乗算剰余演算装置。

【請求項 5】

請求項 1 又は請求項 2 に記載の剰余系の計算方法をコンピュータに実行させるためのプログラム。

【発明の詳細な説明】

50

【技術分野】

【0001】

本発明は、剰余系の計算方法及び装置並びにプログラムに関する。

【背景技術】

【0002】

従来、ネットワーク上で送受信されるデータのセキュリティを確保するために、データを暗号化・復号化するRSA (Rivest-Shamir-Adleman)等の公開鍵暗号システムが用いられている。

【0003】

その公開鍵暗号システムでは、使用される公開鍵のサイズ(桁数)を大きくすれば、不正な方法による暗号の解読が困難となり、情報を送受信する際のセキュリティをより強化することができる。

【0004】

ところが、RSA等の公開鍵暗号システムでは、暗号化と復号化のために剰余系指数演算を行う必要があり、その剰余系指数演算の量は公開鍵のサイズにともなって多くなる。

そして、ほとんどの場合、その剰余系指数演算は非常に大きな整数の乗算剰余算(剰余系乗算)を繰返し計算することにより実現されており、その乗算剰余算の繰返し計算には膨大な時間が必要であった。

【0005】

したがって、この乗算剰余算の繰返し計算の実行速度を速くすることができれば、使用する鍵のサイズをさらに大きくでき、ネットワーク上で送受信されるデータのセキュリティ強化を図ることができる。

【0006】

乗算剰余算の繰返し計算を高速化する方法として、変数をモンゴメリの領域に変換して個々の乗算剰余算をモンゴメリ乗算に置き換える方法があり(例えば、非特許文献1参照)、さらに、そのモンゴメリ乗算の高速化法として基数を増す方法がある(例えば、非特許文献2参照)。

【0007】

また、個々の乗算剰余算の高速化の方法としてインターリーブ法で基数を増す方法がある(例えば、非特許文献3参照)。

【非特許文献1】P.L. Montgomery, "Modular Multiplication without Trial Division," Mathematics of Computation, vol.44, no.170, pp.519-521, Apr.1985.

【非特許文献2】S.E Eldridge and C.D. Walter, "Hardware Implementation of Montgomery's Modular Multiplication Algorithm," IEEE Transactions on Computers, vol.42, no.6, pp.693-699, Jun.1993.

【非特許文献3】N. Takagi, "A Radix-4 Modular Multiplication Hardware Algorithm for Modular Exponentiation," IEEE Transactions on Computers, vol.41, no.8, pp.949-956, Aug. 1992.

【発明の開示】

【発明が解決しようとする課題】

【0008】

ところで、上記非特許文献2及び非特許文献3に記載されている改良された乗算剰余算方法は、演算の基数を増すことにより、演算に必要なクロックサイクル数を削減するものである。

【0009】

しかし、基数が増すと演算回路が複雑になり、演算のサイクル時間が大きくなる。従って、基数が増すにつれ、高速化の割合は小さくなるという問題がある。

本発明は、こうした問題に鑑みなされたもので、剰余系における乗算剰余算の繰返し計算をサイクル時間を増すことなく高速化するための計算方法を提供することを目的とする。

10

20

30

40

50

【課題を解決するための手段】

【0010】

本発明の理解をより明確にするために、特許請求の範囲に記載した課題解決手段を具体的に解説する前に、本発明の技術的思想の創作過程について説明する。

前述したように、RSA等の公開鍵暗号システムでは、暗号化と復号化のために非常に大きな整数の乗算剰余算（剰余系乗算）を繰返し計算することにより実現されている。

【0011】

このような、大きな整数に関する繰返し計算においては、その大きな整数を上位部分と下位部分とに分割して、上位部分の計算と下位部分の計算とを並列処理して高速化することが考えられる。

【0012】

ところが、乗算剰余算は、被乗数に乗数を掛け、その掛け算の結果を法Mで割り、そのときの余りを得るという演算であるため、乗数を分割して並列処理をするメリットが得られない。

【0013】

これを分かりやすく説明するために、以下に具体的な例により説明する。

例えば、法 $M = 9753$ とする乗算剰余算、 $5432 \times 4321 \pmod{9753}$ を考える。

【0014】

$5432 \times 4321 \pmod{9753}$ をそのまま計算すると、

$$5432 \times 4321 \pmod{9753}$$

$$= 23471672 \pmod{9753} = 5954$$

となり、8桁を4桁で割る割り算を行う必要がある。

【0015】

次に、乗数の4321を上位2桁の43と下位2桁の21とに分割して計算すると、

$$5432 \times 4321 \pmod{9753}$$

$$= (5432 \times 4300 \pmod{9753}$$

$$+ 5432 \times 21 \pmod{9753}) \pmod{9753}$$

となり、上位部分と下位部分とを並列計算することができるようになる。

【0016】

実際に計算すると、乗数の上位部分は、 $23357600 \pmod{9753} = 8918$ となり、下位部分は、 $114072 \pmod{9753} = 6789$ になる。

