# 数式処理システムの現代暗号への応用

富士通研究所
竹島 卓
tak@para.flab.fujitsu.co.jp

**Abstract**

これまでの数理科学の産業応用では, 数値計算に適した数学がもっぱら活用されていたが, 現代暗号技術は数論や代数といった数値計算では対応仕切れない数学の成果を産業界において巧妙に活用する場を提供した. 計算機代数 (数式処理) の現代暗号との関わり方には二つの面がある. ひとつは, 安全な暗号の設計と提供であり, 今一つは暗号への攻撃である. ともに, 現代の計算機の通常の数値計算とは異なる計算能力を最大限に活用する高性能数理計算の技術である.

## 1 まえがき

インターネットの発展による世界的な情報の交換と流通におけるセキュリティの確保のための基礎技術として「暗号」が注目を浴びるようになった. 1977 年の DES (Data Encryption Standard) や 1998 年の Rivest 等による RSA 公開鍵暗号の発明以降発展したと考えられる「現代暗号」は, それまでの暗号に対する「秘密」, 「隠密」, 「諜報活動」, 「非合法」, といった暗いイメージを一新した. すなわち, 暗号化の方式自体が公開され, 誰にでもその安全性や危険性の検証が可能であるため, 暗号提供者自体にだまされるという危険がなくなった. また, 暗号を「情報の秘匿」といった防御的な目的だけでなく, 電子署名や本人認証, 原本 (無改竄) 保証など, 「正当性および真性の証明」という能動的な目的に広く使用する道が開けた.

このようなセキュリティ関連問題の要素技術に暗号があるが, 現代暗号, とくに公開鍵暗号は数学背景を強く持っており, ここに数式処理の全面的な応用が始まった.

### 1.1 安全性と暗号攻撃

外交や軍事といった特定集団の中での情報交換にのみ使用されて来た暗号では, 暗号鍵ばかりでなく, 暗号の方式そのものを含むできる限りの情報を秘密にすることができたが, オープンな世界で使用される現代暗号では, 「検閲」や「盗み見」, 「改竄」といった不正行為が, 暗号方式の提供者自体でさえも不可能であることを示す必要があり, そのため「暗号の算法 (アルゴリズム)」が公開されることが原則である.

暗号の算法公開は暗号を攻撃から防御する上では全く不利な条件であるが, そのようなより厳しい条件の下で「安全性 = 攻撃に対する耐性」を保証する必要がある.

しかしながら, 「どのような攻撃に対しても安全である」という絶対的な安全の保証を数学的に与えることは不可能であり, 「各時点で知られている最強の攻撃法に対して, どの程度の計算資源を費す必要があるか?」という尺度で測らざるを得ないのが実情である.

### 1.2 公開鍵暗号と共通鍵暗号

現代暗号には共通鍵方式と, 公開鍵方式の 2 通りの方式がある. 共通鍵方式とは, 情報の送信者が平文を暗号化するのに用いる鍵と, 暗号の受信者が暗号文を復号するのに用いる鍵とが共通である方式をいう. 一方の, 公開鍵方式は, 暗号化と復号とに用いる鍵が (むろん関連してはいるが) 異なっている方式である.

秘密鍵暗号では DES が現在世界の標準として広く使われている. しかし最近, 汎用の高性能計算機により 4 日足らずで解読されることも分かっており, より安全性が高い次世代の共通鍵暗号 AES の検討が始まっている.

公開鍵暗号として現在実用に供されている RSA 暗号はその安全性の根拠を「素因数分解」の計算量的困難さに置いており, 暗号化復号化には「大きな整数」の冪乗演算と剰余を求める演算が用いられている.

Computer Algebra Application to Cryptography,
Taku Takeshima, Fujitsu Laboratories.

一方, 次世代の公開鍵暗号として有望視されている「楕円曲線暗号」では, 楕円曲線と呼ばれる曲線上の有理点 (座標値が有理数の点) がなす加法群上の演算が利用され, その上での「離散対数問題」の困難さが安全性の根拠となっている.

これらの公開鍵暗号方式ではいずれも数論を基礎としているが, RSA 暗号が「整数」の演算を直接扱うのに対して, 楕円曲線暗号は「楕円曲線上の点」というやや複雑な対象 (より代数的な対象) を扱う点で, 計算機代数 (数式処理) の計算が直接前面に出て来ることになる.

## 2 計算機代数とは

本稿では楕円曲線暗号と計算機代数 (数式処理) 技術の関わりを中心に紹介するが, その前に計算機代数について簡単に紹介しておく.

計算機代数 (Computer Algebra) は古くは数式処理 (Formula Manipulation) と呼ばれていた. 日本では「数式処理」の方が一般には通用している. 天体力学の計算に現れる長大な有理式の計算に計算機資源を利用したのが発端といわれている. 一方, 大学学部程度の不定積分をパターンマッチと試行錯誤に基づいて実行するプログラムのように, 人工知能からのアプローチも数式処理と呼ばれていた.

1970 年代に不定積分の Risch 算法, Berlekamp-Zassenhaus の因数分解算法, 多項式イデアルの Gröbner 基底を求める Buchberger 算法の 3 つの重要な算法が開発され, これらの算法の洗練と効率化が 1980 年代に進むことによって, 数式の「記号と構造の処理」の側面よりも, 数式のもつ数学的 (とくに代数的) な側面が探求され, より高度の数学的対象を扱う算法や, 代数構造に着目した算法の効率化が盛んになった. この頃から, 海外では Computer Algebra という名称が使用されるようになり, 数学への応用が盛んになって来た. 1990 年代になると, 数式処理システムの現代化, すなわち数値計算やグラフ表示, ビットマップディスプレイ上のウィンドウシステムを利用した数式の 2 次元 (入) 出力のサポートなどで, 数理計算のデスクトップ環境ともいうべきシステムが商用化されるようになった. 代表格は Maple や Mathematica であろう.

これら商用のシステムとは別に, 代数幾何や群論, 整数論といった数学の分野に応じた専門性の高いシステムも各地で開発され利用されるようになっている.

