(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4752313号
(P4752313)
(45)発行日　平成23年8月17日(2011.8.17)　(24)登録日　平成23年6月3日(2011.6.3)

(51)Int.Cl.　F I
G09C 1/00 (2006.01)　G09C 1/00 620A
G09C 1/00 650A

請求項の数 19　(全 84 頁)

(21)出願番号　特願2005-119587(P2005-119587)
(22)出願日　平成17年4月18日(2005.4.18)
(65)公開番号　特開2006-227562(P2006-227562A)
(43)公開日　平成18年8月31日(2006.8.31)
審査請求日　平成20年3月6日(2008.3.6)
(31)優先権主張番号　特願2004-287166(P2004-287166)
(32)優先日　平成16年9月30日(2004.9.30)
(33)優先権主張国　日本国(JP)
(31)優先権主張番号　特願2005-15071(P2005-15071)
(32)優先日　平成17年1月24日(2005.1.24)
(33)優先権主張国　日本国(JP)

(73)特許権者　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100093241
弁理士　宮田　正昭
(74)代理人　100101801
弁理士　山田　英治
(74)代理人　100086531
弁理士　澤田　俊夫
(72)発明者　北村　出
東京都品川区北品川６丁目７番３５号　ソニー株式会社内
(72)発明者　堅木　雅宣
東京都品川区北品川６丁目７番３５号　ソニー株式会社内

最終頁に続く

(54)【発明の名称】暗号処理演算方法、および暗号処理装置、並びにコンピュータ・プログラム

(57)【特許請求の範囲】
【請求項１】
暗号処理装置において、超楕円曲線暗号に基づく暗号処理演算を実行する暗号処理演算方法であり、
前記暗号処理装置の演算実行部が、
超楕円曲線Ｃ上の点Ｐ＝（ｘ、ｙ）の関係式、
$y^2+h(x)y=f(x)$
ただし、
$f(x)=x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$
$h(x)=h_2x^2+h_1x+h_0$、
ただし、f_4，f_3，f_2，f_1，f_0，h_2，h_1，h_0は、超楕円曲線の定義パラメータｆ（ｘ），ｈ（ｘ）の係数、
によって表現される超楕円曲線の因子Ｄに対するスカラー倍算の演算において、演算処理として、１／２倍算を含む演算を実行する演算ステップを有し、
前記演算ステップは、
入力因子を、$D_1=(U_1, V_1)$、出力因子を、$D_2=(U_2, V_2)$とする因子Ｄの２倍算アルゴリズムの逆演算を実行するとともに、
未知数k_0，k_1を含む一次多項式$k(x)=k_1x+k_0$を定義し、
前記２倍算アルゴリズムにおける計算過程において前記入力因子D_1の要素V_1の更新値として算出されるV_1’を、

$V_1' = h + (k_1 x + k_0) U_2 + V_2$として表現し、
ただし、$h = h(x)$、
前記２倍算アルゴリズムの逆演算の実行過程において得られる前記未知数k_0，k_1に関する２次方程式を解いて、解k_0，k_0'、k_1，k_1'を算出し、算出解の中から正答解を選択して前記２倍算アルゴリズムの逆演算を正しく実行することで超楕円曲線の因子Ｄの１／２倍算を行う暗号処理演算方法。

【請求項２】
前記演算ステップは、
種数２、標数２のランダムパラメータを持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする請求項１に記載の暗号処理演算方法。

【請求項３】
前記演算ステップは、
種数２、標数２の$h(x) = x^2 + x + h_0$，$f_4 = 0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする請求項１に記載の暗号処理演算方法。

【請求項４】
前記演算ステップは、
種数２、標数２の$h(x) = x^2 + h_1 x + h_0$，$f_4 = 0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする請求項１に記載の暗号処理演算方法。

【請求項５】
前記演算ステップは、
種数２、標数２の$h(x) = x$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする請求項１に記載の暗号処理演算方法。

【請求項６】
前記暗号処理演算方法は、
前記演算実行部が、
予め固定された因子Ｄについての［$1/2^i D$］計算値に基づいて、k_1，k_1'，（k_0，k_0'）のいずれが正しいかを記録したテーブルを参照するテーブル参照ステップを有し、
ただし、ｉは、任意の整数であり、１／２倍算の繰り返し回数を示す、
前記演算ステップは、
前記テーブルの参照に基づく判定処理によって、１／２倍算の計算量を削減した演算処理として実行することを特徴とする請求項１に記載の暗号処理演算方法。

【請求項７】
前記演算ステップは、
超楕円曲線の因子Ｄの１／２倍算における、
入力：$D_2 = (U_2, V_2)$、
出力：$D_1 = (U_1, V_1) = [1/2] D_2$、
ただし、$U_i(x) = x^2 + u_{i1} x + u_{i0}$，$V_i(x) = v_{i1} x + v_{i0}$，$gcd(h, U_i) = 1$，$i = 1, 2$、
u_{i1}，u_{i0}，v_{i1}，v_{i0}，は超楕円曲線の因子の表現するためのパラメータ、
とした１／２倍算の演算アルゴリズムに基づいて導出される以下の関係式、
$1/k_1 = h_2 + k_1 u_{21}$
を適用し、逆元演算を行なうことなく乗算および加算処理によって逆元$1/k_1$の値を算出するステップを含むことを特徴とする請求項１に記載の暗号処理演算方法。

【請求項８】
前記暗号処理演算方法は、

前記演算実行部が、
入力：$D_2=(U_2, V_2)$、
出力：$D_1=(U_1, V_1)=[1/2]D_2$、
ただし、$U_i(x)=x^2+u_{i1}x+u_{i0}$，$V_i(x)=v_{i1}x+v_{i0}$，$gcd(h, U_i)=1$，$i=1, 2$、
u_{i1}，u_{i0}，v_{i1}，v_{i0}，は超楕円曲線の因子の表現するためのパラメータ、
とした１／２倍算演算アルゴリズムにおいて、$1/u_{21}$を入力値として適用しない設定を持つアルゴリズムに従った演算を実行することを特徴とする請求項１に記載の暗号処理演算方法。

【請求項９】
前記暗号処理演算方法は、
前記演算実行部が、
種数２、標数２の$h(x)=x^2+h_1x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算を実行する演算方法であり、
前記演算ステップは、
算出済みの値として$1/h_1^2$を入力値として設定し、
逆元$1/h_1^2$を算出する処理を実行することなく、算出済みの入力値$1/h_1^2$を適用するステップを含むことを特徴とする請求項１に記載の暗号処理演算方法。

【請求項１０】
超楕円曲線暗号に基づく暗号処理演算を実行する暗号処理装置であり、
超楕円曲線Ｃ上の点Ｐ＝（ｘ、ｙ）の関係式、
$y^2+h(x)y=f(x)$
ただし、
$f(x)=x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$
$h(x)=h_2x^2+h_1x+h_0$、
ただし、f_4，f_3，f_2，f_1，f_0，h_2，h_1，h_0は、超楕円曲線の定義パラメータｆ（ｘ），ｈ（ｘ）の係数、
によって表現される超楕円曲線の因子Ｄに対するスカラー倍算の演算において、演算処理として、１／２倍算を含む演算を実行する演算実行部を有し、
前記演算実行部は、
入力因子を、$D_1=(U_1, V_1)$、出力因子を、$D_2=(U_2, V_2)$とする因子Ｄの２倍算アルゴリズムの逆演算を実行するとともに、
未知数k_0，k_1を含む一次多項式$k(x)=k_1x+k_0$を定義し、
前記２倍算アルゴリズムにおける計算過程において前記入力因子D_1の要素V_1の更新値として算出されるV_1'を、
$V_1'=h+(k_1x+k_0)U_2+V_2$として表現し、
ただし、$h=h(x)$、
前記２倍算アルゴリズムの逆演算の実行過程において得られる前記未知数k_0，k_1に関する２次方程式を解いて、解k_0，k_0'、k_1，k_1'を算出し、算出解の中から正答解を選択して前記２倍算アルゴリズムの逆演算を正しく実行することで超楕円曲線の因子Ｄの１／２倍算を行う暗号処理装置。

【請求項１１】
前記演算実行部は、
種数２、標数２のランダムパラメータを持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であることを特徴とする請求項１０に記載の暗号処理装置。

【請求項１２】
前記演算実行部は、
種数２、標数２の$h(x)=x^2+x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であること

を特徴とする請求項１０に記載の暗号処理装置。
【請求項１３】
前記演算実行部は、
種数２、標数２のｈ（ｘ）＝ｘ^{2}＋ｈ$_{1}$ｘ＋ｈ$_{0}$，ｆ$_{4}$＝０をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であることを特徴とする請求項１０に記載の暗号処理装置。
【請求項１４】
前記演算実行部は、
種数２、標数２のｈ（ｘ）＝ｘをパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であることを特徴とする請求項１０に記載の暗号処理装置。
【請求項１５】
前記暗号処理装置は、
予め固定された因子Ｄについての［１／２^{i}Ｄ］計算値に基づいて、ｋ$_{1}$，ｋ$_{1}$’，（ｋ$_{0}$，ｋ$_{0}$’）のいずれが正しいかを記録したテーブルを格納した記憶部を有し、
ただし、ｉは、任意の整数であり、１／２倍算の繰り返し回数を示す、
前記演算実行部は、
前記テーブルの参照に基づく判定処理によって、１／２倍算の計算量を削減した演算処理を実行する構成であることを特徴とする請求項１０に記載の暗号処理装置。
【請求項１６】
前記演算実行部は、
超楕円曲線の因子Ｄの１／２倍算における、
入力：Ｄ$_{2}$＝（Ｕ$_{2}$，Ｖ$_{2}$）、
出力：Ｄ$_{1}$＝（Ｕ$_{1}$，Ｖ$_{1}$）＝［１／２］Ｄ$_{2}$、
ただし、Ｕ$_{i}$（ｘ）＝ｘ^{2}＋ｕ$_{i1}$ｘ＋ｕ$_{i0}$，Ｖ$_{i}$（ｘ）＝ｖ$_{i1}$ｘ＋ｖ$_{i0}$，ｇｃｄ（ｈ，Ｕ$_{i}$）＝１，ｉ＝１，２、
ｕ$_{i1}$，ｕ$_{i0}$，ｖ$_{i1}$，ｖ$_{i0}$，は超楕円曲線の因子の表現するためのパラメータ、
とした１／２倍算の演算アルゴリズムに基づいて導出される以下の関係式、
１／ｋ$_{1}$＝ｈ$_{2}$＋ｋ$_{1}$ｕ$_{21}$
を適用し、逆元演算を行なうことなく乗算および加算処理によって逆元１／ｋ$_{1}$の値を算出する構成であることを特徴とする請求項１０に記載の暗号処理装置。
【請求項１７】
前記演算実行部は、
入力：Ｄ$_{2}$＝（Ｕ$_{2}$，Ｖ$_{2}$）、
出力：Ｄ$_{1}$＝（Ｕ$_{1}$，Ｖ$_{1}$）＝［１／２］Ｄ$_{2}$、
ただし、Ｕ$_{i}$（ｘ）＝ｘ^{2}＋ｕ$_{i1}$ｘ＋ｕ$_{i0}$，Ｖ$_{i}$（ｘ）＝ｖ$_{i1}$ｘ＋ｖ$_{i0}$，ｇｃｄ（ｈ，Ｕ$_{i}$）＝１，ｉ＝１，２、
ｕ$_{i1}$，ｕ$_{i0}$，ｖ$_{i1}$，ｖ$_{i0}$，は超楕円曲線の因子の表現するためのパラメータ、
とした１／２倍算演算アルゴリズムを実行する構成であり、１／ｕ$_{21}$を入力値として適用しない設定を持つアルゴリズムに従った演算を実行する構成であることを特徴とする請求項１０に記載の暗号処理装置。
【請求項１８】
前記演算実行部は、
種数２、標数２のｈ（ｘ）＝ｘ^{2}＋ｈ$_{1}$ｘ＋ｈ$_{0}$，ｆ$_{4}$＝０をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算を実行する構成を有し、
算出済みの値として１／ｈ$_{1}$^{2}を入力値とし、逆元１／ｈ$_{1}$^{2}を算出する処理を実行することなく、算出済みの入力値１／ｈ$_{1}$^{2}を適用した演算を実行する構成であることを特徴とする請求項１０に記載の暗号処理装置。
【請求項１９】
超楕円曲線暗号に基づく暗号処理演算をコンピュータ上で実行させるコンピュータ・プ

ログラムであり、
超楕円曲線Ｃ上の点Ｐ＝（ｘ、ｙ）の関係式、
$y^2+h(x)y=f(x)$
ただし、
$f(x)=x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$
$h(x)=h_2x^2+h_1x+h_0$、
ただし、f_4，f_3，f_2，f_1，f_0，h_2，h_1，h_0は、超楕円曲線の定義パラメータｆ（ｘ），ｈ（ｘ）の係数、
によって表現される超楕円曲線の因子Ｄに対するスカラー倍算の演算において、演算処理として、１／２倍算を含む演算を実行する演算ステップを実行させ、
前記演算実行ステップにおいて、
入力因子を、$D_1=(U_1, V_1)$、出力因子を、$D_2=(U_2, V_2)$とする因子Ｄの２倍算アルゴリズムの逆演算を実行するとともに、
未知数k_0，k_1を含む一次多項式$k(x)=k_1x+k_0$を定義し、
前記２倍算アルゴリズムにおける計算過程において前記入力因子D_1の要素V_1の更新値として算出されるV_1’を、
$V_1'=h+(k_1x+k_0)U_2+V_2$として表現し、
ただし、ｈ＝ｈ（ｘ）、
前記２倍算アルゴリズムの逆演算の実行過程において得られる前記未知数k_0，k_1に関する２次方程式を解いて、解k_0，k_0’、k_1，k_1’を算出し、算出解の中から正答解を選択して前記２倍算アルゴリズムの逆演算を正しく実行することで超楕円曲線の因子Ｄの１／２倍算を行わせるコンピュータ・プログラム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、暗号処理演算方法、および暗号処理装置、並びにコンピュータ・プログラムに関する。さらに詳細には、超楕円曲線暗号におけるスカラー倍算の高速化を実現する暗号処理演算方法、および暗号処理装置、並びにコンピュータ・プログラムに関する。
【背景技術】
【０００２】
昨今、ネットワーク通信、電子商取引の発展に伴い、通信におけるセキュリティ確保が重要な問題となっている。セキュリティ確保の１つの方法が暗号技術であり、現在、様々な暗号化手法を用いた通信が行なわれている。
【０００３】
例えばＩＣカード等の小型の装置中に暗号処理モジュールを埋め込み、ＩＣカードと、データ読み取り書き込み装置としてのリーダライタとの間でデータ送受信を行ない、認証処理、あるいは送受信データの暗号化、復号化を行なうシステムが実用化されている。
【０００４】
暗号処理を実行する例えばＩＣカードは、例えば駅の改札などの様々なゲート、あるいはショッピングセンターなどで盛んに利用されるようになっており、小型化および処理の迅速化の要求が厳しくなってきている。
【０００５】
暗号化方式には、大別して共通鍵方式、公開鍵方式がある。共通鍵方式は、対称暗号方式ともよばれ、発信者、受信者の双方で共通の鍵を保有する。共通鍵方式の代表的な方法として、ＤＥＳ（Data Encryption Standard）がある。ＤＥＳアルゴリズムの特徴は、暗号化と復号化とをほぼ同じアルゴリズムで実行可能なことである。
【０００６】
この共通鍵暗号に対して、発信者と受信者の鍵を異なるものとした構成が公開鍵方式または非対称暗号方式である。公開鍵暗号方式では、暗号化、復号化に共通の鍵を用いる共通鍵暗号方式と異なり、秘密に保つ必要のある秘密鍵は、特定の１人が持てばよいため鍵