このように、乗数4321を上位2桁と下位2桁に分割すると、下位部分は、6桁を4桁で割る割り算となり、確かに計算が簡単になる。

【0017】

ところが、上位部分の計算は、結局8桁を4桁で割る割り算が必要となる。つまり、乗算剰余算においては、乗数を単純に上位部分と下位部分とに分割して、それを並列計算しても、結局、上位部分で分割前と同じ桁数同士の割り算（ここでは、8桁を4桁で割る割り算）を行う必要がある。

【0018】

このように、乗算剰余算では、乗数を上位部分と下位部分に分割して計算しても、並列処理を行う利点が得られなかった。

そこで、本発明では、剰余系を新たに定義する領域に変換し、乗数を上位部分と下位部分とに分割した場合に、剰余演算の部分で上位部分、下位部分ともに計算の桁数を減らして並列計算ができるようにし、ひいては計算の高速化を可能としたのである。

【0019】

ここで、乗算剰余算と新たに定義する領域との関係を図1を参照しつつ説明する。

図1に示すように、剰余系における変数 U, V を新たに定義する領域では、各々 $X = U \cdot R \pmod{M}$ 、 $Y = V \cdot R \pmod{M}$ に変換する。また、剰余系における乗算剰余算 $U \cdot V \pmod{M}$ を $X \cdot Y \cdot R^{-1} \pmod{M} = U \cdot V \cdot R \pmod{M}$ に置き換える。

10

20

30

40

50

【0020】

ここで、 M が r 進であるとき、 $R = r^m$ とすると、図1のS100に示すように、被乗数 U は、 $X = U \cdot r^m \bmod M$ に変換される。また乗数 V も同様に、 $Y = V \cdot r^m \bmod M$ に変換される(図1のS102を参照)。つまり、 U, V が新たに定義する領域の X, Y に変換される。

【0021】

そして、乗算剰余算を、 $U \cdot V \bmod M$ から、新たに定義する領域における、 $X \cdot Y \cdot r^{-m} \bmod M$ に置換する(図1のS104を参照)。

このようにして、パラメータ m を導入することにより、 n 桁の乗数 Y を上位($n - m$)桁の Y_H と下位 m 桁の Y_L との二つの部分に分割し、これらを並列に処理することが可能となる。ここで、 $0 < m < n$ 、 m :整数とする。

10

【0022】

このように、その新たに定義した領域において、変数(乗数) Y を上位部分 Y_H と下位部分 Y_L とに分けて、それらを並列に計算し、その計算結果を逆変換して元の領域に戻すことにより最終的に乗算剰余算の結果を得るという剰余系の計算方法を発明したのである。

【0023】

その発明とは、請求項1に記載のように、整数 M を法とする剰余系において、変数 U, V を法 M と互いに素で M より小さな定数 R を用いて、変数 $X = U \cdot R \bmod M$ 、変数 $Y = V \cdot R \bmod M$ 、に変換し、剰余系における乗算剰余算、 $U \cdot V \bmod M$ を演算

20

$$X \cdot Y \cdot R^{-1} \bmod M \cdots \text{式1}$$

に置換し、剰余系における計算と同じ計算を行い、その計算結果 Z を演算

$$Z \cdot R^{-1} \bmod M \cdots \text{式2}$$

にて逆変換して剰余系における計算結果を得ることを特徴とする。

【0024】

このような計算方法によれば、各変数 U, V が定数 R で、 $X = U \cdot R \bmod M$ 、 $Y = V \cdot R \bmod M$ 、に変換され、剰余系における乗算剰余算 $U \cdot V \bmod M$ が新たに定義される $X \cdot Y \cdot R^{-1} \bmod M$ に置換される。この、変数の変換と乗算剰余算式の置換とによって得られる代数系を「新たに定義した領域」と呼ぶことにすると、剰余系での計算が新たに定義した領域において、同じように計算できる。

30

【0025】

この新たに定義した領域における乗算剰余算においては、前述したように乗数 Y を上位部分 Y_H と下位部分 Y_L とに分割して計算することができる。

したがって、剰余系の演算を新たに定義した領域において計算しようとするとき、上記の変換や逆変換が必要となるものの、乗算剰余算において乗数 Y を上位部分 Y_H と下位部分 Y_L とに分けて計算できるので、例えば公開鍵暗号等の計算のように、剰余系において乗算剰余算を繰返し行わなければならない計算を高速化することができる。

【0026】

つまり、新たに定義した領域において、上位部分 Y_H 及び下位部分 Y_L の計算を並列計算によって、剰余系における乗算剰余算と同じ計算結果を得ることができるので、剰余系において乗算剰余算を繰返し行わなければならない計算を高速化することができるのである。