さて著者は, 計算機代数とは, 数学に現れるもろもろの計算に計算機の能力を適用しようとする試みである, と考えている. 計算の対象は具体的に計算できるものであれば数値ばかりとは限らない. 多項式や sin, cos などの記号を含む式, 微分や積分の式, さらに種々のダイアグラムで表される数学の関係式も広くいえば対象とされる. しかし, 一般的な数式がすべて扱えるかというと, そうではなく現時点で効果的に計算ができる対象は, 多項式, 有理式であると考えてもそう間違いではない. 多項式や有理式が計算できるというと, なんだその程度か, sin, cos, log はどうした, 微分方程式は解けないのか, 不定積分, 定積分はどうなんだ, と思われるかも知れない. もちろん, 現在の商用数式処理ではそれらの演算も一応できることになってはいるが, もともと構成的でなかったり, 効率的な算法が存在しなかったりするために, 計算機代数ではまだまだ未成熟な分野である.

多項式や有理式といっても, 多くの有用な問題が多項式の問題に帰着できることを考えるとその演算の高度化と効率化は非常に重要である. 例えば非線形の微分方程式 (ソリトン方程式) の特殊解などは非線形の連立代数方程式 (過少系, イデアルの言葉でいえば非零次元イデアル) を解くことに帰着されるものもある. 以下で詳述する楕円曲線暗号設計における諸演算は, 大きな次数と大きな係数 (表現長が大きいという意味) の多項式の演算が本質的である.

## 3 楕円曲線暗号

楕円曲線暗号は Koblitz と Miller が 1986 年に独立に提案した. 楕円曲線暗号は「楕円曲線」と呼ばれる (「楕円」ではないが) 2 次元曲線を基にして構成される. ただし, 暗号に用いられる曲線は実数や複素数という連続量で考えるのではなく, 要素が有限個で四則演算が定義できる離散的な体—有限体という— を基礎として考えるため, その定義や演算は数式に頼らざるを得ない.

実際, 楕円曲線暗号の安全性設計に関わるあらゆる計算は数式処理の対象そのものといってよい. 以下本節では楕円曲線暗号の概説を行う.

数式処理と現代暗号の関わりを述べることが目的であるため, 例外的な事項への対応など, 数学的な細部には必ずしもこだわらないことをあらかじめご了承頂きたい.

**有限体** 以下, $p > 3$ は大きな奇素数とし, 簡単のために考察する有限体 $K$ は $p$ 個の元から成るとする. GF($p$) と表されることもある. ここでの有限体 GF($p$) は整数を $p$ で除した剰余の作る集合と考えてもらいたい. $p = 5$

の場合は，集合 $\{0,1,2,3,4\}$ であり，$2+3=5=0$, $1-4=-3=2$, $2\times 3=6=1$, $2^{-1}=3$ などが演算の例である．一般の有限体の元の個数 $q$は，素数 $p$，正整数 $k$に対して $q=p^k$となる．ここでは $k=1$ の特別な (しかし実用上十分な) 場合を考えることになる．

**楕円曲線の方程式**

**定義 1** $a,b\in K$とするとき，有限体 $K$上の**楕円曲線**とは $y^2=x^3+ax+b \pmod p$ という方程式を満たす点 $(x,y)\in \overline{K}^2$の集合で，これを，$E(\overline{K})$ と表す．ここに $\overline{K}$ は $K$の代数閉体とする．

上の定義で " $\pmod p$" は演算を $p$ で除した剰余で考える，すなわち GF($p$) 上で演算することを表している．また，$K$の代数閉体
*overline*$K$ とは，有理数に対する代数的数—例をあげれば $\sqrt{2}$や $x^3+3x+1=0$ の (どれと特定しない) 根など，有理数係数の代数方程式の根となるもの—の全体と考えて頂ければよい．

**楕円曲線上の有理点のなす加法群** さて，この楕円曲線上にはその $xy$-座標値がともに $K$の元 (すなわち，$0,1,...,p-1$ までの数) であるような点が幾つかある．このような点のことを有理点と呼ぶ．

有理点は加法群をなす．つまり，有理点の集合全体 (および，無限遠点という特別な点) には加法が定義でき，その全体は加法により閉じている．この加法は幾何学的に理解した方が分かりやすい．

楕円曲線の定義式を有限体上でなく複素数上で考えて，その実数成分をグラフ化すると図 1のような $x$ 軸に関して対称な曲線となる．

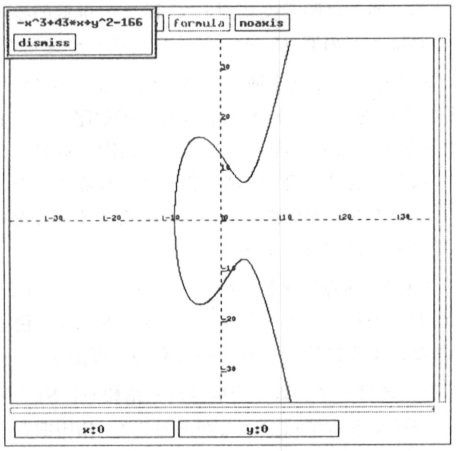


図 1: Elliptic Curve $y^2=x^3-43x+166$ in the Real Plane

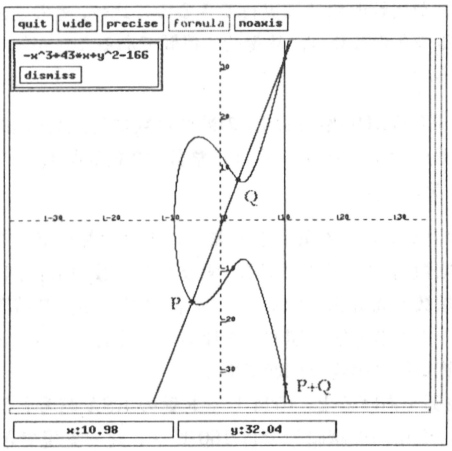


図 2: Addition of Two Points on Elliptic Curve

この曲線上の点 P と Q とに対して，それら和 P+Q がつぎのように定義される．まず，P と Q とを結ぶ直線が再度楕円曲線と交わる第三の点を求める．つぎにこの点の $x$ 軸に対する対称点を求めるとその点が P+Q である．(図 2)