の管理において有利である。ただし、公開鍵暗号方式は共通鍵暗号方式に比較してデータ処理速度が遅く、一般には、秘密鍵の配送、デジタル署名等のデータ量の少ない対象に多く用いられている。公開鍵暗号方式の代表的なものにはＲＳＡ（Rivest-Shamir-Adleman）暗号や、楕円曲線暗号（ＥＣＣ：Elliptic Curve Cryptography）が知られている。

【０００７】

楕円曲線暗号（Elliptic Curve Cryptography）は、素体上の楕円曲線$y^2=x^3+ax+b$（$4a^3+27b^2\neq 0$）や、２の拡大体上の楕円曲線$y^2+xy=x^3+ax^2+b$（$b\neq 0$）などを用いる。これらの曲線上の点に無限遠点（Ｏ）を加えた集合は、加法に関して有限群をなし、無限遠点（Ｏ）はその単位元となる。以下、この有限群上の点の加法を＋で表す。この有限群上の異なる２点Ｐ，Ｑの加算Ｐ＋Ｑを「点の加算」、点Ｐと点Ｐの加算Ｐ＋Ｐ＝２Ｐを「点の２倍算」と呼ぶ。また、点Ｐをｋ回加算した点Ｐ＋Ｐ＋…＋Ｐ＝ｋＰを求める演算を「点のスカラー倍算」と呼ぶ。

【０００８】

点のスカラー倍算は、点の加算、および点の２倍算を用いて構成できることが知られている。素体上の楕円曲線や２の拡大体上の楕円曲線上のアフィン座標系（ｘ，ｙ）や射影座標（Ｘ，Ｙ，Ｚ）における点の加算法、点の２倍算法、および点のスカラー倍算法は、ＩＥＥＥ　P1363/D13 Standard Specifications for Public Key Cryptographyに記されている。

【０００９】

楕円曲線暗号を一般化した方式として、Koblitz、Cantorによって提案された超楕円曲線暗号（ＨＥＣＣ：Hyper Elliptic Curve Cryptography）方式がある。超楕円曲線暗号方式については、例えば非特許文献１、非特許文献２に記載されている。

【００１０】

楕円曲線暗号では有限体Ｆｑ上で定義される楕円曲線上の点をＰとし、スカラー倍算した点ｋＰ（$k\in Z$）をＱとすると、Ｑからｋを求める問題は離散対数問題に帰着できる。一方、超楕円曲線暗号では点の形式的和である因子（ｄｉｖｉｓｏｒ）をD_1とし、スカラー倍算ｋD_1で定義される因子をD_2とすると、D_2からｋを求める問題は超楕円曲線におけるヤコビ多様体上の離散対数問題として公開鍵暗号として用いることができる。

【００１１】

超楕円曲線では曲線を特徴づける値は種数（ｇｅｎｕｓ）ｇである。ｐを素数、ｎを正の整数、$q=p^n$とする。このとき有限体Ｆｑ上で定義される種数ｇの超楕円曲線Ｃは以下の方程式、

$y^2+h(x)y=f(x)$

で定義される。ここで、ｈ（ｘ），ｆ（ｘ）$\in Fq[x]$，ｆ（ｘ）は、次数２ｇ＋１のｍｏｎｉｃ多項式である。

【００１２】

超楕円曲線Ｃ上の点Ｐ＝（ｘ、ｙ）に対して共役点（ｏｐｐｏｓｉｔｅ　ｐｏｉｎｔ）は、－Ｐ＝（ｘ、ｙ＋ｈ（ｘ））として定義される。また、Ｐ＝－Ｐである点を分岐点（ｒａｍｉｆｉｃａｔｉｏｎ　ｐｏｉｎｔ）と呼ぶ。

【００１３】

超楕円曲線暗号の定義体の演算サイズ（ビット長）は楕円曲線暗号と同等の安全性を仮定した場合、楕円曲線の定義体の演算サイズに比べて１／ｇに小さくなることが知られている。演算サイズが小さいことは実装上メリットがあり、超楕円曲線暗号の利点の一つして挙げられる。

【００１４】

次に、超楕円曲線暗号の基本事項について説明する。前述したように、超楕円曲線暗号では点の形式的和である因子（ｄｉｖｉｓｏｒ）をD_1とし、スカラー倍算ｋD_1で定義される因子をD_2とすると、D_2からｋを求める問題は超楕円曲線におけるヤコビ多様体上の離散対数問題として公開鍵暗号として用いることができる。

【００１５】

ここで、因子（ｄｉｖｉｓｏｒ）は有理点の形式和であり、以下の形式で表現することができる。

【数１】

$$D=\sum_i m_i P_i$$

【００１６】

また、半被約因子（ｓｅｍｉ　ｒｅｄｕｃｅｄ　ｄｉｖｉｓｏｒ）は、以下の形式で表現することができる。

【数２】

$$D=\sum_i m_i P_i-(\sum_i m_i)P_\infty,\quad m_i\geq 0$$

ただし、P_i＝（x_i、y_i）かつｉ≠ｊに対して$P_i \neq P_j$である。

【００１７】

また、Σm_iをＤのウェイト（ｗｅｉｇｈｔ）と呼ぶ。さらにウェイト（ｗｅｉｇｈｔ）が種数ｇ以下である半被約因子を被約因子（ｒｅｄｕｃｅｄ　ｄｉｖｉｓｏｒ）と呼ぶ。

【００１８】

超楕円曲線のヤコビ（Ｊａｃｏｂｉ）多様体上の任意の半被約因子Ｄは下記の多項式Ｕ、Ｖ∈Ｆｑ［ｘ］を用いてＤ＝（Ｕ、Ｖ）として表現できる。これをマンホード（Ｍｕｍｆｏｒｄ）表現と呼ぶ。なお、マンホード（Ｍｕｍｆｏｒｄ）表現については、例えば非特許文献３に記載されている。

【数３】

$$U=\prod(x-x_i)^{m_i}$$

$$V(x_i)=y_i$$

$$V(x)^2+V(x)h(x)-f(x)\equiv 0 \mod U(x),\quad \deg V<\deg U$$

【００１９】

種数（ｇｅｎｕｓ）２の任意の被約因子Ｄはマンホード（Ｍｕｍｆｏｒｄ）表現を用いると、有限体Ｆｑ上の元を係数に持つ２次以下の多項式の組、すなわち、
（Ｕ、Ｖ）＝（$x^2+u_1x+u_0$、v_1x+v_0）、または、
（Ｕ、Ｖ）＝（$x+x_0$、y_0）
として表現することができる。
また、零元は、
（Ｕ，Ｖ）＝（１，０）＝Ｏ
として表現される。

【００２０】

次に、超楕円曲線暗号で用いられる因子のスカラー倍算について説明する．因子のスカラー倍算は加算アルゴリズムと呼ばれる因子の加算と２倍算組み合わせで計算することができる。以下では主要な加算アルゴリズムについて説明する。

【００２１】

最初に提案された実用的なアルゴリズムはＣａｎｔｏｒのアルゴリズムである。このアルゴリズムについては、例えば非特許文献１、非特許文献２に記載されている。このアルゴリズムはあらゆる種数の超楕円曲線上の因子に対して適用可能なアルゴリズムであるが、楕円曲線に比べてアルゴリズムが複雑で計算量が多いことが欠点である。

【００２２】

Ｈａｒｌｅｙは種数２の超楕円曲線に限定し、因子のｗｅｉｇｈｔの場合分けをおこない、それぞれの場合で最適化を行うことでより計算量の少ないアルゴリズムを提案した。この研究を受けて超楕円曲線暗号（ＨＥＣＣ）における演算アルゴリズムの改良、拡張の研究が近年盛んになっている。

【００２３】

（ア）Ｈａｒｌｅｙのアルゴリズムは定義体を素体、種数２の曲線、因子の表現はＭｕｍｆｏｒｄ表現を用いている。このアルゴリズムの計算量の改善の研究例として、例えば、非特許文献４、非特許文献５、非特許文献６などが挙げられる。
（イ）さらに、定義体を２の拡大体の場合について拡張した処理例について、非特許文献７、非特許文献８に報告されている。
（ウ）また、因子の表現に拡張Ｍｕｍｆｏｒｄ表現、Ｗｅｉｇｈｔｅｄ　Ｃｏｏｒｄｉｎａｔｅを用いることで計算量を削減した研究として非特許文献１１、非特許文献１２、非特許文献６、非特許文献１３がある。

【００２４】

Ｈａｒｌｅｙアルゴリズムの処理について、図１を参照して説明する。なお、本発明は、標数２の有限体上定義された種数２の超楕円曲線に関するものであり、以下の説明では、曲線の種数を２、定義体の標数を２とする。

【００２５】

図１（Ａ）は、種数２の場合の因子の加算D_1+D_2の処理例を示した図である。なお、因子D_1、D_2は、それぞれD_1＝（U_1、V_1）、D_2＝（U_2、V_2）とする。まず、因子のウェイト（ｗｅｉｇｈｔ）の値によって場合分けが行われる。すなわち、［D_1+D_2］の各ウェイト（ｗｅｉｇｈｔ）の値によって、
（１）ｗｅｉｇｈｔ２＋ｗｅｉｇｈｔ２
（２）ｗｅｉｇｈｔ２＋ｗｅｉｇｈｔ１
（３）例外処理１
の場合分けが行なわれる。

【００２６】

次にｗｅｉｇｈｔ２同士の加算、すなわち（１）ｗｅｉｇｈｔ２＋ｗｅｉｇｈｔ２の場合、因子D_1＝（U_1、V_1）、D_2＝（U_2、V_2）について、最大公約数ｇｃｄ（U_1、U_2）＝１であれば、２つの因子D_1＝（U_1、V_1）、D_2＝（U_2、V_2）は、互いに同一または共役点（ｏｐｐｏｓｉｔｅ　ｐｏｉｎｔ）を含まない。この場合、
（１ａ）ＨａｒｌｅｙＡＤＤ、

すなわちハーレーアルゴリズムに従った加算処理を行う。（１ａ）ＨａｒｌｅｙＡＤＤの処理は例えば非特許文献７に示されているＭｏｓｔ　Ｆｒｅｑｕｅｎｔ　Ｃａｓｅと呼ばれる処理である。このＭｏｓｔ　Ｆｒｅｑｕｅｎｔ　Ｃａｓｅは、種数２の場合の因子の加算D_1+D_2処理において、高確率で発生するケースである。

【００２７】

この（１ａ）ＨａｒｌｅｙＡＤＤの処理は、非常に高い確率でおこる。その他の例外処理の起こる確率は非常に低い。Ｍｏｓｔ　Ｆｒｅｑｕｅｎｔ　Ｃａｓｅの条件を満たさない場合、すなわち、因子$D_1=(U_1、V_1)$、$D_2=(U_2、V_2)$について、最大公約数ｇｃｄ$(U_1、U_2)=1$を満足しない場合は、

（１ｂ）例外処理２を行う。

【００２８】

（２）ｗｅｉｇｈｔ２＋ｗｅｉｇｈｔ１の場合についても同様に、ｇｃｄ$(U_1、U_2)=1$かどうかをチェックし、ｇｃｄ$(U_1、U_2)=1$を満足する場合には、

（２ａ）ＥｘＨａｒＡＤＤ$^{2+1\to 2}$を実行し、

ｇｃｄ$(U_1、U_2)=1$を満足しない場合には、

（２ｂ）例外処理３を行う。

（３）例外処理１は、ｗｅｉｇｈｔの場合分けが上記（１），（２）以外の場合である。

【００２９】

なお、上述した種数２の加算処理のアルゴリズムの詳細については、非特許文献８に記載（Ｔａｂｌｅ１，２）されている。

【００３０】

次に、種数２の場合の２倍算の流れを図１（Ｂ）を参照して説明する。２倍算は、Ｄ＋Ｄ＝２Ｄの処理である。入力を$D_1=(U_1, V_1)$とし、出力を$D_2=[2]D_1$とする。

加算の場合と同様に因子Ｄのｗｅｉｇｈｔが

（４）ｗｅｉｇｈｔ２、

（５）ｗｅｉｇｈｔ１、

（６）ｗｅｉｇｈｔ０、

によってそれぞれ異なる処理を行う。

【００３１】

（４）ｗｅｉｇｈｔ２の場合、因子が分岐点を含むかどうかをチェックし、含まなければ、（４ａ）ＨａｒｌｅｙＤＢＬの処理を行う。因子が分岐点を含む場合は（４ｂ）例外処理６を行う。ｇｃｄ（ｈ，Ｕ１）＝１であれば、D_1は分岐点を含まない。この場合、（４ａ）ＨａｒｌｅｙＤＢＬの処理を行う。ＨａｒｌｅｙＤＢＬの処理アルゴリズムは例えば非特許文献７にＭｏｓｔ　Ｆｒｅｑｕｅｎｔ　Ｃａｓｅとして示されている。ＨａｒｌｅｙＤＢＬの処理アルゴリズム（Ａｌｇｏｒｉｔｈｍ１）を、以下に示す。

【数４】

Algorithm1 *HarleyDBL*

Input : $D_1 = (U_1, V_1)$

Output : $D_2 = (U_2, V_2)$

$U_i(x) = x^2 + u_{i1}x + u_{i0}$, $V_i(x) = v_{i1}x + v_{i0}$, $\gcd(h, U_1) = 1$

1. $U_1' \leftarrow U_1^2$
2. $S \leftarrow h^{-1}(f + hV_1 + V_1^2)/U_1 \bmod U_1$
3. $V_1' \leftarrow SU_1 + V_1$
4. $U_2' \leftarrow (f + hV_1' + V_1'^2)/U'$
5. $U_2 \leftarrow \mathrm{MakeMonic}(U_2')$
6. $V_2 \leftarrow V_1' + h \bmod U_2$
7. return $D_2 = (U_2, V_2)$