40

【0027】

ここで、上記式1及び式2における $\bmod M$ は、 M を法とする剰余算であり、その計算の結果は通常0から($M - 1$)の値をとるが、ここでは計算結果が法 M と合同であり、 M 以上あるいは負である場合も含むものである。なお、上記式1及び式2における $\bmod M$ に限らず、本明細書において、 $\bmod M$ はすべて同様の意味である。

【0028】

そして、新たに定義した領域において、請求項2に記載のように、変数 Y が r 進で n 桁であるとき、定数 R を、

50

$$R = r^m$$

とし、変数 Y を

$$Y = Y_H \cdot r^m + Y_L \cdot \dots \text{式 3}$$

によって、上位 $(n - m)$ 桁の Y_H と下位 m 桁の Y_L とに分割し、(m は、 $m < n$ を満たす整数) 式 1 を、

$$(X \cdot Y_H \bmod M + X \cdot Y_L \cdot R^{-1} \bmod M) \bmod M \cdot \dots \text{式 4}$$

に変換して、式 4 の

$$X \cdot Y_H \bmod M \cdot \dots \text{式 4 a}$$

と、

$$X \cdot Y_L \cdot R^{-1} \bmod M \cdot \dots \text{式 4 b}$$

と、を並列処理で実行するようにするとよい。

10

【0029】

このように、式 4 a による計算と式 4 b による計算とを並列処理すれば、剰余系における乗算剰余算に対応する新たに定義した領域での計算速度を容易に高速化することができる。

【0030】

すなわち、式 4 a による計算と式 4 b による計算とは各々の計算方法が異なるためその計算速度が異なるので、式 4 a による計算と式 4 b による計算とを各々の計算速度に合わせるようにする。

【0031】

例えば、式 4 a による計算の計算速度が速ければ、乗数 Y を分割する際の上位部分 Y_H の桁数 $(n - m)$ を下位部分 Y_L の桁数 m よりも多くし、逆に、式 4 b による計算の計算速度が速ければ、上位部分 Y_H の桁数 $(n - m)$ を下位部分 Y_L の桁数 m よりも少なくするようにする。すると、両方の計算をほぼ同時に終了させることができるので、全体での計算時間を容易に短縮、換言すれば、計算速度を高速化することができるのである。

20

【0032】

以上のように、新たに定義した領域において、乗数 Y を上位部分 Y_H と下位部分 Y_L とに分割して計算することによって、剰余系における乗算剰余算の演算速度を高速化することができるが、分割した上位部分 Y_H を計算するための式 4 a による計算と下位部分 Y_L を計算するための式 4 b による計算とには、「背景技術」で述べたように種々の計算方法がある。

30

【0033】

例えば、式 4 a による計算にインターリーブ法を用い、式 4 b による計算にモンゴメリ乗算における計算方法を用いると、従来のプログラムや演算回路を利用することができるので、プログラムや演算装置を容易に構成することができる。したがって、プログラム作成や演算装置のコストダウン等が可能になる。

【0034】

なお、式 4 a と式 4 b による計算は、必ずしも独立している必要はなく、式 4 a と式 4 b による計算を行っていく過程で同期をとって、お互いの中間計算結果をやり取りしつつ計算を行うようにしてもよい。

40

【0035】

また、 r 進、 n 桁の変数 Y において、 n 桁とは変数 Y の桁数として設定された桁数のことであり、例えば、 $r = 2$ で $n = 8$ (つまり、2 進 8 桁) の場合、 $Y = 00001010$ のように、桁の上位桁 (この場合上位 4 桁) が 0 であるものも含んでいる。

【0036】

請求項 3 に記載の乗算剰余演算装置は、 r 進の整数 M を法とする剰余系において (ただし、 M と r とは互いに素)、法 M 、 r 進で n 桁の変数 Y 及び変数 X が入力されたときに変数 Y を上位 $(n - m)$ 桁の Y_H 及び下位 m 桁の Y_L に分割する分割手段 (10: この欄においては、発明に対する理解を容易にするため、必要に応じて「発明を実施するための最良の形態」欄において用いた符号を付すが、この符号によって請求の範囲を限定することを

50

意味するものではない。)と、変数 Y を分割した上位 $(n - m)$ 桁の Y_H 、変数 X 及び法 M で

$$X \cdot Y_H \bmod M$$

を計算して出力する第 1 乗算剰余算器 (20) と、変数 Y を分割した下位 m 桁の Y_L 、変数 X 及び法 M で

$$X \cdot Y_L \cdot r^{-m} \bmod M$$

を計算して出力する第 2 乗算剰余算器 (30) と、第 1 乗算剰余算器 (20) の出力及び第 2 乗算剰余算器 (30) の出力を加算し、その加算結果を出力する加算器 (40) と、を備えたことを特徴とする乗算剰余演算装置である。

