(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4316760号
(P4316760)
(45)発行日　平成21年8月19日(2009.8.19)　(24)登録日　平成21年5月29日(2009.5.29)

(51)Int.Cl.　　F I
G06F 17/10 (2006.01)　　G06F 17/10 S
G09C 1/00 (2006.01)　　G09C 1/00 650Z

請求項の数 1　(全 14 頁)

(21)出願番号　特願2000-23948(P2000-23948)
(22)出願日　平成12年2月1日(2000.2.1)
(65)公開番号　特開2001-216288(P2001-216288A)
(43)公開日　平成13年8月10日(2001.8.10)
審査請求日　平成19年1月31日(2007.1.31)

(73)特許権者　396017453
リコーソフトウエア株式会社
東京都中央区勝どき３－１２－１
(74)代理人　100102587
弁理士　渡邊　昌幸
(72)発明者　段　暁平
東京都中央区勝どき３－１２－１フォアフロントタワー　リコーシステム開発株式会社内
(72)発明者　梁　青
東京都中央区勝どき３－１２－１フォアフロントタワー　リコーシステム開発株式会社内

最終頁に続く

(54)【発明の名称】積和演算方法およびそのプログラムを記録した記録媒体

(57)【特許請求の範囲】
【請求項１】
少なくとも初期値加算器と排順器と楕円演算器として機能するＣＰＵを内蔵したＩＣカードにより、Ｋｉを多ビット長２進数列の係数、Ｐｉを計算集合とする積和演算式（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）を演算するため、係数Ｋｉの一部のビット列の内容を判定して分類することにより演算する積和演算方法であって、
楕円曲線上で複数点の積和演算を行う場合に、前記積和演算式（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）の係数Ｋ１～Ｋｔを各係数のビット位置をそろえた状態で垂直方向に整列させてマトリックスに配置するとともに、ｊ方向に上位ビットから３ビットずつを２ビット単位でずらしながら分割して配列しビット列ｋ（ｉ，ｊ）を生成し、
前記初期化加算器により、前記生成したビット列ｋ（ｉ，ｊ）をi方向に順次読み出し、該係数Ｋｉの先頭ビット値が１か０かを判定し、１の場合にのみ、そのｉに対応するＰｉについて和をとり、
次に、前記排順器により、前記生成したビット列ｋ（ｉ，ｊ）をｊ方向に３ビットずつに区切って、前記ｋ（ｉ，ｊ）の最初からの３ビットをｉ方向にそのパターンを第１分類（０００，１１１）、第２分類（０１１，１００）、第３分類（００１，０１０，１０１，１１０）の３種類のいずれかに分類し、
前記楕円演算器により、前記排順器による前記分類の結果に基づき、前記３種類のいずれかに分類された全ての３ビットのパターンに対して楕円２倍演算を行い、その結果に対して第２分類の場合にはその３ビットのパターンに対して楕円加算を行い、その結果に対

して再度、前記３種類のいずれかに分類された全ての３ビットのパターンに対して楕円２倍演算を行い、さらにその結果に対して第３分類の場合にはその３ビットのパターンに対して楕円加算を行い、前記ｋ（ｉ，ｊ）の最初からの３ビットに対する処理が最後のｉ（ｉ＝ｔ）まで終了したら、次に、前記ビット列ｋ（ｉ，ｊ）のｊ方向の引き続く３ビットに対して、同様の処理を順次繰り返し、前記ビット列ｋ（ｉ，ｊ）の全てのビットに対する上記処理が終了したら、上記全ての演算結果を加算して積和（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）を得ることを特徴とする積和演算方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、暗号の計算や天文学の計算等に利用される積和演算方法とそのプログラムを記録した記録媒体に関し、特に膨大な項数を持つ積和演算を高速に計算することが可能な積和演算方法とそのプログラムを記録した記録媒体に関する。

【０００２】

【従来の技術】

積和演算方法の１つとして、ここではコンピュータ暗号の組立てと翻訳（復号化）に使用される積和演算について述べる。暗号系の安全性を計る尺度を暗号強度と呼ぶが、その尺度の１つとして、解読可能な限界データ量（原字数）で表わす強度と、一定基準に対する比較係数で表わす実際的強度とがある。

最近、より小さい鍵サイズで、従来用いられているＲＳＡ暗号と同等の安全性が期待できる楕円曲線暗号が注目されている。この楕円曲線暗号は、楕円曲線上の各点Ｐ１，Ｐ２，・・・Ｐｔの積和演算（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）を行うものである。その場合に、曲線上の点の数が多ければ多いほど安全性は高くなるが、この演算を高速に行うことが要望される。従来は、専用の電子計算機で演算を実行させていた。ここで、Ｋｉは係数（１からｔまでの任意の数値）であり、Ｐｉは１からｔまでの計算集合である。

上記の積和演算（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）を従来の符号付きバイナリ演算法に基づいて計算する場合を説明する。従来の符号付きバイナリ演算法は、説明に関しては、楕円曲線上の点である演算楕円点が１つのケースしか扱われておらず、Ｋ１Ｐ１＋・・・＋ＫｔＰｔのような複数点に対しては、各々の演算楕円点に対してＫｉ倍の演算を施し、最後に全体の和を取る手法が取られる。従って、従来の符号付きバイナリ演算法で複数の演算楕円点を持つ楕円曲線暗号を演算する場合には、膨大な時間がかかっていた。

【０００３】

曲線上で複数点の積和の計算において、Ｋ１・・Ｋｔを２（ｎ＋１）ビット長の整数とすると、Ｋｉは図２（ａ）に示すようなビット列となる。ここで、ビット列長は、２（ｎ＋１）とする。

曲線上の複数底点の積和Ｑは、次式（１）となる。

$$Q=K1P1+\cdots KtPt=\Sigma_{(j=2n+1\sim 0)}\{k_{(1,j)}2\hat{}j\}\times P1+\cdots\cdots\cdots+\Sigma_{(j=2n+1\sim 0)}\{k_{(t,j)}2\hat{}j\}\times Pt \quad \cdots\cdots\cdots\cdots (1)$$

上記の手法は、Ｐ１の計算集合についてＫ１回繰り返し演算した後、Ｐ２の計算集合についてＫ２回繰り返し演算し、Ｐ３の計算集合についてＫ３回繰り返し演算する、というように最後まで直列的に計算していく方法である。しかし、これでは極めて時間がかかり過ぎるので、従来も、これより高速に積和計算ができる手法を考えている。その手法は、全部のビットを見ないで一部のビット（ここでは３ビット）のみを取って無駄な計算を省略しようとするものである。