とくに，P と Q とが同一の点の場合には P を通る楕円曲線の接線を P と P とを結ぶ直線と考えて，P+P=2P の点が求められる．

有限体上の楕円曲線は図には表せないが，数式で座標を取り扱う限りは (現れる数が有限体の元であることを除けば) 実数体上と同様に計算できる．

この加法は交換法則と結合法則を満たし，特別な点として無限遠点 (O) を追加することによって，有理点の全体は群 (加法群) を構成する．このとき無限遠点は加法の単位元 (つまり零元) となる．(楕円曲線上の有理点以外の点も加法群をなしている．)

**楕円曲線上の離散対数問題** 楕円曲線上の有理点全体が加法群をなしているので，点 P が与えられるとその

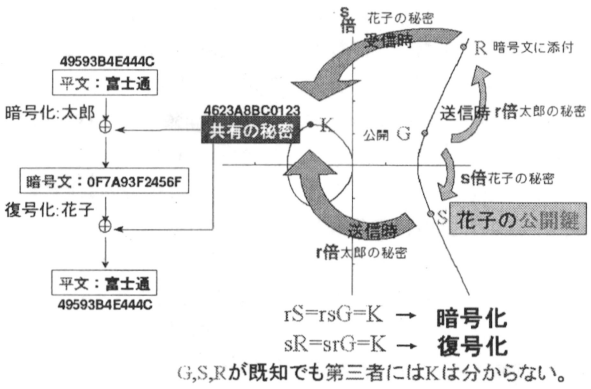


図 3: Scheme of Elliptic Curve Cryptography

$n$ 倍点が計算できる. (P+P=2P, P+P+P=3P, ... など) この計算が比較的簡単に実行できるのに比べ, その逆演算であるつぎの計算は計算量的に難問であることが知られている.

**問題 2 ECDLP(楕円曲線上の離散対数問題)** 楕円曲線上に有理点 P と Q=$n$P *($n$ は正整数)* である点 Q を与えて, $n$ を求めよ.

この計算量は現在知られる限りの最善の算法でも有限体の位数 ($p$ のこと) の対数 (すなわち位数の桁数) に関して指数時間計算量であり, この計算量は準指数時間計算量の素因数分解問題 (RSA 暗号の安全性の根拠) よりも計算量的に困難な問題である.

比較例をあげると, 1024 ビット鍵の RSA 暗号と同じ安全性が楕円曲線暗号では 160 ビット鍵で実現でき, 2048 ビット鍵の RSA 暗号と同等にするには 211 ビット鍵で可能とされている.

**暗号化・復号のスキーム**楕円曲線を使った暗号通信のスキームの一例を図 3に示す.

太郎は花子に「富士通」というメッセージを送るのに, あらかじめ二つの楕円曲線上のデータを作成する. 先ず, 正整数の乱数 $r$を選び, 花子の公開鍵 S を $r$倍して二人のみが共有することになる点 K を求める. さらに, 同じ $r$を使ってシステムで固定され公開されている点 G を $r$倍した点 R をも求めておく. こうした後, 点 K を用いてメッセージのビット列と K のビット列との排他的論理和をとり, 暗号文とする. これにさきに作成しておいた点 R のデータを添付して, 送信データを作成する. これを受信した花子は, 自分だけの秘密である整数 $s$ を使って, 添付されたデータ R を $s$ 倍して 点 K を求める. あとは暗号文に対して K との排他的論理和を求めれば太郎の原メッセージ「富士通」が復元できる.

メッセージ「富士通」を暗号化するために使用する「共有の秘密」というのは, 通常の共通鍵暗号で使われる「共通鍵」の役割を果たしている. 楕円曲線暗号システムではこの共通の鍵を, 第三者に知られることなく必ずしも安全でない通信路を経由して共有することを可能にしている点に注目して頂きたい.

「太郎の秘密」$r$と「花子の秘密」$s$ とは太郎と花子それぞれの秘密として他者には (お互いにも) 決して知られないように管理されているものと仮定する.

第三者はシステム上公開された情報 (G, S) と通信路上で盗聴できる情報 (R) とがあっても, 楕円曲線上の離散対数問題の困難さから, $r$や $s$ を計算することができないため, K の復元は不可能である.

## 4 安全な楕円曲線暗号

先に楕円曲線暗号は RSA 暗号より計算量的に困難な問題に基づいていると述べた. このことには嘘はないが, 楕円曲線暗号システムの安全性は用いる楕円曲線の選び方に依存しており, 不適切な選択をすると簡単に破られてしまうことも有り得るのである. 例をあげると, anomalus curve, supersingular curve と呼ばれる特殊な楕円曲線は安全でないことが知られている. また, このほかにも特殊な楕円曲線を選ぶとその特殊性を突く攻撃

を受ける可能性が高くなると考えられ, 一般の楕円曲線の中から選ぶべきと考えられている.

考察すべきことはいくつかあるが, ここでは天下りにつぎのことを了解して頂くことにする.

**命題 3 安全な楕円曲線** 安全な楕円曲線はその位数 (楕円曲線上の有理点の個数) が素数であることが最重要である.

**安全な楕円曲線暗号の作り方** 上記「命題」を達成するための方法としては, 次の 3 通りの方法がある.

1. Schoof 算法利用
2. 虚数乗法利用
3. Weil 予想利用

「ランダムに楕円曲線を選び, その位数を計算して素数なら採用し, 素数でないなら別の楕円曲線を試す. 」という方法が原理的に最良の方法であることはあきらかである. Schoof 算法は楕円曲線の位数を計算する方法で, これが安全性の面では最良の方法といえるが, 位数の計算が容易でないことが問題であった.

Elkies, Atkin らの改良を加えた Schoof 法の改良 (SEA 法) が我々のグループによって最近実用化された [IKNY98a, IKNY98b] が, それには我々が開発して来た数式処理システム Risa/Asir によるところが大きい. 次節では Schoof 算法と Elkies, Atkin の改良の概要を紹介し, その計算に寄与した数式処理について述べる.