【0037】

このように構成された乗算剰余演算装置の出力、つまり加算器 (40) から出力される計算結果は、0 から $(M - 1)$ の値以外に、法 M と合同であり、かつ、 M 以上あるいは負である場合の計算結果を得ることができる。

【0038】

そして、このように構成された乗算剰余演算装置によれば、第 1 乗算剰余算器 (20) において実行される上位 $(n - m)$ 桁の計算と、第 2 乗算剰余算器 (30) において実行される下記 m 桁の計算とが並列計算される。そして、変数 Y_H と Y_L とは分割される前の変数 Y に比べ、桁数が減っているので、各々の乗算剰余算器 (20, 30) での計算速度は速くなる。したがって、剰余系における乗算剰余算に対応する新たに定義した領域での計算速度を容易に高速化することができる。

【0039】

また、請求項 2 で説明したように、第 1 乗算剰余算器 (20) で実行される計算 (式 4 a に相当) と第 2 乗算剰余算器 (30) で実行される計算 (式 4 b 相当) とが並列処理されているので、剰余系における乗算剰余算に対応する新たに定義した領域での計算速度を容易に高速化することができる。

【0040】

すなわち、第 1 乗算剰余算器 (20) で実行される計算と第 2 乗算剰余算器 (30) で実行される計算とは各々の計算方法が異なるため、その計算速度が異なる。そこで、第 1 乗算剰余算器 (20) で実行される計算と第 2 乗算剰余算器 (30) で実行される計算とを各々の計算速度に合わせて実行する。

【0041】

例えば、第 1 乗算剰余算器 (20) での計算速度が速ければ、乗数 Y を分割する際の上位部分 Y_H の桁数 $(n - m)$ を下位部分 Y_L の桁数 m よりも多くし、逆に、第 2 乗算剰余算器 (30) での計算速度が速ければ、上位部分 Y_H の桁数 $(n - m)$ を下位部分 Y_L の桁数 m よりも少なくするようにする。すると、両方の乗算剰余算器 (20, 30) での計算をほぼ同時に終了させることができるので、全体での計算時間を容易に短縮、換言すれば、計算速度を高速化することができるのである。

【0042】

以上のように、新たに定義した領域において、乗数 Y を上位部分 Y_H と下位部分 Y_L とに分割して計算することによって、剰余系における乗算剰余算の演算速度を高速化することができるが、分割した上位部分 Y_H を計算するための第 1 乗算剰余算器 (20) 及び下位部分 Y_L を計算するための第 2 乗算剰余算器 (30) には種々の回路構成がある。

【0043】

例えば、第 1 乗算剰余算器 (20) にインターリーブ法による乗算剰余を行う回路を用い、第 2 乗算剰余算器 (30) にモンゴメリ乗算における計算を行う回路を用いると、従来の回路を利用することができるので、回路を容易に構成することができる。したがって乗算剰余演算装置のコストダウン等が可能になる。

【0044】

ところで、第 1 乗算剰余算器 (20) の処理と第 2 乗算剰余算器 (30) の処理とは必ずしも独立している必要はなく、第 1 剰余算器 (20) の処理と第 2 乗算剰余算器 (30)

10

20

30

40

50

)の処理との過程で同期をとって、お互いの中間処理結果をやり取りしつつ処理を行うように各々を構成してもよい。

【0045】

そして、請求項5に記載のプログラムは、請求項1又は請求項2に記載の剰余系の計算方法をコンピュータに実行させるためのプログラムである。

つまり、コンピュータに、整数Mを法とする剰余系において、変数U、Vを法Mと互いに素でMより小さな定数Rを用いて、変数 $X = U \cdot R \pmod{M}$ 、変数 $Y = V \cdot R \pmod{M}$ に変換させ、剰余系における乗算剰余算、 $U \cdot V \pmod{M}$ を請求項1に記載の式1により置換させ、剰余系における計算と同じ計算を実行させ、その計算結果Zを請求項1に記載の式2により逆変換させて剰余系における計算結果を得るプログラムである。

10

【0046】

また、そのプログラムにおいて、コンピュータに、更に、法M及び変数Yがr進でn桁であるとき、定数Rを、 $R = r^m$ とし、変数Yを請求項2に記載の式3によって上位(n-m)桁の Y_H と下位m桁の Y_L とに分割させ、(mは、 $m < n$ を満たす整数)、請求項1に記載の式1を請求項2に記載の式4に変換させ、式4のうちの請求項2に記載の式4aと式4bを並列処理で実行させるプログラムである。

【0047】

このようなプログラムは、請求項1又は請求項2に記載の計算方法によって得られる効果を備えたプログラムとなる。

上記各プログラムは、FD、MO、DVD-ROM、CD-ROM、ハードディスク等のコンピュータによって読み取り可能な記録媒体に記録しておき、必要に応じてコンピュータにロードして起動することにより用いることができる。また、ROMやバックアップRAMに本プログラムを書き込んでおき、これらのROMやバックアップRAMをコンピュータに組み込んで用いてもよい。