すなわち、Ｋｉの計算集合の演算結果としては３種類（Ｐ＋Ｑ，Ｐ－Ｑ，２Ｐ）の値しか得られないことが分っている。このうち、＋か－かは同じ分類に含まれ、Ｐ－ＱはＰ＋（－Ｑ）に含ませることができる。結局、計算集合の演算結果は図８（ａ）に示す２Ｐの２倍演算と、図８（ｂ）に示すＰ１＋Ｐ２（＝Ｐ＋Ｑ）の加算の２種類となる。（ａ）は楕円２倍演算器（ＥＡ２Ｐ）を示し、（ｂ）は楕円加算演算器（ＥＡＰＱ）を示している。

【０００４】

図１（ｂ）は、従来の楕円点の積和を計算するアルゴリズムにおいて、係数Ｋｉ（ｉ＝１・・ｔ）のビット走査方法を示す図であり、図１５はビット走査時の判別方法の説明図である。
従来の走査方法では、図１（ｂ）に示すように、先ずＫ１を係数とする計算集合Ｐｉについて３ビット２ｎ＋１～２（ｎ－１）＋１番目を走査し、それらの入力したビットに対して計算を行い、次に係数Ｋ１の次の３ビット２（ｎ－１）＋１～２（ｎ－２）＋１番目を走査し、それらの入力したビットに対して計算を行い、順次１ビット重複させながら３ビットずつ走査を行い、内容を判別してその判別通りの計算を行っていく。Ｋ１を係数とする計算集合について走査が終了したならば、次にＫ２を係数とする計算集合について走査、判別、演算を行い、同じようにして、Ｋ３を係数とする計算集合、Ｋ４を係数とする計算集合，・・・Ｋｔの順に直列的に判別を行って、最後にそれらの和を計算して全体の演算を終了する。
【０００５】
例えば、Ｓ＝１５＊Ｐ１＋９＊Ｐ２の演算ステップで、仮にＫ１＝１５であれば、１５の２進数＝Ｂ＇００１１１１であるから、３ビット毎に走査するビット列は図１５に示すように、１回目は〔００１〕、２回目は〔１１１〕、３回目は〔１１０〕となる。〔００１〕のステップ１～３で計算結果はＳ１＝Ｐ１となり、〔１１１〕のステップ１～３で計算結果は４Ｐ１となり、〔１１０〕のステップ１～３で１５Ｐ１となる。次に、９の２進数＝Ｂ＇００１００１であるから、３ビット毎に走査するビット列は図１５に示すように、１回目は〔００１〕、２回目は〔１００〕、３回目は〔０１０〕となる。〔００１〕のステップ１～３で計算結果はＰ２となり、〔１００〕のステップ１～３で計算結果は２Ｐ２となり、〔０１０〕のステップ１～３で計算結果は９Ｐ２となる。ステップ７では両者を加算してＳ１＋Ｓ２＝１５Ｐ１＋９Ｐ２となり、これがＳの演算結果である。
【０００６】
図１２は、従来のアルゴリズムのビット走査方法を示すフローチャートである。
先ず、ｉ＝１とし（ステップ６１）、Ｋｉ，Ｐｉを入力して（ステップ６２）、係数Ｋ１の上位から３ビットＫ（ｉ，ｊ），ｋ（ｉ，ｊ－１），ｋ（ｉ，ｊ－２）（初期値ｊ＝２ｎ＋１）を走査し、それらの入力したビットに対してＫｉＰｉの演算器（ＥＡＳＰ）により計算（ＳＥＰ）を行い（ステップ６３）、ｉ＝１＋１にインクリメントして（ステップ６４）、ｉ＝ｔになるまでこれを繰り返し行う（ステップ６５）。
【０００７】
図１３は、図１２におけるＫｉ＊Ｐｉの演算器（ＥＡＳＰ）の詳細フローチャートである。
先ず、ビット列長ｊ＝２ｎ＋１を設定し（ステップ７１）、次に係数Ｋの値ｋ（ｉ，ｊ），ｋ（ｉ，ｊ－１），ｋ（ｉ，ｊ－２）を入力する（ステップ７２）。そして、入力３ビットに対して楕円演算（ＳＥＰ）を行い（ステップ７３）、次にｊ＝ｊ－２にディクリメントして（ステップ７４）、ｊ＝０になるまで（ステップ７５）、この演算操作を繰り返し行う。ｊ＝０になった時点で、全てのビット列の演算が終了したので、Ｓ＝Ｋｉ＊Ｐｉの演算結果を出力する（ステップ７６）。
【０００８】
図１４は、３ビットデータに対応する楕円演算器（ＳＥＰ）のフローチャートである。
従来の楕円演算器では、先ず、係数Ｋｉの３ビットｋ（ｉ，ｊ），ｋ（ｉ，ｊ－１），ｋ（ｉ，ｊ－２）を入力するとともに（ステップ８１）、楕円Ｐｉを入力する（ステップ８２）。次に、演算器ＥＡ２Ｐにより楕円２倍演算を行う（ステップ８３）。次に、入力した３ビット値を判別し（ステップ８４）、その３ビットが〔０００〕または〔１１１〕であれば楕円２倍演算を行って、Ｓ＝２Ｓとし（ステップ８６）、３ビットが〔００１〕，〔０１０〕，〔１０１〕または〔１１０〕であれば楕円２倍演算を行って、Ｓ＝２Ｓとし（ステップ８５）、さらに次の３ビットが〔００１〕または〔０１０〕であれば楕円加算を行って、Ｓ＝Ｓ＋Ｐｉとし（ステップ８９）、また次の３ビットが〔１０１〕または〔１１０〕であれば楕円加（減）算を行い、Ｓ＝Ｓ＋（－）Ｐｉとする（ステップ９０）。

最初の３ビットが〔０１１〕であれば楕円加算を行って、Ｓ＝Ｓ＋ＰｉとしⅠステップ８７）、３ビットが〔１００〕であれば楕円加（減）算を行って、Ｓ＝Ｓ＋（－）ＰｉとしⅠステップ８８）、これらの後に楕円２倍演算を行って、Ｓ＝２Ｓとする（ステップ９１）。

【０００９】

このように、従来の３ビットデータに対応する楕円演算器の計算では、入力した３ビットデータに対して、必ず２回の楕円２倍演算（ＥＡ２Ｐ）を行うため、ｔ個の３ビット列に対する演算では、２ｔ回の楕円２倍演算が必要であった。