## 4.1 Schoof の算法

有限体 $K$上の楕円曲線の有理点の個数を楕円曲線の位数と呼び $\#E(K)$ と書く.

Schoof の算法は$\#E(K)$ を求めるために $E(\overline{K})$ の点を $E(\overline{K})$ の点に写す線形写像である Frobenius 写像$\phi$ の固有多項式を利用する. ここに, $E(\overline{K}) = \{(x,y) \in \overline{K}^2 | y^2 = x^3 ax + b\}$ である. $P = (x,y) \in E(\overline{K})$ に対して, $\phi(P) = (x^p, y^p) \in E(\overline{K})$ である. この$\phi$の固有多項式は$\phi^2 - t\phi + p$ と書かれる. ここで, $t$ は$\phi$の trace であり, $p$ は基礎となる体 $K = \mathrm{GF}(p)$ の位数 (要素の個数) であった.

一方, 楕円曲線理論から有理点の個数 $\#E(K)$ は, 式 $\#E(K) = p + 1 - t$ を満たすことが分かっている. そこで, なんらかの方法で Frobenius 写像 $\phi$の trace が計算できるならば, $\#E(K)$ が計算できることになる.

Schoof のアイデアの基は, $\phi$の固有多項式を $P \in E(\overline{K})$ に作用させた関係式

$$\phi^2(\mathrm{P}) + p\mathrm{P} = t\phi(\mathrm{P}) \tag{1}$$

にある.

この左辺が適当な $\mathrm{P} \in E(\overline{K})$ に対して計算できれば, 右辺の $t$ をその可能な値の範囲でスイープして右辺の値を求めれば, 左辺と一致した値がでたところで, $t$ の値が見つかる. ただし, スイープする $t$ の値は $0, 1, ..., p-1$ となり, p が小さい場合には良いが, 実用的な暗号に用いられる $p$ は $p \sim 2^{160} \sim 10^{48}$程度以上なので, 実際上計算できない. これよりはましとは言え, $t\phi$(P) の値から, $t$ を逆算する効率的な方法がない (前述問題 2 ECDLP) ことに注意して欲しい.

Schoof の工夫は, P として $l$-分点 ( $l$は $p$ とは異なる素数) を用いるというものである.

**定義 4 $n$-分点** 整数 $n > 1$ に対して, $n\mathrm{P} = O$ (無限遠点) となる点 $\mathrm{P} \in E(\overline{K})$ を $n$-分点という.

$l$-分点 $\mathrm{P}_l$については 式 (1) の代わりに

$$\phi^2(\mathrm{P}_l) + (p \bmod l)\,\mathrm{P}_l = (t \bmod l)\,\phi(\mathrm{P}_l) \tag{2}$$

が成立する. つまり, $t \bmod l$ ($t$ を $l$で割った剰余) が計算出来ることになる. ここで, $l$は $p$ よりはるかに小さくとることが可能であり, 従って $t \bmod l$ の値を $0, 1, ..., l-1$ とスイープしても手間は掛からないことがポイントである.

$t$ の値の範囲は,

$$-2\sqrt{p} \le t \le 2\sqrt{p} \tag{3}$$

であることが知られている (Hasse の定理) ので, $\prod_l l > 4\sqrt{p}$ を満たすまで, 必要なだけ多くの素数 $l \neq p$ について式 (2) により $t \bmod l$ の値を計算すれば, 中国剰余算法により式 (3) を満たす $t$ が再構成できる.

## 4.2 $l$-分点利用の計算法と数式処理

$l$-分点利用の計算法は以下にみるように数式処理そのものである.

まず, $l$-分点は一般には有理点ではない. すなわち, その座標値は $K = \mathrm{GF}(p)$ の元そのものではなく, $K$の代数拡大体の元 (代数的数) である. $K$ の代数拡大体の元とは, $K$の元を係数とする多項式の根のことであり, 元そのものもそれが満たす最小次数の多項式 (最小多項式) で表される. たとえば, 有理数の体の拡大体では, $r = \sqrt{2}$

は $r^2-2=0$ の根, 虚数単位 $i$ は $i^2+1=0$ を満たす何者かとして表現される.

このような表現上で代数的数 $r$や $i$ を含む式を計算するとは, この代数的数を定義する多項式を利用して, $r^2 \to 2$ とか, $i^2 \to -1$ などと簡約を行うことである. 一般には簡約の結果 (代数的数を表す変数の多項式, 例では $r$や $i$ の多項式) が一意的になるように定義しておく必要がある. したがって, 代数的数を含む計算は多変数の多項式の演算として実行される.

**$l$-分多項式** 実際には $l$-分点の $x$ 座標の満たす多項式で, 楕円曲線の方程式の係数から計算できる多項式が計算に用いられる. この多項式を $l$-分多項式と呼ぶ.

一般に, 整数 $n>1$ に対して $n$-分多項式は楕円曲線方程式 (定義 1) の係数 $a,b$ から漸化式で計算できる. Schoof に従ってその式を書いておく.

まず, 2 変数多項式 $\Psi_n(x,y) \in K[x,y]$, $n=-1,0,1,2,...$を次の漸化式で定義する.

$$\Psi_{-1}(x,y)=-1,\ \Psi_0(x,y)=0,$$
$$\Psi_1(x,y)=1,\ \Psi_2(x,y)=2y,$$
$$\Psi_3(x,y)=3x^4+6ax^2+12bx-a^2,$$
$$\Psi_4(x,y)=4y(x^6+5ax^4+20bx^3-5a^2x^2 -4abx-8b^2-a^3),$$
$$\Psi_{2n+1}(x,y)=\Psi_{n+2}\Psi_n^3-\Psi_{n+1}^3\Psi_{n-1} \quad \text{for } n\geq 2.$$
$$\Psi_{2n}(x,y)=\Psi_n(\Psi_{n+2}\Psi_{n-1}^2-\Psi_{n-2}\Psi_{n+1}^2)/(2y) \quad \text{for } n\geq 3,$$

$\Psi_n$ の式で $y^2 \to x^3+ax+b$ と置き換え, 結果を $\Psi'_n$ とすると, $\Psi'_n$ は $n$ が奇数の場合は $K[x]$ の元, $n$ が偶数の場合は $yK[x]$ の元 となる. この $\Psi'_n$ を用いて, $n$-分多項式 $f_n(x)$ はつぎのように計算さ れる.

$$f_n(x) = \Psi'_n(x,y) \text{ if } n \text{ is odd and } n\neq p,$$
$$f_n(x) = \Psi'_n(x,y)/y \text{ if } n \text{ is even and } n\neq p.$$

ここで, n-分多項式の次数は,

$$\deg f_n = (n^2-1)/2 \text{ if } n \text{ is odd and } n\neq p,$$
$$\deg f_n = (n^2-4)/2 \text{ if } n \text{ is even and } n\neq p$$

となる.