【００３２】

後述するように、この処理が非常に高い確率でおこる。その他の例外処理の起こる確率は非常に低い。上述したように、Ｍｏｓｔ　Ｆｒｅｑｕｅｎｔ　Ｃａｓｅの条件を満たさない場合は例外処理６を行う。

【００３３】

ｗｅｉｇｈｔ１の場合も同様にｇｃｄ（ｈ，Ｕ１）＝１かどうかをチェックし，（５ａ）の処理、ＥｘＨａｒＤＢＬ$^{1+1\to 2}$もしくは、（５ｂ）の処理である例外処理７を行う。ＥｘＨａｒＤＢＬ$^{1+1\to 2}$のアルゴリズムについては、非特許文献８［４．１２．（ａ）］に示されている。

【００３４】

ＨａｒｌｅｙＡＤＤ，ＨａｒｌｅｙＤＢＬは、上述のようにｍｏｓｔ　ｆｒｅｑｕｅｎｔ　ｃａｓｅと呼ばれ、ランダムに因子を発生させて加算または２倍算を行うと、非常に高い確率でＨａｒｌｅｙＡＤＤ、ＨａｒｌｅｙＤＢＬの処理になる。なお、このＨａｒｌｅｙＡＤＤ、ＨａｒｌｅｙＤＢＬが、ｍｏｓｔ　ｆｒｅｑｕｅｎｔ　ｃａｓｅとなることの説明は、例えば非特許文献１４に説明されている。

【００３５】

非特許文献１４によると、このｍｏｓｔ　ｆｒｅｑｕｅｎｔ　ｃａｅｓ以外の処理になる確率はＯ（１／ｑ）である。ここでｑは定義体の要素数であり、安全な暗号用途ではｑ^{g}が１６０ｂｉｔ程度必要になる大きな数であるので現実的にはＨａｒｌｅｙＡＤＤ、ＨａｒｌｅｙＤＢＬしか発生しない状況とみなすことができる。

【００３６】

従って超楕円曲線暗号（ＨＥＣＣ）の加算アルゴリズムをＨａｒｌｅｙアルゴリズムまたはその改良アルゴリズムを使って、例えばＩＣカードなどの暗号処理演算手段として実装する場合、

ＨａｒｌｅｙＡＤＤ、

ＨａｒｌｅｙＤＢＬ

だけを実装し、その他の確率的にほとんど起こらない複雑な例外処理についての演算を

実行しない実装とする場合も多い。この場合、例外処理については、ｗｅｉｇｈｔの場合分けが必要のないＣａｎｔｏｒのアルゴリズムを実行する構成とするなどの方法が適用される。種数が高くなるほど複雑な例外処理の負担は増すため、非特許文献９，１０ではこの実装方法について説明している。

【００３７】

次に超楕円曲線暗号（ＨＥＣＣ）アルゴリズムにおける因子のスカラー倍算について説明する。超楕円曲線暗号（ＨＥＣＣ）アルゴリズムにおいて、因子のスカラー倍算は、超楕円加算と超楕円２倍算の組み合わせで計算することができる。スカラー倍算のアルゴリズムとして基本的なｂｉｎａｒｙ法とｄｏｕｂｌｅ－ａｎｄ－ａｄｄ－ａｌｗａｙｓ法を例に挙げて説明する。

【００３８】

前述したように、楕円曲線暗号では有限体Ｆｑ上で定義される楕円曲線上の点をＰとし、スカラー倍算した点ｋＰ（ｋ∈Ｚ）をＱとすると、Ｑからｋを求める問題は離散対数問題に帰着できる。一方、超楕円曲線暗号では点の形式的和である因子（ｄｉｖｉｓｏｒ）をD_1とし、スカラー倍算kD_1で定義される因子をD_2とすると、D_2からｋを求める問題は超楕円曲線におけるヤコビ多様体上の離散対数問題として公開鍵暗号として用いることができる。

【００３９】

因子Ｄに対して、スカラー倍算（Ｄ＝ｄＤ）に適用する乗数としてのスカラー値:ｄの２進数表現を、

$d=(d_{l-1}, \cdots\cdots, d_0)$、

ただし、$d_{l-1}=1$、$d_{l-2}, \ldots, _0=1$ｏｒ０とする。

【００４０】

スカラー倍算のアルゴリズムとして基本的なｂｉｎａｒｙ法の演算アルゴリズムには、

ｂｉｎａｒｙ（ｒｉｇｈｔ－ｔｏ－ｌｅｆｔ）法

ｂｉｎａｒｙ（ｌｅｆｔ－ｔｏ－ｒｉｇｈｔ）法

がある。

【００４１】

ｂｉｎａｒｙ（ｒｉｇｈｔ－ｔｏ－ｌｅｆｔ）法は、ｄを下位ビットから見ていき、$d_i=1$の場合に、［2^i］Ｄを加算することが特徴である。ｂｉｎａｒｙ（ｒｉｇｈｔ－ｔｏ－ｌｅｆｔ）法のアルゴリズム（Ａｌｇｏｒｉｔｈｍ２）を以下に示す。

【数５】

Algorithm 2 binary(right-to-left)法

Input D_0

Output $D = dD_0$

$T \leftarrow D_0$

$D \leftarrow O$

for i *from* 0 *to* $l-1$

{

if $d_i = 1$ *then* $D \leftarrow D+T$ //加算*HarleyADD*

$T \leftarrow 2T$ //２倍算*HarleyDBL*

}

return D

【００４２】

一方、ｂｉｎａｒｙ（ｌｅｆｔ－ｔｏ－ｒｉｇｈｔ）法は、ｄを上位ビットから見ていき、毎ビットＤを２倍し、$d_i = 1$の場合に、ベースポイントを加算することが特徴である。ｂｉｎａｒｙ（ｌｅｆｔ－ｔｏ－ｒｉｇｈｔ）法のアルゴリズム（Ａｌｇｏｒｉｔｈｍ３）を以下に示す。

【数６】

Algorithm 3 binary(left-to-right)法

Input D_0

Output $D = dD_0$

$D \leftarrow D_0$

for i *from* $l-2$ *downto* 0

{

$D \leftarrow [2]D$ //２倍算*HarleyDBL*

if $d_i = 1$ *then* $D \leftarrow D+D_0$ //加算*HarleyADD*

}

return D

【００４３】

次に、ベースポイントの生成処理について説明する。スカラー倍算の計算を暗号技術で

用いる場合、入力に必要な因子D_0は、
（１）事前に決めた因子の場合、
（２）事前に決められないランダムに発生する因子の場合、
の２つのタイプに分けることができる。

【００４４】

ここで（１）事前に決めた因子の場合、の入力因子をベースポイントと呼ぶことにする。

一般的なベースポイントの生成アルゴリズムを以下に示す。

（ａ）

定義体F_q上の元をランダムにｇ個選び、ｇ個の超楕円曲線上の点P_i（ｉ＝１、…、ｇ）を生成する。

（ａ１）　ランダムに選んだ各元をｘ座標x_i（ｉ＝１…ｇ）とし、次に超楕円曲線上に乗るようにx_iに対応するｙ座標を求める。

（ｂ）

ベースポイントの因子をD_0＝（Ｕ（ｘ）、Ｖ（ｘ））とする。

（ｂ１）$U(x)=(x-x_1)(x-x_2)\cdots(x-x_g)$

（ｂ２）$V(x)=v_{g-1}x^{g-1}+v_{g-2}x^{g-2}+\cdots+v_0$の係数$v_i$を決定する。例えば生成した点が全て異なる場合、$V(x_i)=y_i$より$v_i$を求めることができる。

（ｃ）　上記アルゴリズムにより生成される因子はｗｅｉｇｈｔ　ｇの因子となる。

【００４５】

スカラー倍算の計算を暗号技術で用いる場合、入力に必要な因子D_0の生成、すなわちベースポイントの生成において、事前に決めた因子を適用する場合、上述の処理（ａ）～（ｃ）によってｗｅｉｇｈｔ　ｇのベースポイントに適用する因子を求めることができる。

【００４６】

さらに、楕円曲線暗号では、有理点の１／２倍算が提案されている。例えば、非特許文献１５、特許文献１、特許文献２に楕円曲線暗号における有理点の１／２倍算が示されている。有理点のスカラー倍算を計算する際に、加算と２倍算を用いるのではなくて、加算と１／２倍算を用いた処理を開示している。

【００４７】

楕円曲線暗号の１／２倍算は，一般に２倍算よりも高速であり、結果として１／２倍算を用いたスカラー倍算は高速となる。非特許文献１６によると、プロセッサとして、インテル社製［Ｉｎｔｅｌ　ＰｅｎｔｉｕｍＩＩＩ　８００ＭＨｚ］でソフトウェア実装を行った結果、１／２倍算は、２倍算に比べて、Ｆｑ，$q=2^{163}$の定義体では、約２．１倍高速であり、$q=2^{233}$の定義体では、約２．６倍高速であると報告されている。超楕円曲線暗号は、楕円曲線暗号の一般化であり、楕円曲線暗号で用いられている演算を超楕円曲線に拡張できる場合がある。例えば、楕円曲線暗号でｙ座標を演算で用いないため、高速な計算を行なうことの出来るＭｏｎｔｇｏｍｅｒｙ法が、非特許文献１７、１８において、超楕円曲線暗号に拡張された例が示されている。超楕円曲線暗号でも、２倍算よりも高速な１／２倍算が実現できれば、因子のスカラー倍算も、従来よりも高速に計算することが期待できる。しかしながら、従来は、この１／２倍算が知られていなかった。なお、２倍算を用いた高速演算手法として発表された文献として、非特許文献１９がある。

【特許文献１】E.Knudsen. COMPUTING METHOD FOR ELLIPTIC CURVE CRYPTOGRAPHY, WO 01/04742 A1, 18 Jan 2001

【特許文献２】R.Schroeppel. Elliptic curve point ambiguity resolution apparatus and method, WO 01/35573 A1, 17 May 2000

【非特許文献１】N.Koblitz. Hyperelliptic curve cryptosystems. J.Cryptology, vol. 1,No.3, pp.139-150, 1989.

【非特許文献２】D.G.Cantor. Computing in the Jacobian of hyperelliptic curve. Math. Comp., Vol.48, No. 177, pp.95-101, 1987
【非特許文献３】「D. Mumford, Tata lectures on theta II, Progress in Mathematics, no. 43, Birkhauser, 1984.」
【非特許文献４】K.Matsuo, J.Chao, and S.Tsujii. Fast Genus two hyperelliptic curve cryptosystems. Technical Report ISEC2001-31, IEICE Japan, 2001.
【非特許文献５】M.Takahashi. Improving Harley algorithms for Jacobians of genus 2 hyperelliptic curves. SCIS2002. (Japanese).
【非特許文献６】T.Lange. Inversion-free arithmetic on genus 2 hyperelliptic curves. Cryptology ePrint Archive, 2002/147, IACR, 2002.
【非特許文献７】T.Sugizaki, K.Matsuo, J.Chao, and S.Tsujii. An extension of Harley addition algtorithm for hyperelliptic curves over finite fields of characteristic two. ISEC2002-9, IEICE, 2001
【非特許文献８】T.Lange, Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae. Cryptology ePrint Archive, 2002/121, IACR, 2002.
【非特許文献９】J.Kuroki, M.Gonda, K.Masuo, J.Chao and S.Tsujii. Fast genus three hyperellipitc curve cryptosystems. SCIS2002
【非特許文献１０】J.Pelzl, T.Wollinger, J. Guajardo, and C. Paar. Hyperelliptic curve Cryptosystems: Closing the Performance Gap to Elliptic Curves. Cryptology ePrint Archive, 2003/026, IACR, 2003.
【非特許文献１１】Y.Miyamoto, H.Doi, K.Matsuo, J.Chao and S.Tsujii. A fast addition algorithm of genus two hyperelliptic curves. SCIS2002. (Japanese).
【非特許文献１２】N.Takahashi, H.Morimoto and A.Miyaji. Efficient exponentiation on genus two hyperelliptic curves (II). ISEC2002-145, IEICE, 2003. (Japanese)
【非特許文献１３】T.Lange. Weighed coordinate on genus 2 hyperellipitc curve. Cryptology ePrint Archive, 2002/153, IACR, 2002.
【非特許文献１４】N. Nagao. Improving group law algorithms for Jacobians of hyperelliptic curves. ANTS-IV, LNCS 1838, pp.439-448, Springer-Verlag, 2000.
【非特許文献１５】E.Knudsen. Elliptic Scalar Multiplication Using Point Halving. ASIACRYPTO '99, LNCS 1716, pp.135-149, Springer-Verlag, 1999.
【非特許文献１６】K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18.pdf
【非特許文献１７】S. Dquesne. Montgomery Scalar Multiplication for Genus 2 Curves. ANTS-VI, LNCS 3076, pp.153-168, 2004.
【非特許文献１８】T. Lange. Montgomery Addition for Genus Two Curves. ANTS-VI, LNCS 3076, pp.309-317, 2004.
【非特許文献１９】T. Lange. Efficient Doubling on Genus Two Curves over Binary Fields, SAC 2004, pre-proceedings, pp.189-202, 2004.