すなわち、演算量は、楕円２倍演算器（ＥＡ２Ｐ）の演算回数＝２ｔ（ｎ＋１）であり、楕円加（減）算器（ＥＡＰＱ）の演算回数＝（ｎ＋１）ｔ＋ｔ－１である。

結局、従来の演算方法では、楕円２倍演算器（ＥＡ２Ｐ）の回数２ｔ（ｎ＋１）は、ｔの倍数であるため、楕円曲線の点数を多くした場合、つまり積和乗算の項数ｔが増加したならば、２倍演算の回数が増加することである。

【００１０】

【発明が解決しようとする課題】

このように、従来の符号付きバイナリ演算法では、全体を直列的に演算するため、演算に時間がかかり過ぎるという問題があった。また、従来による３ビットデータに対応する楕円演算器では、３ビットデータ全体を直列的に順次判定していく方法をとっており、その場合の楕円２倍演算器（ＥＡ２Ｐ）の演算回数は２ｔ（ｎ＋１）回であり、また楕円加（減）算器（ＥＡＰＱ）の演算回数は（ｎ＋１）ｔ＋ｔ－１回も必要であった。従って、いずれもｔの倍数だけ演算回数が増加するため、ｔが増加したとき、すなわち、楕円暗号の安全性を大にするためｔの値を大にした場合には、演算回数が膨大な数になるので、演算時間が膨大となる。

また、暗号アルゴリズムをＩＣカードに実装する場合に、ＩＣカードは演算能力や記憶容量に限界があるため、膨大な容量のアルゴリズムを実装することには問題があった。

【００１１】

そこで、本発明の目的は、これら従来の課題を解決し、積和演算を行う場合に、楕円２倍演算の回数を式の項数ｔと無関係にして、項数ｔを大にした場合でも高速に積和演算を行うことができ、かつ容量の限られたＩＣカード等の記録媒体にも容易に実装して、積和演算を実行させることが可能な積和演算方法およびそのプログラムを記録した記録媒体を提供することにある。

【００１２】

【課題を解決するための手段】

上記目的を達成するため、本発明の積和演算方法では、複数点の積和演算（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）において、従来のようにＫ１を係数とする計算集合について一部のビット（例えば３ビット）の走査、判別、計算を行い、次にＫ２、Ｋ３、・・・Ｋｔの順に直列的に判別および計算を行う方法を採用せずに、少なくとも初期値加算器と排順器と楕円演算器として機能するＣＰＵを内蔵したＩＣカードにより、Ｋｉを多ビット長２進数列の係数、Ｐｉを計算集合とする積和演算式（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）を演算するため、係数Ｋｉの一部のビット列の内容を判定して分類することにより演算する積和演算方法であって、楕円曲線上で複数点の積和演算を行う場合に、積和演算式（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）の係数Ｋ１～Ｋｔを各係数のビット位置をそろえた状態で垂直方向に整列させてマトリックスに配置するとともに、ｊ方向に上位ビットから３ビットずつを２ビット単位でずらしながら分割して配列しビット列ｋ（ｉ，ｊ）を生成し、前記初期化加算器により、前記生成したビット列ｋ（ｉ，ｊ）をｉ方向に順次読み出し、該係数Ｋｉの先頭ビット値が１か０かを判定し、１の場合にのみ、そのｉに対応するＰｉについて和をとり、次に、前記排順器により、前記生成したビット列ｋ（ｉ，ｊ）をｊ方向に３ビットずつに区切って、前記ｋ（ｉ，ｊ）の最初からの３ビットをｉ方向にそのパターンを第１分類（０００，１１１）、第２分類（０１１，１００）、第３分類（００１，０１０，１０１，１１０）の３種類のいずれかに分類し、次に、前記楕円演算器により、前記排順器によ

る前記分類の結果に基づき、前記３種類のいずれかに分類された全ての３ビットのパターンに対して楕円２倍演算を行い、その結果に対して第２分類の場合にはその３ビットのパターンに対して楕円加算を行い、その結果に対して再度、前記３種類のいずれかに分類された全ての３ビットのパターンに対して楕円２倍演算を行い、さらにその結果に対して第３分類の場合にはその３ビットのパターンに対して楕円加算を行い、前記ｋ（ｉ，ｊ）の最初からの３ビットに対する処理が最後のｉ（ｉ＝ｔ）まで終了したら、次に、前記ビット列ｋ（ｉ，ｊ）のｊ方向の引き続く３ビットに対して、同様の処理を順次繰り返し、前記ビット列ｋ（ｉ，ｊ）の全てのビットに対する上記処理が終了したら、上記全ての演算結果を加算して積和（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）を得ることを特徴としている。

これにより、本発明では、項数の多い積和演算式を少ない実行回数で高速に演算することができ、従って演算にかかる時間も少なくて済む。また、従来のような専用プロセッサの演算能力は不要となり、ＩＣカードの演算能力で楕円曲線の積和演算が可能となり、ＩＣカード等の記録媒体へのアルゴリズムの実装も容易となった。

【００１３】

【発明の実施の形態】

以下、本発明の実施例を、図面により詳細に説明する。

図１（ａ）は、本発明の一実施例を示す積和演算方法のビット走査手順の説明図であり、図２（ｂ）は図１（ａ）で形成されたマトリックスのビット配列表の図である。

曲線上の複数点の積和演算式（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）において、Ｋ１～Ｋｔごとに図２（ｂ）に示すマトリックスを形成し、係数Ｋ１～Ｋｔを上記ビット列ｋ（ｉ，ｊ）ずつに分割し、係数Ｋ１からＫｔまでの最初のビット列ｋ（ｉ，ｊ）を垂直方向に走査して、内容を判定し予め定めた分類を行った後に、該分類に基づいて演算を行い、係数Ｋ１からＫｔまでの次のビット列ｋ（ｉ，ｊ）を垂直方向に走査して、内容を判定し分類した後に該分類に基づいて演算を行い、これを繰り返し実行して、最後に全ての演算結果を加算する。

すなわち、図１（ａ）に示すように、係数Ｋ１からＫｔまでの上位３ビットＫ１の２ｎ＋１～２（ｎ－１）＋１を矢印のように縦方向に走査し、それらの３ビットについて入力ビットを分類し、分類したビット列に対して演算を行う。次に、係数Ｋ１からＫｔまでの最終ビットを重複させた次の３ビット２（ｎ－１）＋１～２（ｎ－２）＋１を縦方向に走査し、それらの入力ビットを分類し、分類したビット列に対して演算を行う。さらに係数Ｋ１からＫｔまでの次の３ビット２（ｎ－２）＋１～２（ｎ－３）＋１を縦方向に走査し、演算を行い、順次、同じ動作を繰り返す。最後に、これらの積演算を行って処理を終了する。