P $=(x,y)$ は 2-分点ではないとして, P が $n$ 分点であることと $f_n(x)=0$ であることとは同値である.

素数 $l(>2)$ に対して $f_l(x)$ は最小多項式とは限らないが, $l$-分点の $x$ 座標を定義する多項式として使用できる.

式 (2) で $l$-分点の関係式から $t \bmod l$を計算する算法はつぎのとおりとなる.

**アルゴリズム 5**

入力: 楕円曲線方程式のパラメータ $a,b$;
$l$-分点 P$=(x,y)\in E(\overline{K})$.
出力: 式 (1) の $t$ に対応する $t \bmod l$.
*(1)* $\phi(P),\phi^2(\mathrm{P})=\phi(\phi(\mathrm{P}))$ を計算する.
*(2)* $k:=p \bmod l$ を計算する.
*(3)* $\phi^2(\mathrm{P})+k\mathrm{P}=\tau\phi(\mathrm{P})$ となる $\tau$を $0<\tau<l-1$ の中から捜す.
*(4)* $\tau$を出力する.

この算法のなかで, 楕円曲線上の点の座標は数値 ($K=GF(p)$ の元 すなわち $0,1,...,p-1$) ではなく, $K$係数の $x,y$の多項式として表されており, 点の座標の演算は常に, $y^2 \to x^3+ax+b$ という簡約と, $(l^2-1)/2$ 次の多項式 $f_l(x)$ で剰余を取るという演算とを伴っている.

最終的に必要なものは $x$-座標であるので, Frobenius 写像$\phi(P)=(x^p,y^p)$ の計算では, $x^p \bmod f_l(x)$ を計算することになる. $x \to x^2 \to x^3 \cdots$と計算するのではなく, $x \to x^2 \to x^4 \cdots$という形で $\log p$ 回程度の多項式乗算と除算で計算する. また, $k$倍点の計算にはつぎに示す 2 倍公式と加法公式とを繰り返して使い, やはり $\log k$回程度の計算で済むように工夫する. $\tau$倍点の計算は必要な $\tau$が見つかるまで $\tau=1,2,...$が順に必要になるので, 順次計算する必要がある. $n$ 倍点の $x$ 座標 $[n\mathrm{P}]_x$も $x$ のみの有理式で表すことができる.

**加法公式** P $=(\xi_1,\eta_1)$ と Q $=(\xi_2,\eta_2)$ の和の座標の計算公式 (ただし、P≠Q)

$$[P+Q]_x = \lambda^2-\xi_1-\xi_2$$
$$[P+Q]_y = -\lambda[P+Q]_x-\nu$$

ここで, $\lambda=\frac{\eta_2-\eta_1}{\xi_2-\xi_1}$, $\nu=\eta_1-\lambda\xi_1$ である.

**2 倍公式** P $=(\xi,\eta)$ の 2 倍点の座標の公式

$$[2\mathrm{P}]_x = \frac{\xi^4-2a\xi^2-8b\xi+a^2}{4\xi^3+4a\xi+4b},$$
$$[2\mathrm{P}]_y = \eta+([2\mathrm{P}]_x-\xi).$$

ここで, $\xi,\eta$や $[2\mathrm{P}]_x,[2\mathrm{P}]_y$などの座標値は $l$-分点の座標 $x,y$の有理式で表される. 第 (3) ステップでの等号の確認の時点で分母を払い, 多項式として( $\pmod{p,f_l(x)}$) による除算はあるが, ) $x$ の多項式として両辺を比較すれば良い.

**問題点**　上記は Schoof の算法の核心部分であり，Schoof 自体さらに工夫をしているが，ここでは深入りしない．

問題は $l$分多項式 $f_l(x)$ を用いる限り，その次数が $l^2$ オーダーで大きくなることである．これは，$f_l(x)$ で剰余を求める演算の手間が $l$の増加とともに急増することを意味している．

このため，Schoof の算法の計算量は，全体として $O((\log p)^8)$ と見積もられる．

実用の暗号では $p$ として 160 ビット程度以上の素数が使用される．中国剰余算法を使用する Schoof の方法では，$l$-分点に必要な $l$は，その積が $4\sqrt{2^{160}}$を越える必要があるため，3 以上の素数が 18 個は必要であり，少なくとも $l = 71$ の計算をせねばならない．この $l$に対しては $l$分多項式の次数が 2520 次となり，(実行不可能というわけではないが) 実用的な計算というには不満がある．また，より強い安全性を望む場合に対応できない．

Atkin や Elkies らはこの点を改良し，$O((\log p)^6)$ の算法を得た．

### 4.3　Atkin-Elkies の改良とモジュラ多項式

Schoof の算法の核心は $l$-分点を利用して Frobenius 写像の trace を求めることにあった．このときに，$l$-分点の $x$-座標を $l$-分多項式の零点として扱ったが，次数が $(l^2-1)/2$ であるため $l$の増加とともに剰余演算の計算量が $l^4$オーダーで急増することになった．

Elkies の改良は，$f_l(x)$ の代わりに $f_l(x)$ の因子で 次数が $(l-1)/2$ 次の $g_l(x)$ を利用するものである．この $g_l(x)$ は $l$-分点の集合における$\phi$の固有空間を核とする同種写像の計算により求めることができ，$t \bmod l$は固有値から得られる．ただし，そのような $g_l$はほぼ 1/2 の確率で存在し，その判定にはモジュラー多項式が用いられる．