【発明の開示】
【発明が解決しようとする課題】
【００４８】
現在、実用フェーズに入りつつある楕円曲線暗号（ＥＣＣ）アルゴリズムに対して、その拡張概念である超楕円曲線暗号（ＨＥＣＣ）アルゴリズムは、現在、学会レベルで高速アルゴリズムや、その実装方法についての研究が進められている。超楕円曲線暗号（ＨＥＣＣ）のスカラー倍算の演算時間は、楕円曲線暗号（ＥＣＣ）に近づきつつある程度に過ぎず、さらなる高速化が望まれている。
【００４９】
本発明は、このような現状に鑑みてなされたものであり、超楕円曲線暗号（ＨＥＣＣ）

のスカラー倍算の演算時間を短縮し、高速処理可能な超楕円曲線暗号（ＨＥＣＣ）演算処理を実現する暗号処理演算方法、および暗号処理装置、並びにコンピュータ・プログラムを提供することを目的とする。

【００５０】

本発明は、楕円曲線暗号の１／２倍算を、超楕円曲線暗号に拡張し高速に計算できるアルゴリズム、曲線パラメータ、定義体を見つけ、超楕円曲線暗号において１／２倍算を適用した演算処理により、高速演算処理を実現する暗号処理演算方法、および暗号処理装置、並びにコンピュータ・プログラムを提供することを目的とする。

【課題を解決するための手段】

【００５１】

本発明の第１の側面は、

超楕円曲線暗号に基づく暗号処理演算を実行する暗号処理演算方法であり、

超楕円曲線の因子Ｄに対するスカラー倍算の演算において、演算処理として、１／２倍算を含む演算を実行する演算ステップ、

を有することを特徴とする暗号処理演算方法にある。

【００５２】

さらに、本発明の暗号処理演算方法の一実施態様において、前記演算ステップは、種数２、標数２のランダムパラメータを持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする。

【００５３】

さらに、本発明の暗号処理演算方法の一実施態様において、前記演算ステップは、種数２、標数２の$h(x)=x^2+x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする。

【００５４】

さらに、本発明の暗号処理演算方法の一実施態様において、前記演算ステップは、種数２、標数２の$h(x)=x^2+h_1x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする。

【００５５】

さらに、本発明の暗号処理演算方法の一実施態様において、前記演算ステップは、種数２、標数２の$h(x)=x$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行するステップであることを特徴とする。

【００５６】

さらに、本発明の暗号処理演算方法の一実施態様において、前記暗号処理演算方法は、予め固定された因子Ｄについての［$1/2^iD$］計算値に基づいて、k_1，k_1'，（k_0，k_0'）のいずれが正しいかを記録したテーブルを参照するテーブル参照ステップを有し、前記演算ステップは、前記テーブルの参照に基づく判定処理によって、１／２倍算の計算量を削減した演算処理として実行することを特徴とする。

【００５７】

さらに、本発明の暗号処理演算方法の一実施態様において、前記演算ステップは、

入力：$D_2=(U_2, V_2)$、

出力：$D_1=(U_1, V_1)=[1/2]D_2$、

ただし、$U_i(x)=x^2+u_{i1}x+u_{i0}$，$V_i(x)=v_{i1}x+v_{i0}$，$gcd(h, U_i)=1$，$i=1, 2$、

とした１／２倍算の演算アルゴリズムに基づいて導出される以下の関係式、

$1/k_1=h_2+k_1u_{21}$

を適用し、逆元演算を行なうことなく乗算および加算処理によって逆元$1/k_1$の値を算出するステップを含むことを特徴とする。

【００５８】

さらに、本発明の暗号処理演算方法の一実施態様において、前記暗号処理演算方法は、
入力：$D_2=(U_2, V_2)$、
出力：$D_1=(U_1, V_1)=[1/2]D_2$、
ただし、$U_i(x)=x^2+u_{i1}x+u_{i0}$，$V_i(x)=v_{i1}x+v_{i0}$，$gcd(h, U_i)=1$，$i=1, 2$、
とした１／２倍算演算アルゴリズムにおいて、$1/u_{21}$を入力値として適用しない設定を持つアルゴリズムに従った演算を実行することを特徴とする。

【００５９】

さらに、本発明の暗号処理演算方法の一実施態様において、前記暗号処理演算方法は、種数２、標数２の$h(x)=x^2+h_1x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算を実行する演算方法であり、前記演算ステップは、算出済みの値として$1/h_1^2$を入力値として設定し、逆元$1/h_1^2$を算出する処理を実行することなく、算出済みの入力値$1/h_1^2$を適用するステップを含むことを特徴とする。

【００６０】

さらに、本発明の第２の側面は、
超楕円曲線暗号に基づく暗号処理演算を実行する暗号処理装置であり、
超楕円曲線の因子Ｄに対するスカラー倍算の演算において、演算処理として、１／２倍算を含む演算を実行する演算実行部を有することを特徴とする暗号処理装置にある。

【００６１】

さらに、本発明の暗号処理装置の一実施態様において、前記演算実行部は、種数２、標数２のランダムパラメータを持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であることを特徴とする。

【００６２】

さらに、本発明の暗号処理装置の一実施態様において、前記演算実行部は、種数２、標数２の$h(x)=x^2+x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であることを特徴とする。

【００６３】

さらに、本発明の暗号処理装置の一実施態様において、前記演算実行部は、種数２、標数２の$h(x)=x^2+h_1x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であることを特徴とする。

【００６４】

さらに、本発明の暗号処理装置の一実施態様において、前記演算実行部は、種数２、標数２の$h(x)=x$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する構成であることを特徴とする。

【００６５】

さらに、本発明の暗号処理装置の一実施態様において、前記暗号処理装置は、予め固定された因子Ｄについての$[1/2^iD]$計算値に基づいて、k_1，k_1'，(k_0，k_0')のいずれが正しいかを記録したテーブルを格納した記憶部を有し、前記演算実行部は、前記テーブルの参照に基づく判定処理によって、１／２倍算の計算量を削減した演算処理を実行する構成であることを特徴とする。

【００６６】

さらに、本発明の暗号処理装置の一実施態様において、前記演算実行部は、
入力：$D_2=(U_2, V_2)$、
出力：$D_1=(U_1, V_1)=[1/2]D_2$、
ただし、$U_i(x)=x^2+u_{i1}x+u_{i0}$，$V_i(x)=v_{i1}x+v_{i0}$，$gcd(h, U_i)=1$，$i=1, 2$、
とした１／２倍算の演算アルゴリズムに基づいて導出される以下の関係式、

$1/k_1 = h_2 + k_1 u_{21}$

を適用し、逆元演算を行なうことなく乗算および加算処理によって逆元$1/k_1$の値を算出する構成であることを特徴とする。

【００６７】

さらに、本発明の暗号処理装置の一実施態様において、前記演算実行部は、

入力：$D_2 = (U_2, V_2)$、

出力：$D_1 = (U_1, V_1) = [1/2] D_2$、

ただし、$U_i(x) = x^2 + u_{i1}x + u_{i0}$，$V_i(x) = v_{i1}x + v_{i0}$，$gcd(h, U_i) = 1$，$i = 1, 2$、

とした１／２倍算演算アルゴリズムを実行する構成であり、$1/u_{21}$を入力値として適用しない設定を持つアルゴリズムに従った演算を実行する構成であることを特徴とする。

【００６８】

さらに、本発明の暗号処理装置の一実施態様において、前記演算実行部は、種数２、標数２の$h(x) = x^2 + h_1 x + h_0$，$f_4 = 0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算を実行する構成を有し、算出済みの値として$1/h_1^2$を入力値とし、逆元$1/h_1^2$を算出する処理を実行することなく、算出済みの入力値$1/h_1^2$を適用した演算を実行する構成であることを特徴とする。

【００６９】

さらに、本発明の第３の側面は、

超楕円曲線暗号に基づく暗号処理演算をコンピュータ上で実行させるコンピュータ・プログラムであり、

超楕円曲線の因子Ｄに対するスカラー倍算の演算において、演算処理として、１／２倍算を含む演算を実行する演算ステップ、

を有することを特徴とするコンピュータ・プログラムにある。

【００７０】

なお、本発明のコンピュータ・プログラムは、例えば、様々なプログラム・コードを実行可能なコンピュータ・システムに対して、コンピュータ可読な形式で提供する記憶媒体、通信媒体、例えば、ＣＤやＦＤ、ＭＯなどの記録媒体、あるいは、ネットワークなどの通信媒体によって提供可能なコンピュータ・プログラムである。このようなプログラムをコンピュータ可読な形式で提供することにより、コンピュータ・システム上でプログラムに応じた処理が実現される。

【００７１】

本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。なお、本明細書においてシステムとは、複数の装置の論理的集合構成であり、各構成の装置が同一筐体内にあるものには限らない。

【発明の効果】

【００７２】

本発明の構成によれば、楕円曲線暗号の１／２倍算を、超楕円曲線暗号に拡張し、高速演算が実現される。超楕円曲線上の因子の演算を使う暗号処理演算において、処理の重い演算は因子のスカラー倍算であり、本発明の処理によりスカラー倍算を高速化することで超楕円曲線暗号の処理を大幅に改善することができる。

【００７３】

本発明の構成によれば、超楕円曲線暗号の因子Ｄに対するスカラー倍算において、演算処理として、１／２倍算を含む演算を実行することでスカラー倍算を高速化することができる。例えば、種数２、標数２の$h(x) = x^2 + x + h_0$，$f_4 = 0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算、あるいは、種数２、標数２の$h(x) = x^2 + h_1 x + h_0$，$f_4 = 0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算、あるいは、種数２、標数２の$h(x) = x$をパラメータに持つ超楕円曲線の因子Ｄに

対するスカラー倍算において１／２倍算を含む演算を実行することで、高速演算が実現される。

【００７４】

さらに、本発明の構成によれば、予め固定された因子Ｄについての［１／２^{i}Ｄ］計算値に基づいて、k_1，k_1'，（k_0，k_0'）のいずれが正しいかを記録したテーブルを適用することで、因子のスカラー倍算における計算量をさらに削減することが可能となり、さらなる高速化が実現される。

【００７５】

さらに、本発明の構成によれば、超楕円曲線暗号の因子Ｄに対するスカラー倍算において、演算処理として、１／２倍算を含む演算を実行する構成とするとともに、１／２倍算の演算処理において逆元演算の実行回数を減少させるアルゴリズムを適用したので、因子のスカラー倍算における計算量をさらに削減することが可能となり、さらなる高速化が実現される。

【発明を実施するための最良の形態】

【００７６】

以下、本発明の暗号処理装置および暗号処理演算方法、並びにコンピュータ・プログラムの詳細について説明する。

【００７７】

本発明は、楕円曲線暗号を一般化した超楕円曲線暗号（ＨＥＣＣ：Hyper Elliptic Curve Cryptography）についての高速化演算手法である。前述したように、超楕円曲線では曲線を特徴づける値は種数（ｇｅｎｕｓ）ｇである。ｐを素数、ｎを正の整数、$q=p^n$とする。このとき有限体Ｆｑ上で定義される種数ｇの超楕円曲線Ｃは以下の方程式、

$y^2+h(x)y=f(x)$

で定義される。ここで、ｈ（ｘ），ｆ（ｘ）∈Ｆｑ［ｘ］，ｆ（ｘ）は、次数２ｇ＋１のｍｏｎｉｃ多項式である。

【００７８】

超楕円曲線Ｃ上の点Ｐ＝（ｘ、ｙ）に対する共役点（ｏｐｐｏｓｉｔｅ　ｐｏｉｎｔ）は、－Ｐ＝（ｘ、ｙ＋ｈ（ｘ））として定義される。また、Ｐ＝－Ｐである点を分岐点（ｒａｍｉｆｉｃａｔｉｏｎ　ｐｏｉｎｔ）と呼ぶ。

【００７９】

超楕円曲線暗号の定義体の演算サイズ（ビット長）は楕円曲線暗号と同等の安全性を仮定した場合、楕円曲線の定義体の演算サイズに比べて１／ｇに小さくなることが知られている。演算サイズが小さいことは実装上メリットがあり、超楕円曲線暗号の利点の一つして挙げられる。

【００８０】

前述したように、超楕円曲線暗号では点の形式的和である因子（ｄｉｖｉｓｏｒ）をD_1とし、スカラー倍算kD_1で定義される因子をD_2とすると、D_2からｋを求める問題は超楕円曲線におけるヤコビ多様体上の離散対数問題として公開鍵暗号として用いることができる。

【００８１】

ここで、因子（ｄｉｖｉｓｏｒ）は有理点の形式和であり、以下の形式で表現することができる。

【数７】

$$D=\sum_i m_i P_i$$

【００８２】
また、半被約因子（ｓｅｍｉ　ｒｅｄｕｃｅｄ　ｄｉｖｉｓｏｒ）は、以下の形式で表現することができる。
【数８】

$$D=\sum_i m_i P_i-(\sum_i m_i)P_\infty,\quad m_i\geq 0$$

ただし、P_i＝（x_i、y_i）かつｉ≠ｊに対して$P_i \neq P_j$である。
【００８３】
また、Σm_iをＤのウェイト（ｗｅｉｇｈｔ）と呼ぶ。さらにウェイト（ｗｅｉｇｈｔ）が種数ｇ以下である半被約因子を被約因子（ｒｅｄｕｃｅｄ　ｄｉｖｉｓｏｒ）と呼ぶ。
【００８４】
超楕円曲線のヤコビ（Ｊａｃｏｂｉ）多様体上の任意の半被約因子Ｄは下記の多項式Ｕ、Ｖ∈Ｆｑ［ｘ］を用いてＤ＝（Ｕ、Ｖ）として表現できる。これをマンホード（Ｍｕｍｆｏｒｄ）表現と呼ぶ。
【数９】

$$U=\prod(x-x_i)^{m_i}$$

$$V(x_i)=y_i$$

$$V(x)^2+V(x)h(x)-f(x)\equiv 0 \mod U(x),\quad \deg V<\deg U$$

【００８５】
種数（ｇｅｎｕｓ）２の任意の被約因子Ｄはマンホード（Ｍｕｍｆｏｒｄ）表現を用いると、有限体Ｆｑ上の元を係数に持つ２次以下の多項式の組、すなわち、
（Ｕ、Ｖ）＝（$x^2+u_1x+u_0$、v_1x+v_0）、または、

（Ｕ、Ｖ）＝（ｘ＋ｘ$_0$、ｙ$_0$）
として表現することができる。
また、零元は、
（Ｕ，Ｖ）＝（１，０）＝Ｏ
として表現される。

【００８６】

本発明は、楕円曲線暗号の１／２倍算を、超楕円曲線暗号に拡張し、２倍算よりも高速に計算できるアルゴリズム、曲線パラメータ、定義体を見つけ、超楕円曲線暗号において、２倍算よりも高速な１／２倍算を適用した演算処理を実現するものである。以下、本発明の実施例として、まず、前半において、以下に示す処理例１～６の手法について説明する。

【００８７】

（処理例１：提案法Ａ１）種数２、標数２のランダムパラメータを持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法。

（処理例２：提案法Ｆ１）種数２，標数２のｈ（ｘ）＝ｘ^2＋ｈ$_1$ｘ＋ｈ$_0$、ｆ$_4$＝０をパラメータに持つような超楕円曲線において因子Ｄの１／２倍算を計算する方法。

（処理例３：提案法Ｂ１）種数２、標数２のｈ（ｘ）＝ｘ^2＋ｘ＋ｈ$_0$、ｆ$_4$＝０をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法。

（処理例４：提案法Ｅ１）種数２、標数２のｈ（ｘ）＝ｘをパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法。

（処理例５：提案法Ｃ１）種数２、標数２のランダムパラメータを持つような超楕円曲線、ｈ（ｘ）＝ｘ^2＋ｈ$_1$ｘ＋ｈ$_0$，ｆ$_4$＝０をパラメータに持つ超楕円曲線、およびｈ（ｘ）＝ｘ^2＋ｘ＋ｈ$_0$、ｆ$_4$＝０、またはｈ（ｘ）＝ｘをパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する際に、１／２倍したものの候補が２つ出てくる。２つの候補のうち、正しい値を選ぶ必要があるが、選ぶ際に、有限体のトレース、乗算、平方根を計算する必要がある。どちらが正しいかは、因子Ｄに依存する。そのため、因子Ｄが固定されている場合、事前に２つの候補のうち、どちらが正しいかをテーブルに保持し、正しい値を選ぶ際に、このテーブルを参照することにより、上記の余計な計算を省略する方法。