曲線上の複数基底点の積和は、次式（２）で表わされる。

$$K1P1+\cdots KtPt=\Sigma_{(j=2n+1\sim 0)}\{k_{(1,j)}\times P1+\cdots+k_{(t,j)}\times Pt\}\times 2^{j} \quad \cdots\cdots\cdots (2)$$

【００１４】

図３は、本発明の一実施例を示す積和演算のアルゴリズムの基本フローチャートである。

本実施例では、先ず初期値計算（ＥＡＩＰQ）を行い（ステップ１１）、次に最初の項ｊ＝２ｎ＋１を設定して（ステップ１２）、次に、図５に示す演算を行う排順器（ＥＡＳＯＲＴ）により順次演算を行う（ステップ１３）。そして、表ＴＡ，表ＴＢを参照しながら、図６に示す演算を行う楕円演算器（ＥＡＴＡＬＬ）により演算を行う（ステップ１４）。なお、初期値計算を行う初期値加算器（ＥＡＩＰQ）の詳細については図４に、排順器（ＥＡＳＯＲＴ）の詳細については図５に、また楕円演算器（ＥＡＴＡＬＬ）の詳細については図６に、それぞれ示されている。

ここまでが、マトリックスのビット列ｋ（ｉ，ｊ）の垂直方向の処理であって、次に、マトリックスの次の３ビットを指定するために、ｊ＝ｊ－２を設定する（ステップ１５）。そして、この動作を積和演算の項数が終了するまで、つまりｊ＝０になるまで繰り返し行う（ステップ１６）。ｊ＝０になった時点で積演算を行い、処理を終了する（ステップ

１７）。
本実施例においては、楕円２倍演算の回数はｔと関係ないため、２倍演算の回数は２（ｎ＋１）の固定値である。
【００１５】
図４は、図３における初期値加算器（ＥＡＩＰＱ）の詳細フローチャートである。
ＥＡＩＰＱ演算器は、本発明のアルゴリズムのメインループ処理に至る前処理を行う演算器である。先ず、入力Ｋｉ，Ｐｉに初期値ｉ＝１を入力して（ステップ２１）、ｉ≦ｔであることを確認し（ステップ２２）、最初のＫｉのｋ（ｉ，２ｎ＋１）＝１であるか否かを判別する（ステップ２３）。積和演算（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）を行う場合、演算の種類として必ず楕円２倍演算（２Ｐ）が２回発生し、楕円加（減）算（Ｐ＋Ｑ）が１回発生する。従って、本発明では、２つあるものは１回だけ計算して２倍すればよいので、計算が不要な項を先頭ビットが１か０かで判定するのである。従って、ｋ（ｉ，２ｎ＋１）＝１のとき楕円演算器（ＥＡＰＱ）で計算を行い、Ｓ＋Ｐｉ＝Ｓとする（ステップ２４）。一方、ｋ（ｉ，２ｎ＋１）＝０のときには計算は不要となるので、ｉの値をインクリメントしてステップ２１に戻る（ステップ２５）。
【００１６】
図５は、図３における演算順次整列図、すなわち排順器（ＥＡＳＯＲＴ）の詳細フローチャートである。
先ず、３ビットのｋ（ｉ，ｊ），ｋ（ｉ，ｊ－１），ｋ（ｉ，ｊ－２）を入力し、初期値ｉ＝１を設定する（ステップ３１）。次に、ｉ≦ｔであることを確認して（ステップ３２）、入力した３ビットを判断する（ステップ３３）。すなわち、３ビットが〔０００〕，〔１１１〕であれば、何もせずにｉを１だけインクリメントし（ステップ３６）、また、３ビットが〔０１１〕，〔１００〕であれば、入力した係数値および係数対応する楕円点で表ＴＡを作成する（ステップ３４）。また、３ビットが〔００１〕，〔０１０〕，〔１０１〕，〔１１０〕であれば、入力した係数値および係数対応する楕円点で表ＴＢを作成する（ステップ３５）。その後、ｉ＝ｉ＋１に設定して（ステップ３６）、ステップ３１に戻る。
【００１７】
図６は、図３におけるｔ個楕円点の演算を行う演算器（ＥＡＴＡＬＬ）の詳細フローチャートである。
楕円演算器（ＥＡＴＡＬＬ）では、先ず、楕円２倍演算器（ＥＡ２Ｐ）で全てのパターンの係数値に対して２倍演算を行い（Ｓ＝２Ｓ）（ステップ４１）、演算結果Ｓを次の演算器（ＥＡＴＡＢ）に入力する。演算器（ＥＡＴＡＢ）は、表ＴＡ（４２）からのデータと楕円２倍演算器（ＥＡ２Ｐ）からの演算結果Ｓを入力して、演算を行うものであり（ステップ４３）、図７に詳細フローが示されている。演算器（ＥＡＴＡＢ）による演算結果Ｓは次の楕円２倍演算器（ＥＡ２Ｐ）に入力される。楕円２倍演算器（ＥＡ２Ｐ）は、２倍演算を行って（Ｓ＝２Ｓ）、その結果Ｓを次の演算器（ＥＡＴＡＢ）に出力する（ステップ４４）。演算器（ＥＡＴＡＢ）には、この演算結果Ｓと表ＴＢ（４５）からのデータが入力される。この演算器（ＥＡＴＡＢ）の動作も前の演算器（ＥＡＴＡＢ）と同じであって、図７に示されている（ステップ４６）。
【００１８】
図７は、図６における演算器（ＥＡＴＡＢ）の詳細フローチャートである。
表ＴＡまたは表ＴＢの楕円点数＝Ｔａ，Ｔｂが入力されると（ステップ５１）、ｉ≦Ｔａ（Ｔｂ）であることを確認し（ステップ５２）、入力３ビットを判断する（ステップ５３）。そして、入力された３ビットが〔００１〕，〔０１０〕，または〔０１１〕のときには、楕円点加算器（ＥＡＰＱ）により加算Ｓ＝Ｓ＋Ｐを行い（ステップ５４）、また入力された３ビットが〔１００〕，〔１０１〕，または〔１１０〕のときには、楕円点減算器（ＥＡＰＱ）により減算を行い（ステップ５５）、いずれもｉ＝ｉ＋１にインクリメントする（ステップ５６）。そして、ステップ５２に戻って次の３ビットの判断に移る。
【００１９】