20

【発明を実施するための最良の形態】

【0048】

以下に、本発明の実施形態を式及び図面とともに説明する。

(新たに定義した領域における計算方法の説明)

Rを r^m とする。ただし、 $0 < m < n$ で、 r^m は整数とする。このとき、 $R = r^m$ はMと互いに素である。

30

【0049】

この新たに定義した領域における計算を効率よく行う計算方法として、基数をrとした場合について以下に示す。

法を M ($r^{n-1} < M < r^n$ 、かつ、Mとrとが互いに素)とし、n桁の被乗算X、n桁の乗算Y ($0 < X, Y < M$)を入力とし、 $Z = X \cdot Y \cdot r^{-m} \pmod{M}$ を出力する演算であり、具体的には以下の手順で実行される。

【0050】

ステップ1として、S、Tを0に初期化し、被乗数XをAに代入し、乗数Yをパラメータmにより、上位(n-m)桁の Y_H と下位m桁の Y_L とに分割して、各々を B_H と B_L とに代入する。

40

【0051】

すなわち、下位 B_L はm桁、上位 B_H は(n-m)桁となる。

ステップ2としてAと B_H に従来のインターリーブ法を適用して、(n-m)桁分の乗算剰余算を行い、その結果としてSを得る。また、Aと B_L に従来のモンゴメリ乗算における計算方法を適用して、m桁分の計算を行い、Tを得る。

【0052】

このステップ2のインターリーブ法による乗算剰余算と、モンゴメリ乗算における計算とは並列処理で実行される。

そして、並列処理が終了して、SとTの両者が得られると、ステップ3へ移行する。

【0053】

50

ステップ3として、Mを法としてSとTとの加算剰余算を行い、出力Zを得る。
このようにして最終的に得られたZが新たに定義した領域での計算結果となる。
以上の手順を式に表すと下記ようになる。

【0054】

入力：M： $r^{n-1} < M < r^n$ ， $\text{gcd}(M, r) = 1$
0 $X, Y < M$

出力： $Z = X \cdot Y \cdot r^{-m} \text{ mod } M$

演算手順：

ステップ1：A := X；M := M；S := 0；T := 0；

$B_H := Y_H$ ； $B_L := Y_L$ （但し、 $Y = Y_H r^m + Y_L$ ）

10

ステップ2：{ S := Interleaved_modmul(A, B_H , M)；
T := Montgomery_modmul(A, B_L , M)；}

ステップ3：Z := S + T mod M；

なお、ステップ2の中で、Interleaved_modmul(A, B_H , M)は、前述のインターリーブ法による乗算剰余算を示しており、Aは被乗数、 B_H は乗数、Mは法を示している。

【0055】

また、Montgomery_modmul(A, B_L , M)とは、前述のモンゴメリ乗算における計算方法を示しており、Aは被乗数、 B_L は乗数、Mは法を示している。

また、ステップ2における{ }内の計算は並列に行う。

【0056】

20

次に、以上に説明した新たに定義した領域における計算において、乗数を上位部分 Y_H と下位部分 Y_L の桁数を同じにした場合、つまりmを $n/2$ とした場合の演算方法について図2を参照しつつ説明する。

【0057】

図2は、新たに定義した領域での計算手順を被乗数Xと乗数Yとが各々8ビットの場合について模式的に示した図である。

ここでは、基数を2とし、基数2の符号付きディジット表現により、すべての加減算を桁上げの伝搬なしに行うものとする。

【0058】

nビットの被乗数X、nビットの乗数Y（0 $X, Y < M$ ）を入力とし、 $Z = X \cdot Y \cdot 2^{-n/2} \text{ mod } M$ を出力する演算であり、具体的には以下の手順で実行される。

ステップ1として、S, Tを0に初期化し、被乗数XをAに代入する。さらに、乗数Yをパラメータm（= $n/2$ ）により、上位の Y_H と下位の Y_L とに分割し、各々を B_H と B_L とに代入する（図2のS110参照）。

【0059】

ステップ2として、nビットのAと $n/2$ ビットの B_H に従来のインターリーブ法を適用し、また、Aと $n/2$ ビットの B_L に従来のモンゴメリ乗算における計算方法を適用して、各々の計算を並列処理で実行し、各々の計算結果であるSとTとを得る（図2のS112参照）。

【0060】

40

ここで、上位部分の計算は以下の手順（H1）～（H7）を $n/2$ 回繰り返すことにより行われる。

（H1）Sを左（上位）へ1桁シフトする。

【0061】

（H2）シフト後のSの（ $n+1$ ）番目の桁（シフトした際の桁上がり）、n番目の桁、（ $n-1$ ）番目の桁を q_1 、つまり、 $q_1 = [s_{n+1} s_n s_{n-1}]$ とする。