（処理例６：提案法Ｄ１）上記処理例１～５に示す因子の１／２倍算の計算方法を用いて、因子のスカラー倍算を計算する方法。

【００８８】

さらに、後半において、上記処理例１～３処理例５～６をさらに改善した手法として、

（処理例７：提案法Ａ２）種数２、標数２のランダムパラメータを持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法。

（処理例８：提案法Ｆ２）種数２，標数２のｈ（ｘ）＝ｘ^2＋ｈ$_1$ｘ＋ｈ$_0$、ｆ$_4$＝０をパラメータに持つような超楕円曲線において因子Ｄの１／２倍算を計算する方法。

（処理例９：提案法Ｂ２）種数２、標数２のｈ（ｘ）＝ｘ^2＋ｘ＋ｈ$_0$、ｆ$_4$＝０をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法。

（処理例１０：提案法Ｃ２）種数２、標数２のランダムパラメータを持つような超楕円曲線、ｈ（ｘ）＝ｘ^2＋ｈ$_1$ｘ＋ｈ$_0$，ｆ$_4$＝０をパラメータに持つ超楕円曲線、およびｈ（ｘ）＝ｘ^2＋ｘ＋ｈ$_0$、ｆ$_4$＝０をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する際に、１／２倍したものの候補が２つ出てくる。２つの候補のうち、正しい値を選ぶ必要があるが、選ぶ際に、有限体のトレース、乗算、平方根を計算する必要がある。どちらが正しいかは、因子Ｄに依存する。そのため、因子Ｄが固定されている場合、事前に２つの候補のうち、どちらが正しいかをテーブルに保持し、正しい値を選ぶ際に、このテーブルを参照することにより、上記の余計な計算を省略する方法。

（処理例１１：提案法Ｄ２）上記処理例７～１０に示す因子の１／２倍算の計算方法を用いて、因子のスカラー倍算を計算する方法。

【００８９】

以下、各処理例について、順次、その詳細を説明する。

【００９０】

［処理例１：提案法Ａ１］

処理例１（提案法Ａ１）は、種数２、標数２のランダムパラメータを持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法である。

【００９１】

また、これから扱う因子は位数がｒのものとする。つまり、扱う因子は分岐点を持たない。入力因子を、

$D_2=(U_2, V_2)$、

$U_2=u_{22}x^2+u_{21}x+u_{20}$、

$V_2=v_{21}x+v_{20}$、

ここで、D_2のｗｅｉｇｈｔが２の場合は$u_{22}=1$、

D_2のｗｅｉｇｈｔが１の場合は$u_{22}=0$、$u_{21}=1$、$v_{21}=0$とする。

分岐点を持たないため、１／２倍算として、

$ExHarDBL^{1+1\to 2}$、$ExHarDBL^{2+2\to 1}$、$ExHarDBL^{2+2\to 2}$、および、ＨａｒｌｅｙＤＢＬの４つの逆演算を考えればよい。ＨａｒｌｅｙＤＢＬ以外は例外ケースである。

【００９２】

ここで$ExHarDBL^{2+2\to 1}$は入力因子のｗｅｉｇｈｔが２で、出力因子のｗｅｉｇｈｔが１の場合を計算する方法である。また、$ExHarDBL^{2+2\to 2}$は入力因子のｗｅｉｇｈｔが２で、かつU_2の１次の項の係数が$u_{21}=0$を満たし、出力因子のｗｅｉｇｈｔが２の場合を計算する方法である。ただし、$ExHarDBL^{2+2\to 2}$はＨａｒｌｅｙＤＢＬで計算できるが、これの逆演算である１／２倍算は、例外ケースとなるため、ここでは、$ExHarDBL^{2+2\to 2}$を例外ケースとして扱う。

【００９３】

これら、$ExHarDBL^{1+1\to 2}$、$ExHarDBL^{2+2\to 1}$、$ExHarDBL^{2+2\to 2}$、および、ＨａｒｌｅｙＤＢＬに対応する１／２倍算をそれぞれ、$ExHEC\_HLV^{2\to 1+1}$、$ExHEC\_HLV^{1\to 2+2}$、$ExHEC\_HLV^{2\to 2+2}$、およびＨＥＣ＿ＨＬＶとする。

【００９４】

因子の１／２倍算を行なう場合、まず、入力因子により、図２に示す通りに場合分けを行なう。入力因子のｗｅｉｇｈｔが２で、かつU_2の１次の項の係数が$u_{21}\neq 0$を満たす場合は、ＨＥＣ＿ＨＬＶで計算を行なう。また、入力因子のｗｅｉｇｈｔが２で、かつU_2の項の係数が$u_{21}=0$を満たす場合は、$ExHEC\_HLV^{2\to 2+2}$、もしくは、$ExHEC_HLV^{2\to 1+1}$で計算を行なう。また、入力因子のｗｅｉｇｈｔが１の場合は$ExHEC_HLV^{1\to 2+2}$で計算を行う。ＨＥＣ＿ＨＬＶのアルゴリズムについて、図３を参照して説明する。

【００９５】

図３は、ＨＥＣ＿ＨＬＶのアルゴリズムを示すフローチャートである。

ステップＳ１０１において、入力を、

$D_2=(U_2, V_2)$、

$U_2=x^2+u_{21}x+u_{20}$、

$V_2=v_{21}x+v_{20}$、

とする。

【００９６】

ステップＳ１０２において、$k_1h_2+k_1^2u_{21}+1=0$の根、k_1，k_1'を求め、ステップＳ１０３において、$c_1\leftarrow f_3+h_2v_{21}+u_{20}+(f_4+u_{21})u_{21}$とおき、ステップＳ１０４において、$k_1h_0+k_0h_1+k_0^2u_{21}+c_1=0$が根を持つか否かを判定し、根を持たない場合は、ステップＳ１０５において、$k_1\leftarrow k_1'$とし、根を持つ場合は、ステップＳ１０６に進み、$k_1h_0+k_0h_1+k_0^2u_2$

$_1$＋ｃ$_1$＝０の根、ｋ$_0$，ｋ$_0$'を求める。

【００９７】

次にステップＳ１０７に進み、ｕ$_{11}$を計算し、ステップＳ１０８において、ｘｈ$_2$＋ｘ^{2}ｕ$_{11}$＋１＝０が根を持つか否かを判定し、根を持たない場合は、ステップＳ１０９において、ｋ$_0$←ｋ$_0$'とし、根を持つ場合は、ステップＳ１１０に進み、ｕ$_{10}$を計算する。さらに、ステップＳ１１１において、ｖ$_{11}$，ｖ$_{10}$を計算し、ステップＳ１１２において、

Ｕ$_1$←ｘ^2＋ｕ$_{11}$ｘ＋ｕ$_{10}$

Ｖ$_1$←ｖ$_{11}$ｘ＋ｖ$_{10}$

として、

ステップＳ１１３において、出力、

Ｄ$_1$←（Ｕ$_1$，Ｖ$_1$）

を得る。

【００９８】

因子の１／２倍算は、因子の２倍算を行うアルゴリズム、すなわち、以下に示す［Ａｌｇｏｒｉｔｈｍ　１　ＨａｒｌｅｙＤＢＬ］

【数１０】

Algorithm 1 *HarleyDBL*

Input : $D_1 = (U_1, V_1)$

Output : $D_2 = (U_2, V_2)$

$U_i(x) = x^2 + u_{i1}x + u_{i0},\ V_i(x) = v_{i1}x + v_{i0},\ \gcd(h, U_1) = 1$

1. $U_1' \leftarrow U_1^2$
2. $S \leftarrow h^{-1}(f + hV_1 + V_1^2)/U_1 \bmod U_1$
3. $V_1' \leftarrow SU_1 + V_1$
4. $U_2' \leftarrow (f + hV_1' + V_1'^2)/U'$
5. $U_2 \leftarrow \mathrm{MakeMonic}(U_2')$
6. $V_2 \leftarrow V_1' + h \bmod U_2$
7. return $D_2 = (U_2, V_2)$

【００９９】

上記アルゴリズム１［Ａｌｇｏｒｉｔｈｍ　１　ＨａｒｌｅｙＤＢＬ］を逆から計算することにより実現する。Ａｌｇｏｒｉｔｈｍ　１のステップ６では、

Ｖ$_1$'＋ｈ＝（ｋ$_1$ｘ＋ｋ$_0$）Ｕ$_2$＋Ｖ$_2$

を満たすような多項式、

ｋ（ｘ）＝ｋ$_1$ｘ＋ｋ$_0$

が唯一存在する。

【０１００】

これを、

Ｖ$_1$'＝ｈ＋（ｋ$_1$ｘ＋ｋ$_0$）Ｕ$_2$＋Ｖ$_2$

と式変形し、ステップ４に現れる式、

（ｆ＋ｈＶ$_{1}$'＋Ｖ$_{1}$'^{2}）
に代入すると、ステップ４は、次のようになる。
Ｕ$_{2}$'Ｕ$_{1}$'＝ｆ＋ｈ（ｋＵ$_{2}$＋Ｖ$_{2}$）＋ｋ^{2}Ｕ$_{2}$^{2}＋Ｖ$_{2}$^{2}・・・式（１）
上記式において、
（Ｕ$_{2}$，Ｖ$_{2}$）は分かっているので、式（１）から、ｋとＵ$_{1}$'との関係式が得られる。
ここで、
Ｕ$_{2}$'＝ｋ$_{1}$^{2}Ｕ$_{2}$
であることに注意する。

【０１０１】

上記式（１）を展開して整理すると、次のようになる。
Ｕ$_{1}$'＝ｘ^{4}＋（（ｋ$_{1}$ｈ$_{2}$＋ｋ$_{1}$^{2}ｕ$_{2\,1}$＋１）／ｋ$_{1}$^{2}）ｘ^{3}＋（（ｋ$_{1}$ｈ$_{1}$＋ｋ$_{0}$ｈ$_{2}$＋ｋ$_{1}$^{2}ｕ$_{2\,0}$＋ｋ$_{0}$^{2}＋ｃ$_{2}$）／ｋ$_{1}$^{2}）ｘ^{2}＋（（ｋ$_{1}$ｈ$_{0}$＋ｋ$_{0}$ｈ$_{1}$＋ｋ$_{0}$^{2}ｕ$_{2\,1}$＋ｃ$_{1}$）／ｋ$_{1}$^{2}）ｘ＋（ｋ$_{0}$ｈ$_{0}$＋ｋ$_{0}$^{2}ｕ$_{2\,0}$＋ｃ$_{0}$）／ｋ$_{1}$^{2}
・・・式（２）
ここで，
ｃ$_{2}$＝ｆ$_{4}$＋ｕ$_{2\,1}$
ｃ$_{1}$＝ｆ$_{3}$＋ｈ$_{2}$ｖ$_{2\,1}$＋ｕ$_{2\,0}$＋ｃ$_{2}$ｕ$_{2\,1}$
ｃ$_{0}$＝ｆ$_{2}$＋ｈ$_{2}$ｖ$_{2\,0}$＋ｈ$_{1}$ｖ$_{2\,1}$＋ｖ$_{2\,1}$^{2}＋ｃ$_{2}$ｕ$_{2\,0}$＋ｃ$_{1}$ｕ$_{2\,1}$
を満たす。

【０１０２】

また、ステップ１より、
Ｕ$_{1}$'＝Ｕ$_{1}$^{2}、
つまり、
Ｕ$_{1}$'＝ｘ^{4}＋ｕ$_{1\,1}$^{2}ｘ^{2}＋ｕ$_{1\,0}$^{2}
・・・式（３）
が成り立つ。

【０１０３】

上記式（２）、（３）の各係数を比較することにより、関係式を導き、この関係式を解くことにより、１／２倍算が計算できる。上記の処理手順を示すアルゴリズムを以下に、アルゴリズム４［Ａｌｇｏｒｉｔｈｍ　４　Ｓｋｅｔｃｈ　ＨＥＣ＿ＨＬＶ］として示す。

【数１１】

Algorithm 4 *Sketch HEC _ HLV*

Input : $D_2 = (U_2, V_2)$

Output : $D_1 = (U_1, V_1) = [1/2]D_2$

$U_i(x) = x^2 + u_{i1}x + u_{i0}$, $V_i(x) = v_{i1}x + v_{i0}$, $\gcd(h, U_i) = 1$, $i = 1, 2$

1. reconstruct k_0, k_1
   1.1 $V_1' \leftarrow V_2 + h + kU_2$, $k(x) = k_1 x + k_0$
   1.2 $U_1' \leftarrow (f + hV_1' + V_1'^2)/(k_1^2 U_2)$
   1.3 derive k_0, k_1 from $\mathrm{coeff}(U_1', 3) = 0, \mathrm{coeff}(U_1', 1) = 0$
2. compute u_{11} by substituting k_0, k_1 in $\mathrm{coeff}(U_1', 2)$
3. compute u_{10} by substituting k_0, k_1 in $\mathrm{coeff}(U_1', 0)$
4. $U_1 \leftarrow x^2 + u_{11}x + u_{10}$
5. $V_1 \leftarrow V_2 + h + kU_2 \bmod U_1$
6. return $D_1 = (U_1, V_1)$

【０１０４】

具体的には，次の関係式が得られる。

【数１２】

$$k_1 h_2 + k_1^2 u_{21} + 1 = 0 \qquad \cdots \text{式（４）}$$

$$k_1 h_0 + k_0 h_1 + k_0^2 u_{21} + c_1 = 0 \qquad \cdots \text{式（５）}$$

$$u_{11} = \sqrt{k_1 h_1 + k_0 h_2 + k_1^2 u_{20} + k_0^2 + c_2}\,/\,k_1 \qquad \cdots \text{式（６）}$$

$$u_{10} = \sqrt{k_0 h_0 + k_0^2 u_{20} + c_0}\,/\,k_1 \qquad \cdots \text{式（７）}$$

【０１０５】

これらの関係式から，正しいｋ$_0$，ｋ$_1$を計算する必要があるが、これは、次の補題によって計算できる。

［補題１］

ｈ（ｘ）を既約多項式であると仮定する。このとき、式（４），（５）を満たすｋ$_1$は唯一である。また、式（５）は、正しいｋ$_1$に対してのみ、根を持つ。また、Ａｌｇｏｒｉｔｈｍ４でのｌ／２倍された因子Ｄ$_1$を計算できるｋ$_0$は唯一存在する。また、以下に示す式、

ｘｈ$_2$＋ｘ^{2}ｕ$_{1\ 1}$＋１＝０

は、正しいｋ$_0$に対してのみ根を持つ。

【０１０６】

Ａｌｇｏｒｉｔｈｍ４に対して上記補題１を適用した。詳細な１／２倍算の計算方法を以下のアルゴリズム５［Ａｌｇｏｒｉｔｈｍ　５　ＨＥＣ＿ＨＬＶ］として示す。

【数１３】

Algorithm 5 *HEC_HLV*

Input : $D_2 = (U_2, V_2)$

Output : $D_1 = (U_1, V_1) = [1/2]D_2$

$U_i(x) = x^2 + u_{i1}x + u_{i0},\ V_i(x) = v_{i1}x + v_{i0},\ \gcd(h, U_i) = 1, i = 1, 2$