図８（ａ）（ｂ）は本発明の方法に利用される部品となる楕円２倍演算器、および楕円加（減）算器の定義図である。
楕円２倍演算器（ＥＡ２Ｐ）では、入力として楕円点Ｐ１の座標値、出力として楕円点Ｓの座標値、演算はＳ＝Ｐ１＋Ｐ２＝２Ｐ１を行うことを示している。その他に座標値が示されている。楕円加（減）算器（ＥＡＰＱ）では、入力として楕円点Ｐ１とＰ２の座標値、出力として楕円点Ｓの座標値、演算はＳ＝Ｐ１＋Ｐ２を行うことを示している。
これらの楕円２倍演算器（ＥＡ２Ｐ）は、図６のステップ４１とステップ４４に使用され、また楕円加（減）算器（ＥＡＰＱ）は図４のステップ２４および図７のステップ５４，５５に使用される。

【００２０】

本発明の積和演算アルゴリズムにより、曲線上点の積和演算（Ｋ１Ｐ１＋・・・＋ＫｔＰｔ）の必要な楕円演算の計算量は、先ず楕円加法では最悪（ｔ－１）回であり、また楕円２倍演算は２（ｎ＋１）回である。また、演算インデクス２ｊと２ｊ＋１に対して係数Ｋｉの演算ビットは同じものであるため、その演算ビット列に対して、楕円加（減）法演算は１回しか行わない。その結果、３ビット判定処理ステップは、最悪ｔ（ｎ＋１）回の楕円加（減）算となる。従って、総計実行回数は（ｔ－１）＋２（ｎ＋１）＋（ｎ＋１）ｔ以下となる。
このように、本発明の演算方法では、従来の演算方法に比べて楕円２倍演算回数は２ｎ（ｔ－１）だけ減少させることができる。
ここで、本発明の積和演算アルゴリズムの特長を記述すると、▲１▼ビット〔０００〕と〔１１１〕に対しては、楕円演算をまとめて行うこと、▲２▼ｔ個楕円点の和をまとめた後、その結果に対して２（ｎ＋１）回の楕円２倍演算だけ必要であるので、楕円点数ｔは楕円２倍演算の回数に対して何等影響がないこと、である。

【００２１】

図９～図１１は、本発明の積和演算方法と従来方法との効果の比較を示す図である。
図９には、従来の拡張符号付きバイナリ法と本発明の積和演算アルゴリズムの計算量（点の加算・２倍演算）の演算回数を示している。点数ｔが１のときには両者の量は同じであるが、ｔの値が多くなればなるほど両者の差は大となる。最右欄に両者の差が示されている。ｔ＝２のときの差は２（ｎ＋１）であり、ｔ＝３のときの差は４（ｎ＋１）である。
次に、図１０には、実装例として１６０（＝２（ｎ＋１））ビット長の係数Ｋｉに対する実際の演算量が示されている。点数ｔが１のときには両者の演算量は同じであるが、ｔの値が多くなればなるほど両者の差は大となる。最右欄には、両者の比率＝（従来の方法演算量－本発明の演算量）／（従来の方法演算量）が示され、ｔ＝２の場合で３３％の差、ｔ＝３の場合で４４％の差が示されている。
次に、図１１には、本発明の実装複数楕円点の積和の演算（１６０ビット）に関する測定時間（ｍｓ）が示されている。
比率１は、１－（２点積和演算実測時間）／（２倍の１点演算実測時間）を求めた値であり、比率２は、１－（３点積和演算実測時間）／（３倍の１点演算実測時間）を求めた値である。複数楕円点の積和演算の時間について、本発明では従来の方法よりも２点の場合には２９％～３６％の範囲で速くなっている。また、３点の場合には３９％～４８％の範囲で速くなっている。

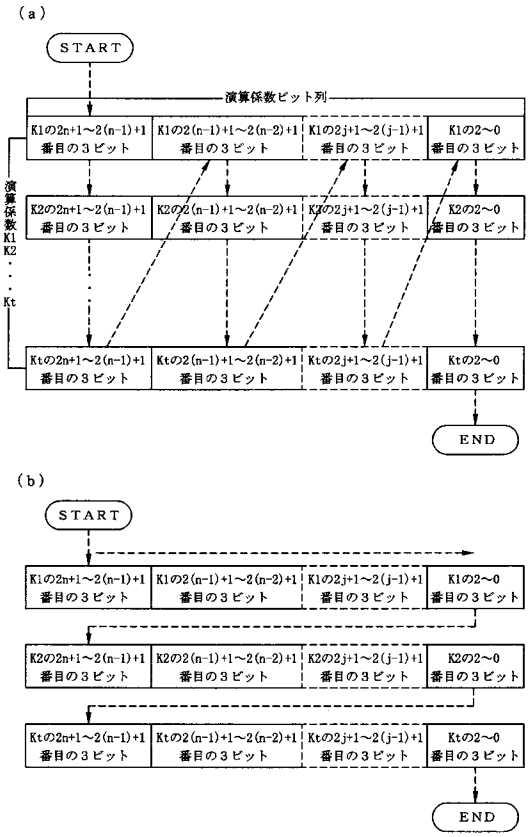
【００２２】

図１５および図１６は、既存の方法と本発明の方法における演算ステップの比較図であり、図１７は既存方法と本発明の方法における演算回数における対照図である。
ここではＳ＝１５＊Ｐ１＋９＊Ｐ２の演算ステップを比較する。
従来方法である図１５では、Ｋ１とＫ２の演算にステップ１～７が必要となる。図１５については、先ずＫ１Ｐ１に対して、（００１→１１１→１１０）のように直列で楕円演算（ステップＳ１～ステップＳ３）を行い、そしてＫ２Ｐ２に対して（００１→１００→０１０）のように直列で同じ演算を行い、最後はＳ←Ｓ１＋Ｓ２よりＳ＝１５Ｐ１＋９Ｐ２を計算している。各演算３ビットに対しては必ず２回の楕円２倍演算（図１５の網掛け部