（H3） q_1 が0よりも大きければSから法Mを減じて、それを新たなSとし、 q_1 が0よりも小さければSに法Mを加算したものを新たなSとする。

【0062】

（H4）被乗数Aと乗数 B_H の最上位ビット b_{n-1} との論理積をとり、その結果にSを

50

加算したものを新たな S とする。

(H5) 新たな S の $(n+1)$ 番目の桁、 n 番目の桁、 $(n-1)$ 番目の桁を q_2 、つまり、 $q_2 = [s_{n+1} s_n s_{n-1}]$ とする。

【0063】

(H6) q_2 が 0 よりも大きければ S から法 M を減じて、それを新たな S とし、 q_2 が 0 よりも小さければ S に法 M を加算したものを新たな S とする。

(H7) B_H を左へ 1 ビットシフトする。

【0064】

また、下位部分の演算は以下の手順 (L1) ~ (L4) を $n/2$ 回繰り返すことにより行われる。

10

(L1) 被乗数 A と乗数 B_L の最下位ビット b_0 との論理積をとり、その結果に T を加算したものを新たな T とする。

【0065】

(L2) T の最下位桁 t_0 を q_3 とする。

(L3) q_3 が 0 でなければ、 T に法 M を加算して、それを右へ 1 桁シフトしたものを新たな T とする。一方、 q_3 が 0 であれば T を右へ 1 桁シフトし、それを新たな T とする。

【0066】

(L4) B_L を右へ 1 ビットシフトする。

以上のように、並列処理で上位部分の乗算剰余算結果 S と下位部分の乗算剰余算結果 T とを得る。

20

【0067】

次に、ステップ 3 として、 S と T とを加算し、それを新たな S とする (図 2 の S 1 1 4 参照)。

そして、新たな S の $(n+1)$ 番目の桁、 n 番目の桁、 $(n-1)$ 番目の桁を q_2 、つまり、 $q_2 = [s_{n+1} s_n s_{n-1}]$ とする。

【0068】

q_2 が 0 よりも大きければ S から法 M を減じて、それを新たな S とし、 q_2 が 0 よりも小さければ S に法 M を加算したものを新たな S とする。

ステップ 4 として、 S を周知の演算方法により、基数 2 の符号付きディジット表現から通常の 2 進数表現に変換し、その結果を出力 Z とする (図 2 では省略)。

30

【0069】

ステップ 5 として、出力 Z が 0 よりも小さければ、出力 Z に法 M を加算して、それを出力 Z とする (図 2 の S 1 1 6 参照)。

このようにして最終的に得られた出力 Z が乗数 Y と被乗数 X との法 M に基づく新たに定義した領域における乗算剰余算結果となる。

【0070】

以上の手順を式に表すと下記のようなになる。

入力: $M: 2^{n-1} < M < 2^n$ 、 $\text{gcd}(M, 2) = 1$

40

$X, Y: 0 < X, Y < M$

出力: $Z = X \cdot Y \cdot 2^{-n/2} \bmod M$ ($0 < Z < M$)

演算手順:

ステップ 1: $A := X; B_H := Y_H; B_L := Y_L; S := 0; T := 0; M := M;$

ステップ 2: for $i := 1$ to $n/2$

do H and L in parallel

H: do

$S := 2 \cdot S;$

$q_1 := [s_{n+1} s_n s_{n-1}]$

if $q_1 > 0$ then $S := S - M$

50

```

else if  $q_1 < 0$  then  $S := S + M$ ;
 $S := S + b_{n-1} \cdot A$ ;
 $q_2 := [s_{n+1} s_n s_{n-1}]$ ;
if  $q_2 > 0$  then  $S := S - M$ 
else if  $q_2 < 0$  then  $S := S + M$ ;
 $B_H := B_H < < 1$ ;
enddo
L: do
 $T := T + b_0 \cdot A_i$ ;
 $q_3 := t_0$ 
if  $q_3 > 0$  then  $T := (T + M) > > 1$ 
else  $T := T > > 1$ ;
 $B_L := B_L > > 1$ ;
enddo
endfor
ステップ3:  $S := S + T$ ;
 $q_2 := [s_{n+1} s_n s_{n-1}]$ ;
if  $q_2 > 0$  then  $S := S - M$ 
else if  $q_2 < 0$  then  $S := S + M$ ;
ステップ4:  $Z := SD2\_to\_Binary(S)$ ;
ステップ5: if  $Z < 0$  then  $Z := Z + M$ ;

```

以上に説明した、基数を2とした場合の新たに定義した領域における計算では、 n ビットの乗数が同じビット数($n/2$)の上位部分と下位部分とに分割され、上位部分はインターリーブ法で乗算剰余算され、下位部分はモンゴメリ乗算における計算方法で計算されている。さらに、その2つの計算は並列処理で、ほぼ同時に実行されている。