Atkin は $t \bmod l$の値そのものを求めるのではなく，ある束縛条件から $t \bmod l$値の候補を選び出す方法を考案した．Atkin のケースはちょうど Elkies のケースを補完している．

ここで，モジュラー多項式という多項式が Elkies ケースと Atkin ケースの判定に用いられる．この多項式は $p$ や楕円曲線には依存せず，データベースとしてあらかじめ計算しておくことができる．これは，次数が大きくなると係数膨張が激しく起こる多項式で，チャレンジングな数式処理計算といえる．(ただし，実用上はモジュラー多項式より係数膨張が少なく，同じ目的に使える Canonical Modular Polynomial を用いている．)

## 5　モジュラー多項式

モジュラー多項式とは楕円モジュラー関数 $j(z)$ と素数 $l$に対し，

$$\Phi_l(X,j) = (X - j(lz)) \prod_{k=0}^{l-1} (X - j(\frac{z+k}{l}))$$

で与えられる 2 変数多項式をいう．この多項式の具体的な計算は古くから行われており，数式処理の話題としても E. Kaltofen らが $l = 11$ の場合を計算している.[KY84] このモジュラー多項式 $\Phi_l(X,j)$ は整数係数であるが，$l$ が大きくなるとその係数は巨大になり，求めることが非常に困難になる．最近になって楕円曲線への応用からより大きな $l$に対する計算結果が現れて来た．

われわれは，Schoof 算法とその改良を数式処理システム Risa/Asir によって実現し，現在も改良を続けているが，この改良において大きな $l$に対する$\Phi_l(X,j)$ が必要になり，$l = 199$ まで 47 個のモジュラー多項式を計算した．実験では Newton 公式を用いた方法と Borcherds 積を用いた方法の 2 通りを試したが，より効率の良かった Newton 公式を用いた方法を紹介する．

### 5.1　対称式に関する Newton 公式

$n$ 個の変数 $X = \{x_1, x_2, \ldots, x_n\}$ による基本対称式を $t_r$ であらわす．ただし $1 \le r \le n$ であり，$t_r = \sum_\sigma x_{\sigma_1} x_{\sigma_2} \ldots x_{\sigma_r}$ とする．また，$X$ の累乗和を $u_k = \sum_{i=1}^n x_i^k = x_1^k + x_2^k + \ldots + x_n^k$　$(0 \le k)$ とする．

このとき $t_r$ と $u_k$ の間には次の漸化式が成立する:

**命題 6 (対称式に関する Newton 公式)**

$$\begin{aligned} t_1 &= u_1, \\ t_2 &= -\frac{1}{2}(u_2 - t_1 u_1), \\ t_3 &= \frac{1}{3}(u_3 - t_1 u_2 + t_2 u_1), \ldots \\ t_{l-1} &= (-1)^l \frac{1}{l-1}(u_{l-1} - t_1 u_{l-2} + \ldots + (-1)^l t_{l-2} u_1). \end{aligned}$$

したがって，累乗の値から基本対称式の値を構成できる．以下ではこの公式を単に Newton 公式と呼ぶ．

## 5.2 Newton公式によるモジュラー多項式の計算

モジュラー多項式 $\Phi_l(X,j)$ を $X$の一変数多項式

$$\Phi_l(X,j) = X^{l+1} + \sum_{k=1}^{l+1}(-1)^k s_k(j) X^{l-k+1}$$

と見なすと, 係数 $s_k(j)$ は $j$の整数係数多項式として定められる. したがって $\Phi_l(X,j)$ を求めるには, 各係数 $s_k(j)$ を求めればよい. ここで, $s_k(j)$ は 集合

$$J_l = \{j(lz), j(\frac{z}{l}), j(\frac{z+1}{l}), ..., j(\frac{z+l-1}{l})\}$$

の元による対称式として表されるのでNewton 公式を利用することができる.

$J_l$の元のうち $j(lz)$ だけは他の元と性質が異なるので別に扱うほうが好ましい. そこで, $J_l$から $j(lz)$ を除いた集合を

$$\overline{J_l} = \{j(\frac{z}{l}), j(\frac{z+1}{l}), ..., j(\frac{z+l-1}{l})\}$$

とおく. さらに, $\overline{J_l}$ による基本対称式を

$$\begin{aligned} t_1 &= \sum j\left(\frac{z+k}{l}\right) \\ t_2 &= \sum j\left(\frac{z+k}{l}\right) j\left(\frac{z+k'}{l}\right) \\ &\vdots \\ t_l &= j\left(\frac{z}{l}\right) j\left(\frac{z+1}{l}\right) \cdots j\left(\frac{z+l-1}{l}\right) \end{aligned}$$

とする. ただし $t_0 = 1$, $t_{l+1} = 0$ とおく. このとき $\{s_k\}$ と $\{t_k\}$ の関係は $s_k = j(lz) \cdot t_{k-1} + t_k$ となる.

ここで $j(z)$ の $r$ 乗を

$$j^r(z) = \sum_{m=-r}^{\infty} c_m^{(r)} q^m,$$

$\overline{J_l}$ の元たちの $r$ 乗和 $u_r$ を

$$u_r = \sum_{k=0}^{l-1} j^r\left(\frac{z+k}{l}\right) \quad (1 \le r \le l)$$

とおくと, 関係式

$$\begin{aligned} u_r &= l\sum_{n=0}^{\infty} c_{ln}^{(r)} q^n \quad (1 \le k \le l-1), \\ u_l &= l\Big(\frac{1}{q} + \sum_{n=0}^{\infty} c_{ln}^{(l)} q^n\Big). \end{aligned}$$

が成立する. つまり $j$ の累乗和の係数から $\overline{J_l}$ の $r$ 乗和 $u_r$ を求めることができる. あとは Newton 公式を用いて, $\{u_r\}$ から基本対称式 $\{t_k\}$ ならびに $\{s_k\}$ を計算すれば良い.