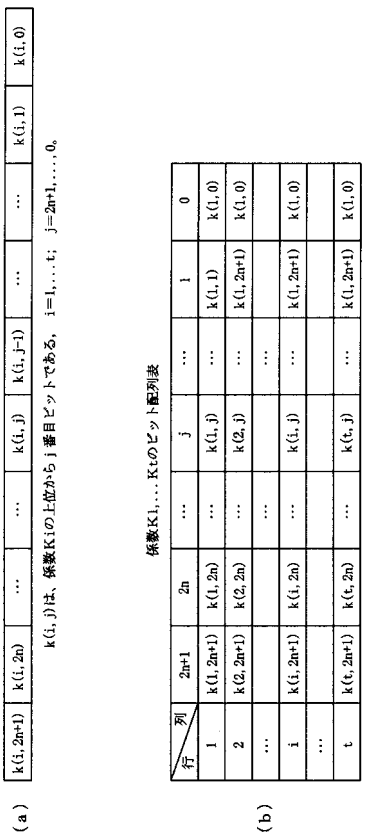
分）が必要である。
一方、本発明の方法は図１６に示すように、先ず、Ｋ１，Ｋ２の先頭ビット（００１，００１）に対して、並列で楕円演算（ステップ１～ステップ４）を行い、そして、次の３ビット（１１１，１００）に対して、並列で同じ演算を行い、このように演算を繰り返す。最後はＳ＝１５Ｐ１＋９Ｐ２を計算する。このように、図１６の演算は、同列のｔ（これ例ではｔ＝２）個楕円点に対して楕円２倍演算は２回しか行いませんので、図１６は図１５より楕円２倍演算の回数を減らすことができる（網掛け部分が少なくてよい）。
【００２３】
図１７には、楕円２倍演算と楕円加（減）算の演算回数の対照が示されている。
楕円２倍演算については、従来の方法は２ｔ（ｎ＋１）＝１２であるのに対して、本発明の方法は２（ｎ－１）＝６である。従って、これらの差は２（ｔ－１）（ｎ＋１）＝６である。
次に、楕円加（減）算については、従来の方法はｔ（ｎ＋１）＝６以下であるのに対して、本発明の方法はｔ（ｎ＋１）＝６以下である。従って、楕円加（減）算については両者の差は０である。
このように、本発明のアルゴリズムは、既存のアルゴリズムより楕円２倍演算の計算量が減少することが特徴である。従って、ＩＣカードに内蔵されたＣＰＵの演算能力でも十分に演算することが可能であり、その利用分野の拡大が期待できる。
【００２４】
なお、図３～図７に示した本発明の方法のアルゴリズムをそれぞれプログラム化して、ＩＣカードは勿論のことＣＤ－ＲＯＭ等の記録媒体に格納することにより、カード読み取り装置やパソコンの設置された任意の場所で本発明をいつでも実現することができるという利点がある。
【００２５】
【発明の効果】
以上説明したように、本発明によれば、積和演算を行う場合に、楕円２倍演算の回数を式の項数ｔと無関係にして、項数ｔを大にした場合でも高速に積和演算を行うことができる。また、従来は積和演算のために専用プロセッサを必要としていたが、本発明ではステップ数が減少できるので、ＩＣカードで十分に演算が可能となり、演算プログラムを容量の限られたＩＣカードに実装することで、その利用範囲の拡大が図れる。
【図面の簡単な説明】
【図１】本発明の一実施例を示す積和演算アルゴリズムの演算方法と比較のために示した従来の方法の説明図である。
【図２】本発明が演算する積和演算のビット列および本発明によりマトリックス状に変換した積和演算のビット配列を示す図である。
【図３】本発明の一実施例を示す積和演算方法の概略動作フローチャートである。
【図４】図３における初期値加算器（ＥＡＩＰＱ）の詳細フローチャートである。
【図５】図３における排順器（ＥＡＳＯＲＴ）の詳細フローチャートである。
【図６】図３における演算器（ＥＡＴＡＢ）の詳細フローチャートである。
【図７】図６における楕円演算器（ＥＡＴＡＬＬ）の詳細フローチャートである。
【図８】本発明に使用される部品としての楕円２倍演算器（ＥＡ２Ｐ）および楕円加（減）算器（ＥＡＰＱ）の定義を示す図である。
【図９】本発明の方法と従来の方法の演算量についての対照を示す図である。
【図１０】本発明の方法と従来の方法の１６０ビット長の場合の演算量の対照を示す図である。
【図１１】楕円積和演算の実行時間の対照を示す図である。
【図１２】従来の演算アルゴリズムを示すフローチャートである。
【図１３】図１２における楕円演算器（ＥＡＳＰ）の詳細フローチャートである。
【図１４】図１３における３ビットデータに対応する楕円演算器（ＳＥＰ）の詳細フローチャートである。

【図１５】従来のアルゴリズムの演算ステップを示す説明図である。
【図１６】本発明のアルゴリズムの演算ステップを示す説明図である。
【図１７】本発明の方法と従来の方法の演算回数についての対照を示す図である。
【符号の説明】
ＥＡＩＰＱ…初期値計算器、Ｓ…積和累計値、ＥＡＳＯＲＴ…排順器、
ＥＡＴＡＬＬ…楕円演算器、ＥＡＴＡＢ…表Ｔに関する演算器、
ＥＡ２Ｐ…楕円２倍演算器、ＥＡＰＱ…楕円加（減）算器、
ＥＡＳＰ…Ｋｉ＊Ｐｉの演算器、ＳＥＰ…３ビットデータに対する楕円演算器。