1. Solve $k_1h_2 + k_1^2u_{21} + 1 = 0$

   $\alpha \leftarrow h_2 / u_{21},\ \gamma \leftarrow 1/(u_{21}\alpha^2)$　　/*$1/(u_{21}\alpha^2) = u_{21}/h_2^2$*/

   $k_1 \leftarrow \mathrm{H}(\gamma)\alpha,\ k_1' \leftarrow k_1 + \alpha$

2. Select correct k_1 by solving $k_1h_0 + k_0h_1 + k_0^2u_{21} + c_1 = 0$

   $c_2 \leftarrow f_4 + u_{21},\ c_1 \leftarrow f_3 + h_2v_{21} + u_{20} + c_2u_{21}$

   $c_0 \leftarrow f_2 + h_2v_{20} + h_1v_{21} + v_{21}^2 + c_2u_{20} + c_1u_{21}$

   $\alpha \leftarrow h_1 / u_{21},\ \gamma \leftarrow (c_1 + k_1h_0)/(u_{21}\alpha^2)$

   if $\mathrm{Tr}(\gamma) = 1$ then $k_1 \leftarrow k_1',\ \gamma \leftarrow (c_1 + k_1h_0)/(u_{21}\alpha^2)$

   $k_0 \leftarrow \mathrm{H}(\gamma)\alpha,\ k_1' \leftarrow k_1 + \alpha$

3. Select correct k_0 by checking trace of $xh_2 + x^2u_{11} + 1 = 0$

   $u_{11} \leftarrow \sqrt{k_1h_1 + k_0h_2 + k_1^2u_{20} + k_0^2 + c_2}\ / k_1,\ \gamma \leftarrow u_{11}/h_2^2$

   if $\mathrm{Tr}(\gamma) = 1$ then

   $k_0 \leftarrow k_0',\ u_{11} \leftarrow \sqrt{k_1h_1 + k_0h_2 + k_1^2u_{20} + k_0^2 + c_2}\ / k_1$

4. *Compute* U_1

   $u_{10} \leftarrow \sqrt{k_0h_0 + k_0^2u_{20} + c_0}\ / k_1$

5. *Compute* $V_1 = V_2 + h + kU_2 \bmod U_1$

   $w \leftarrow h_2 + k_1u_{21} + k_0 + k_1u_{11}$

   $v_{11} \leftarrow v_{21} + h_1 + k_1u_{20} + k_0u_{21} + u_{10}k_1 + u_{11}w$

   $v_{10} \leftarrow v_{20} + h_0 + k_0u_{20} + w$

6. $U_1(x) \leftarrow x^2 + u_{11}x + u_{10},\ V_1(x) \leftarrow v_{11}x + v_{10}$

7. *return* $D_1 = (U_1, V_1)$

【０１０７】

上記アルゴリズム５［Ａｌｇｏｒｉｔｈｍ５］では、k_1'，k_0'が正しい値（言い換えるとk_1，k_0が正しくない値）である場合、計算量は、

３２Ｍ＋５Ｓ＋６Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

となる。ここでＭ，Ｓ，Ｉ，ＳＲ，Ｈ，Ｔはそれぞれ有限体上の乗算、２乗算、逆元演算、平方根演算、ハーフトレース（二次方程式の根を求める演算）、トレース（二次方程式の根があるか否かの判定）を意味する。このk_1'，k_0'が正しい値である場合、計算量は最も多くなる。

【０１０８】

次にk_1，k_0が正しい値（言い換えるとk_1'，k_0'が正しくない値）である場合、計算量は最も少なくなり、ステップ２では２Ｍの計算量が削減でき、ステップ３では、２Ｍ＋１ＳＲの計算量が削減できる。つまり、この場合の計算量は、

２８Ｍ＋５Ｓ＋６Ｉ＋２ＳＲ＋２Ｈ＋２Ｔとなり、計算量は最も少なくなる。

【０１０９】

次にk_1，k_0'が正しい値（言い換えるとk_1'，k_0が正しくない値）である場合，ステップ３では、２Ｍ＋１ＳＲの計算量が削減できる。つまり、この場合の計算量は、

３０Ｍ＋５Ｓ＋６Ｉ＋２ＳＲ＋２Ｈ＋２Ｔとなる。

【０１１０】

最後にk_1'，k_0が正しい値（言い換えるとk_1，k_0'が正しくない値）である場合、ステップ２では，２Ｍの計算量が削減できる。つまり、この場合の計算量は、

３０Ｍ＋５Ｓ＋６Ｉ＋３ＳＲ＋２Ｈ＋２Ｔとなる。

【０１１１】

上記の４つの場合が発生する確率を、計算機実験によって確認した結果、ほぼ同じ割合で発生していることが確認された。これ以降は、上記４つの場合が生じる確率は等しいとする。上記４つの場合の計算量の平均をとると、

３０Ｍ＋５Ｓ＋６Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ

となる。

【０１１２】

次に、例外ケースの、

$ExHEC\_HLV^{2\to 1+1}$、

$ExHEC\_HLV^{1\to 2+2}$、

$ExHEC\_HLV^{2\to 2+2}$、

について考える。これらが発生する確率は無視できる程度であるので、計算量の評価は行なわない。

【０１１３】

まず、$ExHEC\_HLV^{2\to 1+1}$、のアルゴリズムを、図４のフローを参照して説明する。

$ExHEC_HLV^{2\to 1+1}$は、$ExHarDBL^{1+1\to 2}$を逆から計算することにより実現する。$ExHarDBL^{1+1\to 2}$の入力因子を、

$D_1=(U_1、V_1)$、$U_1=x+u_{10}$、$V_1=v_{10}$、

とすると、出力因子、

$D_2=(U_2、V_2)=2D_1$、$U_2=x^2+u_{20}x$、$V_2=v_{21}x+v_{20}$、

は、次のように計算できる。

$U_2=x^2+u_{20}=(x+u_{10})^2$、

$v_{12}=(u_{10}^4+f_3u_{10}^2+f_1+h_1v_{10})/h(u_{10})$，

$v_{20}=v_{10}+v_{21}u_{10}$

【０１１４】

この関係式を用いて、$ExHEC_HLV^{2\to 1+1}$を計算する。

入力因子を、

$D_2=(U_2、V_2)$、$U_2=x^2+u_{20}x$、$V_2=v_{21}x+v_{20}$、

とする（図４のフロー：ステップＳ２０１）と、

出力因子、
$D_1=(U_1、V_1)=[1/2]D_2$、$U_1=x+u_{10}$、$V_1=v_{10}$、
を求めるため、
ステップＳ２０２において、$u_{10}=\sqrt{u_{20}}$、
ステップＳ２０３において、$v_{10}=(v_{21}h(u_{10})+u_{10}^4+f_3u_{10}^2+f_1)/h_1$、
とし、ステップＳ２０４において、
$U_1=x+u_{10}$、$V_1=v_{10}$、
として、ステップＳ２０５において、
出力因子、
$D_1=(U_1、V_1)$
を得る。

【０１１５】

次に、$ExHEC\_HLV^{2\to 2+2}$、の処理手順について、図５のフローを参照して説明する。ステップＳ３０１において、入力因子を、
$D_2=(U_2、V_2)$、$U_2=x^2+u_{20}$、$V_2=v_{21}x+v_{20}$、
とする。

【０１１６】

ステップＳ３０２において、$k_1h_2+1=0$をk_1に関して解き、$k_1\leftarrow 1/h_2$とする。
ステップＳ３０３において、
$c_2\leftarrow f_4$
$c_1\leftarrow f_3+h_2v_{21}+u_{20}+u_{21}c_2$
として、ステップＳ３０４において、
$k_1h_0+k_0h_1+c_1=0$をk_0に関して解き、$k_0\leftarrow(k_1h_0+c_1)/h_1$とする。

【０１１７】

次に、ステップＳ３０５において、u_{11}を計算し、ステップＳ３０６において、
$xh_2+x^2u_{11}+1=0$
が根を持つか否かを判定し、根を持たない場合は、ステップＳ３０７において、
$D_1\leftarrow HEC\_HLV^{2\to 1}(D_2)$
によって出力D_1を決定（ステップＳ３０８）する。

【０１１８】

一方、
$xh_2+x^2u_{11}+1=0$
が根を持つ場合は、ステップＳ３０９に進み、u_{10}を計算し、さらに、ステップＳ３１０においてv_{11}，v_{10}を計算し、ステップＳ３１１において、
$U_1\leftarrow x^2+u_{11}x+u_{10}$
$V_1\leftarrow v_{11}x+v_{10}$
として、ステップＳ３１２において出力、
$D_1\leftarrow(U_1,V_1)$
を得る。

【０１１９】

この、$ExHEC\_HLV^{2\to 2+2}$、の処理は、具体的には、以下の手順で行なわれる。
入力因子を、
$D_2=(U_2、V_2)$、$U_2=x^2+u_{20}$、$V_2=v_{21}x+v_{20}$、
として、
$U_2=x^2+u_{20}$、
の場合、つまりU_2の１次の項が０の場合、出力因子の候補は２つあり、それらを、

$D_1=(x+\sqrt{u_{20}}, V_2(\sqrt{u_{20}}))$
$D_1'=(x^2+u_{11}x+u_{10}, v_{11}x+v_{10})$
と表す。

【０１２０】

D_1が正しければ、ＥｘＨＥＣ＿ＨＬＶ$^{2\to 1+1}$で計算する。
D_1'が正しければ、ＥｘＨＥＣ＿ＨＬＶ$^{2\to 2+2}$を用いて計算する。
どちらのアルゴリズムを用いるかは、次の手順で判定する。
１．D_1'が正しいと仮定する。
２．u_{11}を計算する。
３．$xh_2+x^2u_{11}+1=0$のトレースＴｒ（h_2/u_{11}^2）を計算する。もしＴｒ（h_2/u_{11}^2）＝０ならばD_1'が正しいので、ＥｘＨＥＣ＿ＨＬＶ$^{2\to 2+2}$を用いて計算する。そうではなくてＴｒ（h_2/u_{11}^2）＝１ならば、D_1が正しく、ＥｘＨＥＣ＿ＨＬＶ$^{2\to 1+1}$を用いて計算する。

【０１２１】

ＥｘＨＥＣ＿ＨＬＶ$^{2\to 2+2}$の計算アルゴリズムを、アルゴリズム６［Ａｌｇｏｒｉｔｈｍ６］として以下に示す。

【数１４】

Algorithm 6 $ExHEC\_HLV^{2\to 2+2}$

$Input: D_2 = (U_2, V_2) = (x^2 + u_{20}, v_{21}x + v_{20})$

$Output: D_1 = (U_1, V_1) = (x^2 + u_{11}x + u_{10}, v_{11}x + v_{10}) = [1/2]D_2$

$\gcd(h, U_i) = 1,\ i = 1, 2$

---

1. Solve $k_1 h_2 + 1 = 0$

   $k_1 \leftarrow 1/h_2$

2. Select correct k_1 by solving $k_1 h_0 + k_0 h_1 + k_0^2 + c_1 = 0$

   $c_2 \leftarrow f_4,\ c_1 \leftarrow f_3 + h_2 v_{21} + u_{20} + c_2 u_{21}$

   $c_0 \leftarrow f_2 + h_2 v_{20} + (h_1 + v_{21}) v_{21} + c_2 u_{20} + c_1 u_{21}$

   $k_0 \leftarrow (k_1 h_0 + c_1)/h_1$

3. Select correct algorithm by checking trace of $xh_2 + x^2 u_{11} + 1 = 0$

   $u_{11} \leftarrow \sqrt{k_1 h_1 + k_0 h_2 + k_1^2 u_{20} + k_0^2 + c_2}\ / k_1,\ \gamma \leftarrow u_{11}/h_2^2$

   if $\mathrm{Tr}(\gamma) = 1$ then

   $D_1 \leftarrow ExHEC\_HLV^{2\to 1+1}(D_2)$, return D_1

4. Compute U_1

   $u_{10} \leftarrow \sqrt{k_0 h_0 + k_0^2 u_{20} + c_0}\ / k_1$

5. Compute $V_1 = V_2 + h + kU_2 \bmod U_1$

   $w \leftarrow h_2 + k_1$

   $v_{11} \leftarrow h_1 + k_1 u_{20} + k_0 + u_{11} w$

   $v_{10} \leftarrow v_{20} + h_0 + k_0 u_{20} + u_{10} w$

6. $U_1(x) \leftarrow x^2 + u_{11}x + u_{10},\ V_1(x) \leftarrow v_{11}x + v_{10}$

7. return $D_1 = (U_1, V_1)$

次に、ＥｘＨＥＣ＿ＨＬＶ$^{1\to 2+2}$、の処理手順について、図６のフローを参照して説明する。ステップＳ４０１において、入力を、

$D_2 = (U_2, V_2)$、

$U_2 = x + u_{20}$、

$V_2 = v_{20}$、

とする。

【０１２２】

ステップＳ４０２において、$c_3 \leftarrow f_4 + u_{20}$として、ステップＳ４０３において、

$k_1 h_2 + k_1^2 u_{21} + c_3 = 0$の根、$k_1$，$k_1'$を求め、ステップＳ４０４において、

$c_2 \leftarrow f_3 + c_3 u_{20}$、

$c_1 \leftarrow f_2 + h_2 v_{20} + c_2 u_{20}$、

とおき、ステップＳ４０５において、$k_1 h_0 + k_0 h_1 + k_0^2 + c_1 = 0$が根を持つか否かを判定し、根を持たない場合は、ステップＳ４０６において、$k_1 \leftarrow k_1'$とした後、ステップＳ４０７に進み、根を持つ場合はそのままステップＳ４０７に進む。

【０１２３】

ステップＳ４０７では、$k_1 h_0 + k_0 h_1 + k_0^2 + c_1 = 0$の根、$k_0$，$k_0'$を求め、次にステップＳ４０８に進み、$u_{11}$を計算し、ステップＳ４０９において、$xh_2 + x^2 u_{11} + 1 = 0$が根を持つか否かを判定し、根を持たない場合は、ステップＳ４１０において$k_0 \leftarrow k_0'$とした後ステップＳ４１１に進む。根を持つ場合は、そのままステップＳ４１１に進み、u_{10}を計算する。さらに、ステップＳ４１２において、v_{11}，v_{10}を計算し、ステップＳ４１３において、