【0071】

従って、新たに定義した領域での計算によれば、 n ビットの乗数全体をインターリーブ法により乗算剰余算する場合の約 $1/2$ の時間で実行することができる。

すなわち、通常、処理するビット数が同じであれば、インターリーブ法よりもモンゴメリ乗算における計算の方が高速であるので、インターリーブ法により $n/2$ ビットの乗算剰余算(つまり、上位部分の乗算剰余算)が終了したときには、モンゴメリ乗算における計算方法による $n/2$ ビットの計算(つまり、下位部分の計算)は終了している。

【0072】

従って、新たに定義した領域における計算によれば、乗数全体をインターリーブ法によって乗算剰余算を行う場合に比べ、約 $1/2$ の演算時間で計算を実行することができるのである。

【0073】

なお、本実施形態では、乗数のビットを上位と下位のビット数を同じ、つまり m を $n/2$ としたが、上位部分と下位部分のビット数が異なるようにしてもよい。

例えば、インターリーブ法とモンゴメリ法との演算速度の違いを考慮すれば、全体の計算時間をより短縮できる分割方法を決定することができる。つまり、インターリーブ法とモンゴメリ乗算における計算方法との演算速度の比が $p:q$ のときには、下位部分のビット数 m をおよそ、 $n \cdot q / (p + q)$ とすれば、上位部分と下位部分の計算時間がほぼ同じになり、全体の計算時間を短縮することができる。

(乗算剰余演算装置の説明)

次に、上記説明した新たに定義した領域における計算を実行するための乗算剰余演算装置1について図3に従って説明する。

【0074】

図3は、乗算剰余演算装置1の構成を表すブロック図である。

10

20

30

40

50

図 3 に示すように、乗算剰余演算装置 1 は、主に分割回路 10、乗算剰余算回路 20、モンゴメリ乗算回路 30、加算回路 40、剰余算回路 50 とを備えている。

【0075】

分割回路 10 は、入力された r 進、 n 桁の変数 Y を上位 $(n - m)$ 桁の Y_H と下位 m 桁の Y_L とに分割するための回路である。なお、 n 桁の変数 Y を上位部分の Y_H と下位部分の Y_L とに分割するためのパラメータ m は、分割回路 10 の内部にあらかじめ設定されていてもよいし、外部から入力されるようになっていてもよい。

【0076】

乗算剰余算回路 20 は、分割回路 10 で分割された変数 Y の上位 $(n - m)$ 桁の Y_H と入力された変数 X 及び法 M によって

$$S = X \cdot Y_H \text{ mod } M \cdots \text{式 10}$$

を計算して、 S を加算回路 40 に出力するための公知の回路である。

【0077】

モンゴメリ乗算回路 30 は、分割回路 10 で分割された変数 Y の下位 m 桁の Y_L と入力された変数 X 及び法 M によって、モンゴメリ乗算に基づいた計算、

$$T = X \cdot Y_L \cdot r^{-m} \text{ mod } M \cdots \text{式 20}$$

を実行して、 T を加算回路 40 に出力するための公知の回路である。

【0078】

加算回路 40 は、乗算剰余算回路 20 の出力 S 、モンゴメリ乗算回路 30 の出力 T を入力とし、 $S + T$ を計算して出力するための公知の回路である。

剰余算回路 50 は、加算回路 40 の出力 $S + T$ 及び法 M を入力として剰余算を行い、 Z を出力するための回路、すなわち、

$$Z = S + T \text{ mod } M \cdots \text{式 30}$$

により Z を出力するための公知の回路である。

【0079】

以上のように構成された乗算剰余演算装置 1 における計算の流れについて説明する。

乗算剰余演算装置 1 には、 r 進、 n 桁の変数 Y 、変数 X 及び法 M が入力される。

入力された変数 Y は、分割回路 10 に入力され、変数 X は、乗算剰余算回路 20、モンゴメリ乗算回路 30 に入力され、法 M は、乗算剰余算回路 20、モンゴメリ乗算回路 30 及び剰余算回路 50 に入力される。

【0080】

乗算剰余演算装置 1 に入力された変数 Y は、分割回路 10 に入力され、 $(n - m)$ 桁の上位部分 Y_H と m 桁の下位部分 Y_L とに分割される。

分割された上位部分 Y_H は、乗算剰余算回路 20 に入力され、変数 X と法 M とで上記式 10 に従って乗算剰余算が実行され、その結果 S が出力される。

【0081】

一方、分割された下位部分 Y_L は、モンゴメリ乗算回路 30 に入力され、変数 X と法 M とで上記式 20 に従ってモンゴメリ乗算に基づく計算が実行され、その結果 T が出力される。

【0082】

そして、乗算剰余算回路 20 の出力 S 、モンゴメリ乗算回路 30 の出力 T が加算回路 40 に入力されて加算され、その出力と法 M とが剰余算回路 50 に入力され、上記式 30 に従って、剰余算が実行され、その結果 Z が出力される。