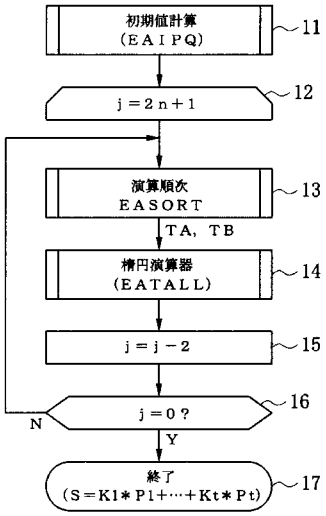
【図１】



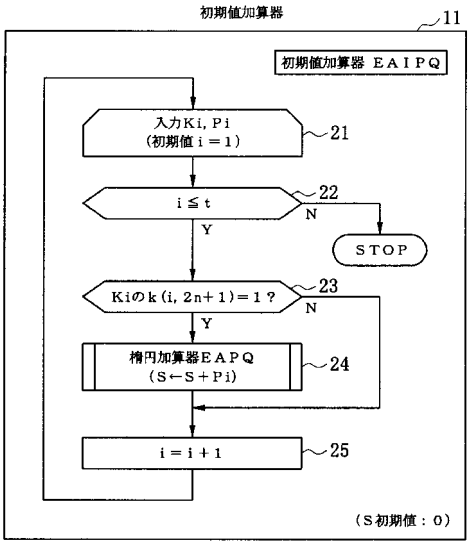
【図２】



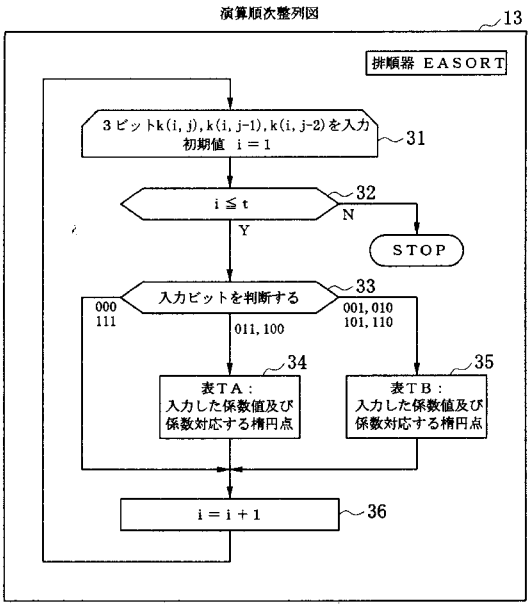
【図３】


初期値計算
（ＥＡＩＰＱ）
11
ｊ＝２ｎ＋１
12
演算順次
ＥＡＳＯＲＴ
13
ＴＡ，ＴＢ
楕円演算器
（ＥＡＴＡＬＬ）
14
ｊ＝ｊ－２
15
ｊ＝０？
16
Ｎ
Ｙ
終了
（Ｓ＝Ｋ１＊Ｐ１＋…＋Ｋｔ＊Ｐｔ）
17

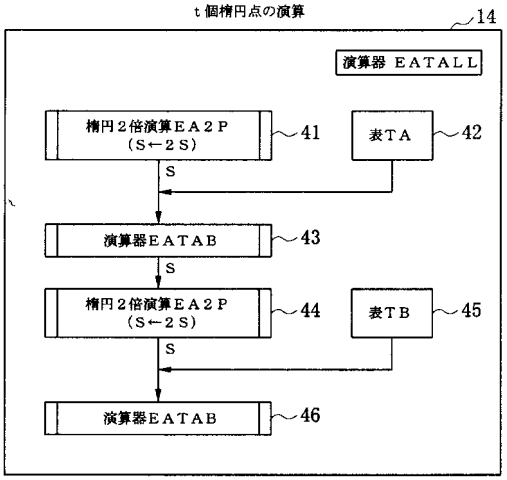
【図４】


初期値加算器
11
初期値加算器 ＥＡＩＰＱ
入力Ｋｉ，Ｐｉ
（初期値ｉ＝１）
21
ｉ≦ｔ
22
Ｎ
Ｙ
ＳＴＯＰ
Ｋｉのｋ（ｉ，２ｎ＋１）＝１？
23
Ｎ
Ｙ
楕円加算器ＥＡＰＱ
（Ｓ←Ｓ＋Ｐｉ）
24
ｉ＝ｉ＋１
25
（Ｓ初期値：０）

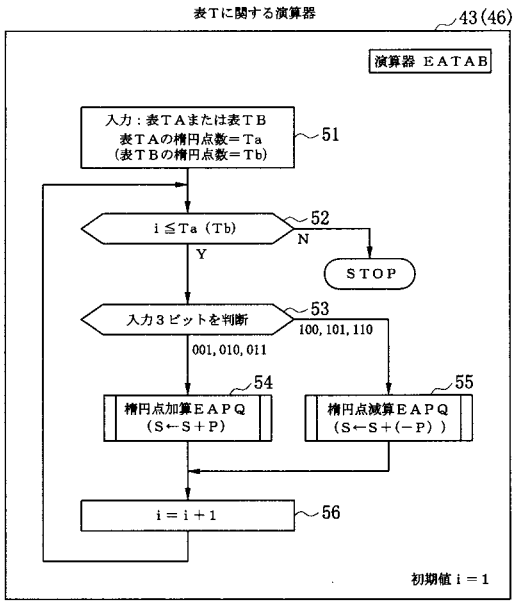
【図５】


演算順次整列図
13
排順器 ＥＡＳＯＲＴ
３ビットｋ（ｉ，ｊ），ｋ（ｉ，ｊ-１），ｋ（ｉ，ｊ-２）を入力
初期値 ｉ＝１
31
ｉ≦ｔ
32
Ｎ
Ｙ
ＳＴＯＰ
入力ビットを判断する
33
000
111
001，010
101，110
011，100
表ＴＡ：
入力した係数値及び
係数対応する楕円点
34
表ＴＢ：
入力した係数値及び
係数対応する楕円点
35
ｉ＝ｉ＋１
36

【図６】


ｔ個楕円点の演算
14
演算器 ＥＡＴＡＬＬ
楕円２倍演算ＥＡ２Ｐ
（Ｓ←２Ｓ）
41
表ＴＡ
42
Ｓ
演算器ＥＡＴＡＢ
43
Ｓ
楕円２倍演算ＥＡ２Ｐ
（Ｓ←２Ｓ）
44
表ＴＢ
45
Ｓ
演算器ＥＡＴＡＢ
46

【図７】



【図８】

楕円２倍演算

(a)

楕円２倍演算器S←２Ｐ１：　EA2P

入力： 楕円点 Ｐ１＝(ｘ１, ｙ１)
出力： 楕円点 Ｓ＝(ｘ, ｙ)
演算： Ｓ＝Ｐ１＋Ｐ１＝２Ｐ１

ｘ＝ｒ^２－２ｘ１
ｙ＝ｒ(ｘ１－ｘ)－ｙ１
ｒ＝(３ｘ１^２＋ａ)／２ｙ１

楕円加算

(b)

楕円加算演算器(Ｓ←Ｐ１＋Ｐ２)：　EAPQ

入力： 楕円点 Ｐ１＝(ｘ１, ｙ１), Ｐ２＝(ｘ２, ｙ２);
出力： 楕円点 Ｓ＝(ｘ, ｙ)
演算： Ｓ＝Ｐ１＋Ｐ２

ｘ＝ｒ^２－ｘ１－ｘ２
ｙ＝ｒ(ｘ１－ｘ)－ｙ１
ｒ＝(ｙ２－ｙ１)／(ｘ２－ｘ１)

【図９】

演算量(点の加算・２倍算の回数)の対照表

演２算点数ｔ	従来の方法	ＲＳＫ実装方法	二者の差
1	3(n＋1)	3(n＋1)	0
2	6(n＋1)＋1	4(n＋1)＋1	2(n＋1)
3	9(n＋1)＋2	5(n＋1)＋2	4(n＋1)