$U_1 \leftarrow x^2 + u_{11} x + u_{10}$

$V_1 \leftarrow v_{11} x + v_{10}$

として、

ステップＳ４１４において、出力、

$D_1 \leftarrow (U_1, V_1)$

を得る。

【０１２４】

$ExHEC_HLV^{1\to 2+2}$の計算手順は、ＨＥＣ＿ＨＬＶと似ているが、大きな違いは入力因子のｗｅｉｇｈｔである。そのため、$ExHEC_HLV^{1\to 2+2}$の$f + hV_1' + V_1'^2$は、５次のモニック多項式となり、

$U_1' \leftarrow (f + hV_1' + V_1'^2) / U_2$、

はＨＥＣ＿ＨＬＶのようにk_1^2で割ることはない。$ExHEC_HLV^{1\to 2+2}$の計算アルゴリズムを、アルゴリズム７［Ａｌｇｏｒｉｔｈｍ７］として以下に示す。

【数１５】

Algorithm 7 $ExHEC\_HLV^{1\to 2+2}$

Input : $D_2=(U_2,V_2)=(x+u_{20},v_{20})$

Output : $D_1=(U_1,V_1)=(x^2+u_{11}x+u_{10},v_{11}x+v_{10})=[1/2]D_2$

$\gcd(h,U_i)=1,\ i=1,2$

1. Solve $k_1h_2+k_1^2u_{21}+c_3=0$

$c_3\leftarrow f_4+u_{20},\ \alpha\leftarrow h_2,\ \gamma\leftarrow c_3/(u_{21}\alpha^2)$

$k_1\leftarrow \mathrm{H}(\gamma)\alpha,\ k_1'\leftarrow k_1+\alpha$

2. Select correct k_1 by solving $k_1h_0+k_0h_1+k_0^2+c_1=0$

$c_2\leftarrow f_3+c_3u_{20},\ c_1\leftarrow f_2+h_2v_{20}+c_2u_{20},\ c_0\leftarrow f_1+h_1v_{20}+c_1u_{20}$

$\alpha\leftarrow h_1,\ \gamma\leftarrow(c_1+k_1h_0)/\alpha^2$

if $\mathrm{Tr}(\gamma)=1$ then $k_1\leftarrow k_1',\ \gamma\leftarrow(c_1+k_1h_0)/\alpha^2$

$k_0\leftarrow \mathrm{H}(\gamma)\alpha,\ k_1'\leftarrow k_1+\alpha$

3. Select correct k_0 by checking trace of $xh_2+x^2u_{11}+1=0$

$u_{11}\leftarrow\sqrt{k_1h_1+k_0h_2+k_1^2u_{20}+c_2},\ \gamma\leftarrow u_{11}/h_2^2$

if $\mathrm{Tr}(\gamma)=1$ then

$k_0\leftarrow k_0',\ u_{11}\leftarrow\sqrt{k_1h_1+k_0h_2+k_1^2u_{20}+c_2}$

4. Compute U_1

$u_{10}\leftarrow\sqrt{k_0h_0+k_0^2u_{20}+c_0}$

5. Compute $V_1=V_2+h+kU_2 \bmod U_1$

$w\leftarrow h_2+k_1$

$v_{11}\leftarrow h_1+k_1u_{20}+k_0+u_{11}w$

$v_{10}\leftarrow v_{20}+h_0+k_0u_{20}+u_{10}w$

6. $U_1(x)\leftarrow x^2+u_{11}x+u_{10},\ V_1(x)\leftarrow v_{11}x+v_{10}$

7. return $D_1=(U_1,V_1)$

【０１２５】

［処理例２：提案法Ｆ１］

処理例２（提案法Ｆ１）は、種数２，標数２のｈ（ｘ）＝ｘ^{2}＋ｈ$_{1}$ｘ＋ｈ$_{0}$、ｆ$_{4}$＝０をパラメータに持つような超楕円曲線において因子Ｄの１／２倍算を計算する方法である。

【０１２６】

Ａｌｇｏｒｉｔｈｍ５を良く見ると、ｈ（ｘ）の係数との乗算、ｈ（ｘ）の係数の逆元演算が多くある。つまり、ｈ（ｘ）の係数を工夫することにより、乗算および逆元演算の計算量を削減することができる。なお、文献（非特許文献１９：T. Lange. Efficient Doubling on Genus Two Curves over Binary Fields, SAC 2004, pre-proceedings, pp.189-202, 2004.）では、高速化のため、ｈ$_{2}$＝１、ｆ$_{4}$＝０を用いている。このパラメータを用いた場合のＨａｒｌｅｙＤＢＬの計算量は、２１Ｍ＋５Ｓ＋１Ｉである。

【０１２７】
ここで説明する処理例２（提案法Ｆ１）もこれに条件を合わせることにするが、補題１より、ｈ（ｘ）は既約多項式を仮定しているため、
$h(x)=x^2+h_1x+h_0$、
$Tr(h_0/h_1^2)=1$ [0]
とする（二次方程式$ax^2+bx+c=0$が既約多項式であるための必要十分条件は$Tr(ac/b^2)=1$である）。この場合の計算法をアルゴリズム８［Ａｌｇｏｒｉｔｈｍ　８］ＨＥＣ＿ＨＬＶ（$h_2=1$、$f_4=0$）に示す。
【０１２８】

【数１６】

Algorithm 8 $HEC\_HLV(h_2=1, f_4=0)$

$Input: D_2=(U_2,V_2),\ invu=1/u_{21},\ 1/h_1^2$

$Output: D_1=(U_1,V_1)=[1/2]D_2,\ invu=1/u_{11}$

$U_i(x)=x^2+u_{i1}x+u_{i0},\ V_i(x)=v_{i1}x+v_{i0},\ \gcd(h,U_i)=1, i=1,2$

1. Solve $k_1+k_1^2u_{21}+1=0$

   $\alpha \leftarrow invu,\ k_1 \leftarrow \mathrm{H}(u_{21})\alpha,\ k_1' \leftarrow k_1+\alpha$

2. Select correct k_1 by solving $k_1h_0+k_0h_1+k_0^2u_{21}+c_1=0$

   $c_1 \leftarrow f_3+v_{21}+u_{20}+u_{21}^2$

   $c_0 \leftarrow f_2+v_{20}+v_{21}(h_1+v_{21})+u_{21}(u_{20}+c_1)$

   $w_0 \leftarrow u_{21}/h_1^2,\ \alpha \leftarrow h_1\alpha,\ \gamma \leftarrow (c_1+k_1h_0)w_0$

   if $\mathrm{Tr}(\gamma)=1$ then $k_1 \leftarrow k_1',\ \gamma \leftarrow (c_1+k_1h_0)w_0$

   $k_0 \leftarrow \mathrm{H}(\gamma)\alpha,\ k_0' \leftarrow k_0+\alpha$

3. Select correct k_0 by checking trace of $x+x^2u_{11}+1=0$

   $w_0 \leftarrow k_1^2,\ w_1 \leftarrow w_0u_{20}+k_1h_1+u_{21}$

   $w_2 \leftarrow k_0+\sqrt{w_1+k_0},\ w_3 \leftarrow k_0'+\sqrt{w_1+k_0'},\ w_4 \leftarrow w_2w_3$

   $w_1 \leftarrow 1/(w_4k_1),\ w_4 \leftarrow w_1w_4,\ u_{11} \leftarrow w_2w_4$

   if $\mathrm{Tr}(\gamma)=1$ then

   $k_0 \leftarrow k_0',\ u_{11} \leftarrow w_3w_4,\ w_2 \leftrightarrow w_3$

   $invu \leftarrow w_0w_1w_3$

4. Compute U_1

   $w_1 \leftarrow k_0u_{20},\ w_5 \leftarrow k_1u_{21},\ w_6 \leftarrow (k_0+k_1)(u_{20}+u_{21})$

   $u_{10} \leftarrow w_4\sqrt{k_0(w_1+h_0)+c_0}$

5. Compute $V_1=V_2+h+kU_2 \bmod U_1$

   $w_4 \leftarrow w_5+k_0+1,\ w_5 \leftarrow w_1+w_5+w_6+v_{21}+h_1$

   $w_6 \leftarrow w_1+v_{20}+h_0,\ w_7 \leftarrow k_1u_{11}+w_4$

   $w_1 \leftarrow w_7u_{10},\ w_3 \leftarrow (k_1+w_7)(u_{10}+u_{11})$

   $v_{11} \leftarrow w_1+w_3+w_4+w_5+w_7$

   $v_{10} \leftarrow w_1+w_6$

6. $U_1(x) \leftarrow x^2+u_{11}x+u_{10},\ V_1(x) \leftarrow v_{11}x+v_{10}$

7. return $D_1=(U_1,V_1),\ invu$

【０１２９】

また、逆元演算の回数を削除するため、モンゴメリトリックと呼ばれる手法を用いている。これは、例えば３個の有限体の元ａ、ｂ、ｃの逆元を求めたい場合、まず、３つの元の積を求め、これの逆元をｗ＝１／（ａ＊ｂ＊ｃ）などと求める。次に、ａの逆元を求める場合は、ｗ＊ｂ＊ｃを計算する。ｂ、ｃの逆元も同様にして、ｗ＊ａ＊ｃ、ｗ＊ａ＊ｂとそれぞれ計算する。

【０１３０】
通常、逆元演算は、乗算の数倍程度（後で示すが、ソフトウェア実装を行った結果、逆元演算は乗算の８倍程度）の計算量である。そのため、例えば３つの元の逆元を求める場合、素直に３回逆元演算を行えば、Ｉ＝８Ｍと仮定すると、２４Ｍの計算量となる。逆に、前述のモンゴメリトリックを用いた場合は、Ｉ＋８Ｍ＝１６Ｍとなり、３回逆元演算を行うよりも高速である。
【０１３１】
ここで説明している処理例２（提案法Ｆ１）では、このモンゴメリトリックを用いて、u_{11}の逆元を求めている。u_{11}の逆元は、次回の１／２倍算の入力として与える。そのため、Ａｌｇｏｒｉｔｈｍ８は、［$1/2^{i}$］Ｄを計算することができ、因子Ｄのスカラー倍算を行う場合、ｒｉｇｈｔ－ｔｏ－ｌｅｆｔ法、つまり、［$1/2^{i}$］Ｄを加算する方法に適用できる。１／２倍算を用いたスカラー倍算については、後ほど述べる。また、この計算量は次の通りである。
（ａ）k_1，k_0が正しい値である場合：２４Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ
（ｂ）k_1，k_0'が正しい値である場合：２６Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ
（ｃ）k_1'，k_0が正しい値である場合：２５Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ
（ｄ）k_1'，k_0'が正しい値である場合：２７Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ
上記（ａ）～（ｄ）のすべてについて平均すると、２５．５Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔとなる。
【０１３２】
ＨａｒｌｅｙＤＢＬの計算量は、２１Ｍ＋５Ｓ＋１Ｉであった。ここで、文献［（非特許文献１５） E.Knudsen. Elliptic Scalar Multiplication Using Point Halving. ASIACRYPTO '99, LNCS 1716, pp.135-149, Springer-Verlag, 1999.］によると、有限体が正規基底で定義されている場合、Ｓ（２乗算），ＳＲ（平方根演算），Ｈ（ハーフトレース（二次方程式の根を求める演算）），Ｔ（トレース（二次方程式の根があるか否かの判定））の計算量は無視することができ、Ｍ（乗算），Ｉ（逆元演算）の計算量のみ考慮すればよいことが知られている。そのため、正規基底を用いた場合、Ａｌｇｏｒｉｔｈｍ８はＨａｒｌｅｙＤＢＬよりも４．５Ｍ低速となる。
【０１３３】
また、有限体が多項式基底で定義されている場合、文献［（非特許文献１６） K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/ corr2003-18.pdf１８］によると、通常ＳＲ（平方根演算），Ｈ（ハーフトレース（二次方程式の根を求める演算））の計算量は、Ｍ（乗算）との対比において、ＳＲ＝Ｈ＝０．５Ｍ程度であることが知られている。また、Ｔ（トレース（二次方程式の根があるか否かの判定））の計算量は無視できる。さらにＳ（２乗算）の計算量は、Ｍ（乗算）の数分の１程度であることが知られている。しかしながら、ＳＲの計算量は、多項式基底の選び方によっては、０．５Ｍよりも少なくなることも、同時に知られている。なお、例外ケースは、前述の処理例１（提案法Ａ１）の例外ケースに基いて計算できる。
【０１３４】
［処理例３（提案法Ｂ１）］
処理例３（提案法Ｂ１）は、種数２、標数２の$h(x)=x^{2}+x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法である。
【０１３５】
前述の処理例２（提案法Ｆ１）において説明したように、処理例１（提案法Ａ１）において説明した１／２倍算の計算アルゴリズム、すなわち、アルゴリズム５［Ａｌｇｏｒｉｔｈｍ ５ ＨＥＣ＿ＨＬＶ］を良く見ると、ｈ（ｘ）の係数との乗算、ｈ（ｘ）の係数の逆元演算が多くある。つまり、ｈ（ｘ）の係数を工夫することにより、乗算および逆元演算の計算量を削減することができる。文献［J.Pelzl, T.Wollinger, and C.Paar. High

Performance Arithmetic for Hyperelliptic Curve Cryptosystems of Genus Two. Cryptology ePrint Archive, 2003/212, IACR, 2003］では、高速化のために、h_2，$h_1 \in \{0, 1\}$，$f_4 = 0$を用いた例を示している。

【０１３６】

このパラメータを用いた場合の［ＨａｒｌｅｙＤＢＬ］の計算量は、

１８Ｍ＋７Ｓ＋１Ｉ

である。

【０１３７】

処理例３（提案法Ｂ１）もこれに条件を合わせることにするが、前述した［補題１］より、ｈ（ｘ）は既約多項式を仮定しているため、

$h(x) = x^2 + x + h_0$，

$Tr(h_0) = 1$

とする（二次方程式$ax^2 + bx + c = 0$が既約多項式であるための必要十分条件は$Tr(ac/b^2) = 1$である）。

【０１３８】

この場合の計算法示すアルゴリズムをアルゴリズム１０［Ａｌｇｏｒｉｔｈｍ１０　ＨＥＣ＿ＨＬＶ（$h_2 = h_1 = 1$，$f_4 = 0$）］として以下に示す。

【数１７】

Algorithm10 $HEC\_HLV(h_2 = h_1 = 1, f_4 = 0)$

Input : $D_2 = (U_2, V_2)$, $invu = 1/u_{21}$

Output : $D_1 = (U_1, V_1) = [1/2]D_2$, $invu = 1/u_{11}$

$U_i(x) = x^2 + u_{i1}x + u_{i0}$, $V_i(x) = v_{i1}x + v_{i0}$, $\gcd(h, U_i) = 1, i = 1, 2$

1. Solve $k_1 + k_1^2 u_{21} + 1 = 0$

   $\alpha \leftarrow invu$, $k_1 \leftarrow \mathrm{H}(u_{21})\alpha$, $k_1' \leftarrow k_1 + \alpha$