【0083】

このような構成の乗算剰余演算装置 1 によれば、乗数 Y を 2 分割し、上位部分 Y_H と下位部分 Y_L とに割り当てて各々の乗算剰余算を独立して並列処理で実行している。従って、従来のように n 桁の乗数をそのまま乗算剰余算する演算方法に比べ、少ない時間で乗算剰余算を実行することができる。

【0084】

なお、本実施形態において、分割回路 10 が分割手段に、乗算剰余算回路 20 が第 1 乗

10

20

30

40

50

算剰余算器に、モンゴメリ乗算回路 30 が第 2 乗算剰余器に、加算回路 40 が加算器に、剰余算回路 50 が剰余算器に各々相当する。

【0085】

以上、本発明の実施形態について説明したが、本発明は、本実施形態に限定されるものではなく、種々の態様を採ることができる。

例えば、本実施形態では、乗数を分割した上位部分の計算（式 4 a による計算）にインターリーブ法を用い、下位部分の計算（式 4 b による計算）にモンゴメリ乗算における計算を用いたが、各計算はそれらに限定されるものではなく、数学的に並列処理できる計算方法であればどのような計算方法であってもよい。

【0086】

また、並列処理の際、式 4 a と式 4 b とによる計算は、必ずしも独立している必要はなく、式 4 a と式 4 b とによる計算を行っていく過程で同期をとって、お互いの中間計算結果をやり取りしつつ計算を行うようにしてもよい。

【0087】

同様に、乗算剰余算回路 20 の処理とモンゴメリ乗算回路 30 の処理とは必ずしも独立している必要はなく、乗算剰余算回路 20 の処理とモンゴメリ乗算回路 30 の処理との過程で同期をとって、お互いの中間処理結果をやり取りしつつ処理を行うように各々を構成してもよい。

【0088】

また、乗算剰余演算装置 1 では、乗除算器 50 の出力を Z としているが、加算器 40 の出力を他の回路や装置、例えば、他の乗算剰余算回路に入力して、加算器 40 の計算結果を基に更なる演算を行うようにしてもよい。

【図面の簡単な説明】

【0089】

【図 1】剰余系における r を基数とする各変数と新たに定義した領域における表現との関係を示す図である。

【図 2】新たに定義した領域における演算手順を被乗数 X と乗数 Y とが各々 8 ビットの場合について模式的に示した図である。

【図 3】乗算剰余演算装置 1 の構成を表すブロック図である。

【符号の説明】

【0090】

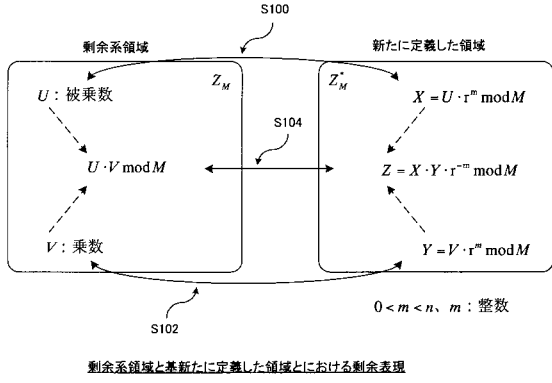
1 ... 乗算剰余演算装置、10 ... 分割回路、20 ... 乗算剰余算回路、30 ... モンゴメリ乗算回路、40 ... 加算回路、50 ... 剰余算回路。

10

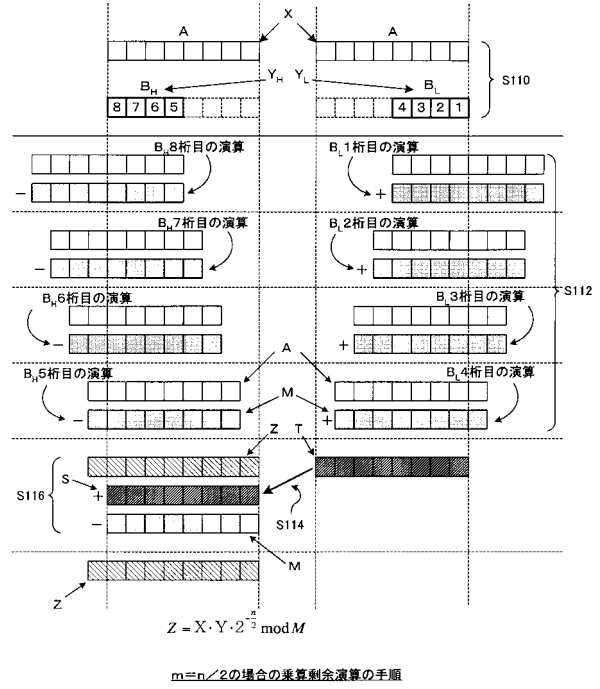
20

30

【図1】



【図2】



【図3】