実際の計算においては, まず $j(q)$ を $l(l+1)$ 次まで求め, 次に $j(q)$ の累乗 $j^r(q)$ $(r = 1, 2, \ldots, l+1)$ を計算する. これらのデータをもとに, 上の手順に従って $\{u_r\}$, $\{t_k\}$, $\{s_k\}$, $\Phi_l(X,j)$ と計算していけばモジュラー多項式が得られる. $j(q)$ が $l(l+1)$ 次まで必要となるのは, $j^{l+1}(q)$ における $q^l$ の項の係数が必要となるからである.

## 5.3 モジュラー多項式の計算結果

$\Phi_l(X,j)$ のデータサイズと, それぞれの方法による計算時間を示す. 大きな $l$ に関するデータは, 紙面の都合で割愛した. 測定は FreeBSD(PentiumPro 200MHz,128MB RAM) 上の Risa/Asir で行なった.

個々の計算を調べると, $l$ にかかわらず N1 と N2 の合計で全体の 90%程度を占めている. うち N2 は $l$ の増加にともなって比率が増大し, $l = 61$ では全体の 85%以上を占めるようになる. つまり N 法の律速段階は N2 であることがわかる. これは $l(l+1)$ 次の密で, しかも係数が膨大な多項式の乗法を繰り返し計算しているからであろう. 無限積表示を用いる Borcherd 法についても実験を行なったが, Newton 公式を用いる方が勝っている.

$n$ 次の多項式の乗算に必要な計算量を $M(n)$ とする. このとき $j(q)$ を $n$ 次まで計算するには $O(n^2)$ の計算量が必要となる. Newton 公式を用いた方法では $n = l(l+1)$ なので, N1 に $O(l^4)$, N2 に $O(lM(l^2))$ かかるので, 全体の計算量は $O(\max\{l^4, lM(l^2)\})$ となる.

多項式の乗法に通常の方法を用いると, $M(n) = n^2$ であり, Newton 公式を用いる方法の計算量は $O(l^5)$ となる. しかし多項式の乗法に Karatsuba 法を用いると $M(n) = n^{\log_2 3}$ となるので, 計算量は $O(l^{1+2\log_2 3}) \sim O(l^{4.2})$ となることが期待できる. 高速乗算法を用いることで Newton 公式を用いる方法はさらに改善の余地がある.

今回の計算では Karatsuba 法[1] を用いており, 実際の計算時間も予想を反映した結果が得られた. Karatsuba 法の効果が実証されたといえるだろう. 大きな $l$ に対しては, FFT 法の利用を考えることもできるが, これについてはこれからの課題としたい.

[1] Karatsuba 法は Risa/Asir に標準では組み込まれていない. 今回の計算を行うために組み込んで実験を行った.

| $l$ | サイズ (KB) | Newton (秒) | N1 | N2 | N3 |
|---|---|---|---|---|---|
| 2 | 1 | 0.02 | — | — | — |
| 3 | 1 | 0.04 | — | — | — |
| 5 | 2 | 0.15 | — | — | — |
| 7 | 4 | 0.55 | — | — | — |
| 11 | 10 | 3.43 | 1.4 | 1.66 | 0.34 |
| 13 | 15 | 6.45 | 2.3 | 3.48 | 0.61 |
| 17 | 30 | 22.83 | 6.6 | 14.23 | 1.96 |
| 19 | 40 | 34.90 | 9.20 | 22.54 | 3.16 |
| 23 | 70 | 99.43 | 20.31 | 71.69 | 7.43 |
| 29 | 138 | 328.29 | 44.29 | 257.80 | 26.20 |
| 31 | 167 | 450.76 | 53.23 | 361.37 | 36.16 |
| 37 | 284 | 1241.1 | 115.8 | 1020.3 | 105.0 |
| 41 | 389 | 2394.3 | 172.1 | 2032.3 | 189.9 |
| 43 | 449 | 3126.2 | 202.0 | 2676.2 | 248.0 |
| 47 | 591 | 5422.8 | 326.7 | 4694.7 | 401.4 |
| 53 | 855 | 10435.8 | 486.4 | 8993.8 | 955.6 |
| 59 | 1191 | 21347.4 | 779.4 | 18777.1 | 1790.9 |
| 61 | 1319 | 26017.6 | 868.1 | 23008.2 | 2141.3 |

N1: $j$ の計算; N2: $j$ の累乗; N3:その他

以上の方法のままでも $l = 79$ 程度までは何とか計算できるが, それ以上の $l$ については現実的でなくなってくる. そこで $l > 80$ に対する Newton 公式を用いた方法では, 次のような工夫を行った. まず, $j(q)$ の累乗をそれぞれの $l$ に対して計算する必要はない[2] ので, あらかじめ適当な大きさの自然数 $n$ に対し, $\{j^i(q)\}$ $(i = 1, 2, \ldots, n)$ を計算しておき, $l < n$ であるモジュラー多項式 $\Phi_l(X, j)$ の計算は, このデータを参照するようにした. また累乗計算は並列計算が可能なので, 複数の計算機を用いて計算するようにした. たとえば, $l = 113$ での $j(q)$ の累乗計算は, 一回の積に 1〜3 時間程度を要するため, これらの工夫の効果は大きかった.

### 5.4 その他の数式処理技法

これまで見て来たように楕円曲線暗号の設計には, 多項式や有理式の乗除算が本質的に現れる. これには, 楕円曲線上の点の表現に起因するものばかりでなく, より良い算法には欠かせない数学的対象物の取扱が生じるからである. モジュラー多項式はその一例に過ぎない.

実は, 楕円曲線暗号は GF($p$) 上のものばかりでなく, GF($2^k$) (これも有限体である) 上でも構成することができる. この場合には設計ばかりなく, その運用時の演算 (暗号化や復号, 署名演算など) にも代数拡大体としての取扱が必要なため, 多項式の演算が必要になる. (多項式の形式を取らない表現法もある. ) また, 暗号を攻撃するために暗号の性質を解析し, 実験で実証するには, やはり数学上の対象が具体的かつ容易に扱える数式処理が重要な役割を果たすと考えられる. この良い例が下山 [Shim98] の DES 解読に見られる.