【図１０】

演算量の対照表(係数長Ki＝２(ｎ＋１)＝１６０bit)長の場合

演算点数個数ｔ	従来の方法	ＲＳＫ実装方法	二者の差	比率＊
1	２４０	２４０	0	0%
2	４８１	３２１	１６０	３３%
3	７２２	４０２	３２０	４４%

＊比率＝(従来の方法演算量－ＲＳＫ実装方法の演算量)／従来の方法の演算量

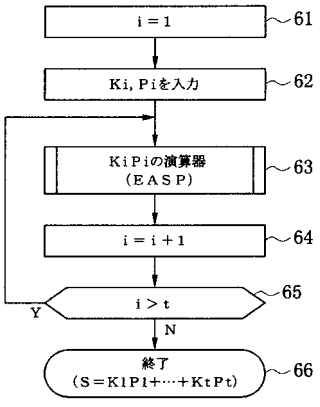
【図１１】

楕円積和演算(１６０bit)実行時間(ms)

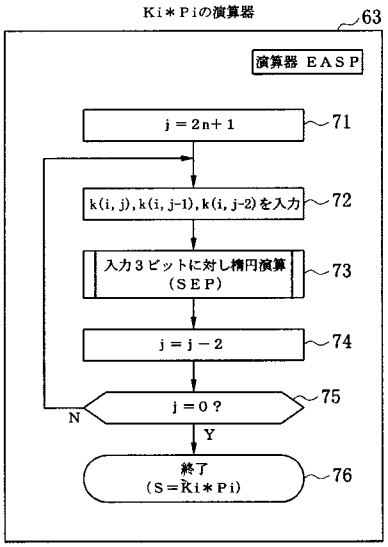
乱数パターン	１点	２点	比率１	３点	比率２
パターン１	４１３.８	５９０.９	(２９%)	７６６.２	(３９%)
パターン２	４３７.８	５８４.０	(３４%)	７５６.４	(４３%)
パターン３	４５８.８	５９３.０	(３６%)	７３４.３	(４７%)
パターン４	４３９.６	６３０.３	(２９%)	７４７.３	(４４%)
パターン５	４７０.０	６０６.１	(３６%)	７３９.４	(４８%)
平均値	４４４.０	６００.９	(３２.３%)	７４８.７	(４４.２%)

比率１＝１－(２点積和演算実測時間)／(２倍の１点演算実測時間)
比率２＝１－(３点積和演算実測時間)／(３倍の１点演算実測時間)

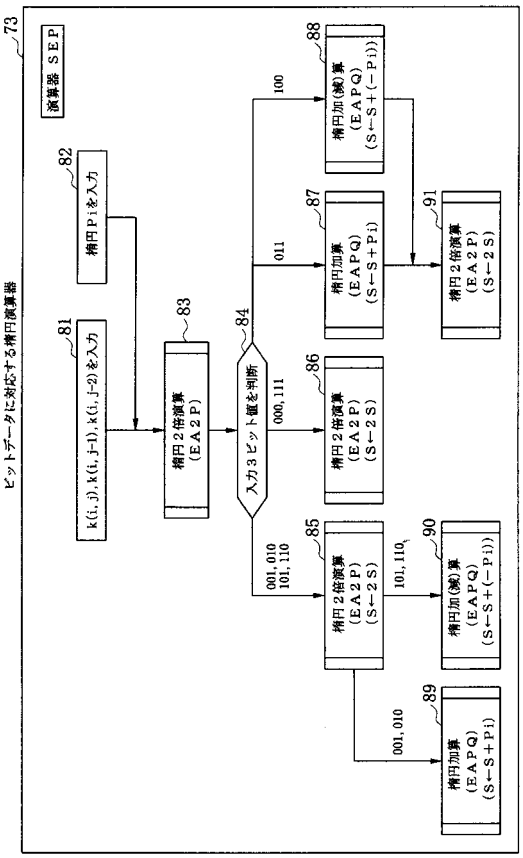
【図１２】



【図１３】



【図１４】



【図１５】

既存アルゴリズムの計算表

演算係数K1＝B'001111	001	111	110	KiPi
Step1	S1←2＊0＝0	S1←2＊S1＝2P1	S1←2＊S1＝8P1	
Step2	S1←2＊S1＝0	S1←2＊S1＝4P1	S1←2＊S1＝16P1	
Step3	S1←S1＋P1＝P1		S1←S1－P1＝15P1	S1＝15P1
演算係数K2＝B'001001	001	100	010	
Step4	S2←2＊0＝0	S2←2＊S2＝2P2	S2←2＊S2＝4P2	
Step5	S2←2＊S2＝0	S2←S2－P2＝P2	S2←2＊S2＝8P2	
Step6	S2←S2＋P2＝P2	S2←2＊S2＝2P2	S2←S2＋P2＝9P2	S2＝9P2
Step7	S←S1＋S2			15P1＋9P2

【図１６】

新しいアルゴリズムの計算表

演算係数K1＝B'001111	001	111	110
演算係数K2＝B'001001	001	100	010
Step1	S←2＊S＝0	S←2＊S＝2P1＋2P2	S←2＊S＝8P1＋4P2
Step2	S←2＊S＝0	処理なし（111）	S←2＊S＝16P1＋8P2
Step3	S←S＋P1＝P1	S←S－P2＝2P1＋P2	S←S－P1＝15P1＋8P2
Step4	S←S＋P2＝P1＋P2	S←2S＝4P1＋2P2	S←S＋P2＝15P1＋9P2

【図１７】

演算回数対照表（ｔ＝２、ｎ＝２）

演算方式	① 既存演算方式	② 新しい演算方式	①−②
楕円２倍演算	２ｔ(ｎ＋１)＝１２	２(ｎ＋１)＝６	２(ｔ−１)(ｎ＋１)＝６
加(減)演算	＜ｔ(ｎ＋１)＝６	＜ｔ(ｎ＋１)＝６	０

フロントページの続き

(72)発明者　吉川　博晴
東京都中央区勝どき３－１２－１フォアフロントタワー　リコーシステム開発株式会社内

審査官　須田　勝巳

(56)参考文献　特開２０００－１５５５２６（ＪＰ，Ａ）
森田　光 外２名，公開鍵系演算の高速化方式，ＮＴＴ　Ｒ＆Ｄ，社団法人電気通信協会，１９９９年１１月１０日，第48巻，第11号，p5-12
原田　俊治 外１名，効率的なｎ変数べき乗剰余演算法の提案とその一応用，電子情報通信学会技術研究報告，社団法人電子情報通信学会，１９９１年１１月２９日，第91巻，第359号，p25-36

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　17/00-17/18
G09C　1/00