**高速乗算法—Karatsuba, FFT** 数式の演算において, 多項式の演算とりわけ乗算と除算は基本的で重要である. また, さらに下位部品に目を向ければ最も基本的なものは係数である多倍長整数の乗除算の高速化が非常に重要である.

本稿では紹介仕切れなかったが, これらの基本演算としての整数乗算と多項式乗算では, Karatsuba 法や FFT(浮動小数点計算ではない) が有効である.

例えば, Schoof 法の場合, $O((\log p)^8)$ というのは, 高速乗算法を用いない場合であって, 高速乗算法として Katatsuba 法を用いれば, $O((\log p)^{6.8})$ に, FFT では $O((\log p)^{5+\epsilon})$ になる. また, Elkies-Atkin の改良法を採り入れた SEA 法に Katasuba 法を適用すれば $O((\log p)^{5.2})$, FFT では $O((\log p)^{4+\epsilon})$ まで改善されることが実験でも確認されている.

数式処理システム Risa/Asir では最近 多項式乗算の Karatsuba 法および FFT をインプリメントし, 通常法や karatsuba 法, FFT の性能のクロスオーバ点などを詳細に解析した. [Kon98]

**三角積** 多項式の積の指定された次数以下の項のみを計算したい場合に用いると効果的な技法である. [IN98] これは, 多項式 $f(x), g(x)$ の積を $x^m$で除した剰余, $f(x)g(x) \bmod x^m$を高速に計算する. 要点は剰余に必要となる組合せのみを計算することであるが, 部分的には従来の高速化算法が適用可能であることと, 並列化計算が可能であるという特徴を併せ持っている.

このような計算が必要となる例として前節で詳述したモジュラー多項式の計算がある. 実際に $\Phi_{113}$の計算に三角積を応用したケースでは, Karatsuba 法の約 2 倍の速度で計算できた.

三角積はより一般的な剰余計算 $f(x)g(x) \bmod n(x)$ にも応用できる. 代数拡大体である有限体 GF($p^r$) の計算はまさしくそのような形式を多用するので, そのような有限体の計算の高速化に役立つと期待できる.

## 6 あとがき

インターネット時代のセキュリティ問題の基本要素技術に計算機代数 (数式処理) が有効に応用されていることを紹介した. ディジタルで, しかも代数的な数学構造をもった対象を扱うには数式処理がまさに最適である.

[2]例えば $l = 13$ の計算に使用したべき乗のデータは $l = 11$ の計算にも用いることができる.

すぐに計算量的な限界が見えがちで，科学計算では敬遠されてきたきらいがある数式の記号的処理も，計算機資源 (CPU およびメモリ) の高性能大規模化に助けられて，具体的実用問題に対処できるようになってきたと考える．とくに，今回は紹介しなかったが，従来数値的にも難問であった多変数の連立高次方程式の解法技術は Gröbner 基底計算の実用化で市民権を得つつある．数式処理の基本演算の強化により新しい応用分野が暗号以外にも拓けて行くことを期待している．

## Risa/Asir について

Risa/Asir とは，富士通研究所でわれわれのグループが開発している数式処理システムであり，プラットフォームは UNIX と Windows である．Risa/Asir は計算ライブラリである Risa と，C 言語風のユーザ言語と dbx 風のデバッガを持ったインタフェイス部分である Asir から構成されている．現在も汎用システムを目指して機能を拡張中である．特に，多項式の因数分解とグレブナ基底の計算に力を注いでおり，現時点では他のシステムよりも優れている (と信じている)．また，フランスのボルドー大学が開発したライブラリ"PARI"が組み込まれているので，このライブラリを使っての各種数値実験を簡単に行うこともできる．Risa/Asir はフリーで配布されており，

`ftp : //endeavor.fujitsu.co.jp/pub/isis/asir/`

から入手することができる[3]．Risa/Asir に関する資料としては，上の ftp 内のファイル，および書籍

齋藤，竹島，平野共著，日本で生まれた数式処理ソフトーリサアジール ガイドブック，SEG 出版 (1998)

または (一般には入手しにくいが)

Masayuki Noro,Taku Takeshima, "High-Quality Computing of Polynomial Problems by Risa/Asir", FUJITSU Scientific and Technical Journal, Vol.32, No.2,1996

を参照されたい．また京都大学数理解析研究所から出版されている講究録の「数式処理における理論とその応用の研究」シリーズには，Risa/Asir による各種の計算例が報告されている．

[3]ただし UNIX 用はパスワードを取得する必要がある．

## 参考文献


[IKNY98a] T. Izu, J. Kogure, M. Noro and K. Yokoyama, Efficient Implementation of Schoof's Algorithm, *Proc. of the Int. Conf. on the Theory and Application of Cryptography and Information Secrity—ASIACRYPT'98*, LNCS 1514, pp.66-79, (1989)

[IKNY98b] 伊豆哲也，小暮淳，野呂正行，横山和弘，スクーフアルゴリズムによる安全な楕円曲線の生成，コンピュータセキュリティシンポジウム'98，(1998).

[IN98] 伊豆哲也，野呂正行，Fast Remainder Calculation in Polynomial Multiplication，アルゴリズム研究会 研究報告 64-6, (1998).

[INY98] 伊豆哲也，野呂正行，横山和弘，モジュラー多項式の計算，SCIS'98 予稿，(1998).

[Kon98] 紺谷拓弥，野呂正行，多項式乗算のさまざまなアルゴリズムの比較，数理解析研究所研究集会 数式処理とその応用 予稿，(1998).

[KY84] E.Kaltofen, N.Yui, "ON THE MODULAR EQUATION OF ORDER 11", *Proc. of the 1984 MACSYMA User's Conference*, pp.472-485,(1997).

[Shim98] T. Shimoyama, T. Kaneko, "Quadratic Relation of S-box and Its Application to the Linear Attack of Full Round DES", *Proc. of the 18th Annual Int. Cryptology Conf., Advances in Cryptology—CRYPTO'98*, LNCS 1462, pp.200-211,(1998)