2. Select correct k_1 by solving $k_1 h_0 + k_0 + k_0^2 u_{21} + c_1 = 0$

   $c_1 \leftarrow f_3 + v_{21} + u_{20} + u_{21}^2$

   $c_0 \leftarrow f_2 + v_{20} + v_{21} + v_{21}^2 + u_{21}(u_{20} + c_1)$

   $\gamma \leftarrow (c_1 + k_1 h_0)u_{21}$

   if $\mathrm{Tr}(\gamma) = 1$ then $k_1 \leftarrow k_1'$, $\gamma \leftarrow \gamma + h_0$

   $k_0 \leftarrow \mathrm{H}(\gamma)\alpha$, $k_0' \leftarrow k_0 + \alpha$

3. Select correct k_0 by checking trace of $x + x^2 u_{11} + 1 = 0$

   $w_0 \leftarrow k_1^2$, $w_1 \leftarrow w_0 u_{20} + k_1 + u_{21}$

   $w_2 \leftarrow k_0 + \sqrt{w_1 + k_0}$, $w_3 \leftarrow k_0' + \sqrt{w_1 + k_0'}$, $w_4 \leftarrow w_2 w_3$

   $w_1 \leftarrow 1/(w_4 k_1), w_4 \leftarrow w_1 w_4$, $u_{11} \leftarrow w_2 w_4$

   if $\mathrm{Tr}(u_{11}) = 1$ then

   $k_0 \leftarrow k_0'$, $u_{11} \leftarrow w_3 w_4, w_2 \leftrightarrow w_3$

4. *Compute* U_1

   $w_0 \leftarrow w_0 w_1$, $w_1 \leftarrow k_0 u_{20}$, $w_5 \leftarrow k_1 u_{21}$, $w_6 \leftarrow (k_0 + k_1)(u_{20} + u_{21})$

   $u_{10} \leftarrow w_4 \sqrt{k_0(w_1 + h_0) + c_0}$

5. *Compute* $V_1 = V_2 + h + kU_2 \bmod U_1$

   $w_4 \leftarrow w_5 + k_0 + 1$, $w_5 \leftarrow w_1 + w_5 + w_6 + v_{21} + 1$, $w_6 \leftarrow w_1 + v_{20} + h_0$

   $invu \leftarrow w_0 w_3$

   $w_0 \leftarrow w_2 + w_4$, $w_1 \leftarrow w_0 u_{10}$, $w_3 \leftarrow (k_0 + k_1)(u_{10} + u_{11})$

   $v_{10} \leftarrow w_1 + w_2 + w_3 + w_5$

   $v_{11} \leftarrow w_1 + w_6$

6. $U_1(x) \leftarrow x^2 + u_{11}x + u_{10}$, $V_1(x) \leftarrow v_{11}x + v_{10}$

7. *return* $D_1 = (U_1, V_1)$, $invu$

【０１３９】

また、逆元演算の回数を削除するため、前述の処理例２（提案法Ｆ１）と同様、モンゴメリトリック手法を用いている。本処理例においてもモンゴメリトリックを用いて、ｕ１１の逆元を求めている。ｕ１１の逆元は、次回の１／２倍算の入力として与える。この処理例における計算量は次の通りである。

（ａ）ｋ１，ｋ０が正しい値である場合：１９Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

（ｂ）ｋ１，ｋ０'が正しい値である場合：２０Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

（ｃ）ｋ１'，ｋ０が正しい値である場合：１９Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

（ｄ）k_1'，k_0'が正しい値である場合：２０Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

【０１４０】

上記（ａ）～（ｄ）のすべての場合について平均すると、

１９．５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

となる。ＨａｒｌｅｙＤＢＬの計算量は、１８Ｍ＋７Ｓ＋１Ｉであった。ここで、前述したように、文献［E.Knudsen. Elliptic Scalar Multiplication Using Point Halving. ASIACRYPTO '99, LNCS 1716, pp.135-149, Springer-Verlag, 1999.］によると、有限体が正規基底で定義されている場合、Ｓ（２乗算），ＳＲ（平方根演算），Ｈ（ハーフトレース（二次方程式の根を求める演算）），Ｔ（トレース（二次方程式の根があるか否かの判定））の計算量は無視することができ、Ｍ（乗算），Ｉ（逆元演算）の計算量のみ考慮すればよいことが知られている。

【０１４１】

そのため、正規基底を用いた場合、上述したアルゴリズム１０［Ａｌｇｏｒｉｔｈｍ１０］は、従来のアルゴリズム［ＨａｒｌｅｙＤＢＬ］よりも、１．５Ｍ低速となる。また、有限体が多項式基底で定義されている場合、文献［K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18.pdf］によると、通常ＳＲ，Ｈの計算量は、ＳＲ＝Ｈ＝０．５Ｍ程度であることが知られている。また、Ｔの計算量は無視できる。さらにＳの計算量は、Ｍの数分の１程度であることが知られている。しかしながら、ＳＲの計算量は、多項式基底の選び方によっては、０．５Ｍよりも少なくなることも同時に知られている。

【０１４２】

さて、前述のアルゴリズム１０［Ａｌｇｏｒｉｔｈｍ１０］の曲線にさらに、$h_0=1$の制約を設ける。前述のアルゴリズム１０［Ａｌｇｏｒｉｔｈｍ１０］は、h_0の乗算が１回あるので、$h_0=1$とすることにより、１Ｍの削減ができ、上記（ａ）～（ｄ）のすべての場合についての平均の計算量は、

１８．５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

となる。一方、ＨａｒｌｅｙＤＢＬは、計算量は、

１５Ｍ＋７Ｓ＋１Ｉ

である。なお、例外ケースは、前述の処理例１（提案法Ａ１）の例外ケースに基いて計算できる。

【０１４３】

［処理例４（提案法Ｅ１）］

処理例４（提案法Ｅ１）は、種数２、標数２のｈ（ｘ）＝ｘをパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法である。

【０１４４】

処理例３（提案法Ｂ１）と同様、アルゴリズム５（Ａｌｇｏｒｉｔｈｍ５）において、ｈ（ｘ）をｈ（ｘ）＝ｘとすることで，因子の１／２倍算に必要な有限体上の元の乗算および逆元演算の計算量を削減することができる。具体例として$f(x)=x^5+f_1x+f_0$の場合のアルゴリズムを、アルゴリズム１２（Ａｌｇｏｒｉｔｈｍ１２）として、以下に示す。

【数１８】

$$\text{Algorithm}\,12\,HEC\_HLV(h(x)=x, f(x)=x^5+f_1x+f_0)$$

$$Input: D_2=(U_2,V_2)$$

$$Output: D_1=(U_1,V_1)=[1/2]D_2$$

$$U_i(x)=x^2+u_{i1}x+u_{i0},\ V_i(x)=v_{i1}x+v_{i0},\ \gcd(h,U_i)=1, i=1,2$$

1. $\text{Solve}\, k_1^2u_{21}+1=0$

   $w_0 \leftarrow 1/u_{21},\ k_1 \leftarrow \sqrt{w_0}$

2. $\text{Solve}\, k_0+k_0^2u_{21}+c_1=0$

   $c_1 \leftarrow u_{20}+u_{21}^2,\ w_1 \leftarrow c_1u_{21}$

   $c_0 \leftarrow v_{21}+v_{21}^2+u_{21}u_{20}+w_1$

   $invk_1 \leftarrow \sqrt{u_{21}}, w_2 \leftarrow \mathrm{H}(w_1), w_3 \leftarrow w_2+1$

   $k_0 \leftarrow w_0w_2,\ k_0' \leftarrow k_0+w_0$

3. *Compute* U_1

   $u_{11} \leftarrow \sqrt{invk_1+k_0}, u_{10} \leftarrow \sqrt{(k_0+c_1)u_{20}+c_0u_{21}}$

   if $\mathrm{Tr}(u_{11}(u_{10}+invk_1+k_0))=1$ then

   $k_0 \leftarrow k'_0, w_2 \leftarrow w_3, u_{11} \leftarrow u_{11}+k_1, u_{10} \leftarrow u_{10}+\sqrt{w_0+u_{20}}$

4. *Compute* $V_1=V_2+h+kU_2 \bmod U_1$

   $w_1 \leftarrow k_1(u_{21}+u_{11})+k_0$

   $v_{11} \leftarrow k_1(u_{20}+u_{10})+w_2+v_{21}+1+u_{11}w_1$

   $v_{10} \leftarrow k_0u_{20}+v_{20}+u_{10}w_1$

5. $U_1(x) \leftarrow x^2+u_{11}x+u_{10}, V_1(x) \leftarrow v_{11}x+v_{10}$

6. *return* $D_1=(U_1,V_1)$

【０１４５】

前述の処理例３（提案法Ｂ１）と同様に、上記のアルゴリズム１２（Ａｌｇｏｒｉｔｈｍ１２）について計算量を評価する。処理例３（提案法Ｂ１）と異なり、ｈ（ｘ）＝ｘのタイプの超楕円曲線の場合、k_1はｓｔｅｐ．１において一意に決まるので、k_0のみ選択するステップが存在する（Ｓｔｅｐ．３）。計算量の少ないＢｅｓｔ　ｃａｓｅはｓｔｅｐ．３のｉｆ文のＴｒａｃｅが０の場合であり、Ｗｏｒｓｔ　ｃａｓｅはＴｒａｃｅが１の場合である。どちらも等確率で起こるため、平均の計算量は、

１１．５Ｍ＋２Ｓ＋１Ｉ＋４．５ＳＲ＋１Ｈ＋１Ｔ

である。この計算量は前述の処理例３よりも少なく高速に演算できる。なお、例外ケースは、前述の処理例１（提案法Ａ１）の例外ケースに基いて計算できる。

【０１４６】

［処理例５（提案法Ｃ１）］

処理例５（提案法Ｃ１）は、種数２、標数２のランダムパラメータを持つような超楕円曲線、$h(x)=x^2+h_1x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線、および$h(x)=x^2+x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する際に、１／２倍したものの候補が２つ出てくる。２つの候補のうち、正しい値を選ぶ必要があるが、選ぶ際に、有限体のトレース、乗算、平方根を計算する必要がある。どちらが正しいかは、因子Ｄに依存する。そのため、因子Ｄが固定されている場合、事前に２つの候補のうち、どちらが正しいかをテーブルに保持し、正しい値を選ぶ際に、このテーブルを参照することにより、上記の余計な計算を省略する方法である。

【０１４７】

k_1，k_1'（k_0，k_0'）のどちらが正しいかは、入力因子Ｄに依存している。したがって、Ｄが固定されている場合、例えばＥＣＤＨ型鍵交換のフェーズ１や、ＥＣＤＳＡ型署名生成、検証など、ベースポイントがあらかじめ決まっている場合、事前に、［$1/2^i$］Ｄを計算し、k_1，k_1'（k_0，k_0'）のどちらが正しい値であるかを、テーブルに記録しておく。

【０１４８】

例えば、ベースポイントの位数と同じビットサイズのテーブルをT_1，T_0と二つ用意し、これらを２進数表現したものを、

$T_1=(t_{1,r-1},\ldots,t_{1,0})$

$T_0=(t_{0,r-1},\ldots,t_{0,0})$

とする。

【０１４９】

［$1/2^i$］Ｄを求める際に、

k_1が正しければ、$t_{1,i}=0$、そうではなくて、k_1'が正しければ$t_{1,i}=1$

k_0が正しければ$t_{0,i}=0$、そうではなくて、k_0'が正しければ$t_{0,i}=1$

などとテーブルに記録すれば、テーブルサイズは、わずかベースポイントの位数のサイズの２倍程度のビット列で済む。このテーブルを参照することにより、１／２倍算の計算量を削減することができる。

【０１５０】

この方法をアルゴリズム８［Ａｌｇｏｒｉｔｈｍ ８］ ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）に適用したものを、アルゴリズム９［Ａｌｇｏｒｉｔｈｍ ９］ ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$，ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）として示す。計算量は、２２Ｍ＋２Ｓ＋１Ｉ＋２ＳＲ＋２Ｈとなる。

【０１５１】

【数１９】

Algorithm 9 $HEC\_HLV(h_2=1, f_4=0,$ table-lookup)

Input : $D_2=(U_2,V_2)$, $invu=1/u_{21}$, $1/h_1^2$, t_0, t_1

Output : $D_1=(U_1,V_1)=[1/2]D_2$, $invu=1/u_{11}$

$U_i(x)=x^2+u_{i1}x+u_{i0}$, $V_i(x)=v_{i1}x+v_{i0}$, $\gcd(h,U_i)=1, i=1,2$

1. Solve $k_1+k_1^2u_{21}+1=0$

   if $t_1=0$ then $k_1 \leftarrow \mathrm{H}(u_{21})invu$ else $k_1 \leftarrow \mathrm{H}(u_{21})invu+invu$

2. Solve $k_1h_0+k_0h_1+k_0^2u_{21}+c_1=0$

   $c_1 \leftarrow f_3+v_{21}+u_{20}+u_{21}^2$

   $c_0 \leftarrow f_2+v_{20}+v_{21}(h_1+v_{21})+u_{21}(u_{20}+c_1)$

   $\alpha \leftarrow h_1invu$, $\gamma \leftarrow (c_1+k_1h_0)u_{21}/h_1^2$

   if $t_0=0$ then $k_0 \leftarrow \mathrm{H}(\gamma)\alpha$ else $k_0 \leftarrow \mathrm{H}(\gamma)\alpha+\alpha$

3. Compute U_1

   $w_0 \leftarrow k_1^2$, $w_1 \leftarrow w_0u_{20}+k_1h_1+u_{21}$, $w_2 \leftarrow k_0+\sqrt{w_1+k_0}$

   $w_1 \leftarrow 1/(w_2k_1)$, $w_3 \leftarrow w_1w_2$

   $u_{11} \leftarrow w_2w_3$, $invu \leftarrow w_0w_1$

   $w_1 \leftarrow k_0u_{20}$, $w_5 \leftarrow k_1u_{21}$, $w_6 \leftarrow (k_0+k_1)(u_{20}+u_{21})$

   $u_{10} \leftarrow w_3\sqrt{k_0(w_1+h_0)+c_0}$

4. Compute $V_1=V_2+h+kU_2 \bmod U_1$

   $w_4 \leftarrow w_5+k_0+1$, $w_5 \leftarrow w_1+w_5+w_6+v_{21}+h_1$

   $w_6 \leftarrow w_1+v_{20}+h_0$, $w_7 \leftarrow k_1u_{11}+w_4$

   $w_3 \leftarrow (k_1+w_7)(u_{10}+u_{11})$, $w_1 \leftarrow w_7u_{10}$

   $v_{11} \leftarrow w_1+w_3+w_4+w_5+w_7$

   $v_{10} \leftarrow w_1+w_6$

6. $U_1(x) \leftarrow x^2+u_{11}x+u_{10}$, $V_1(x) \leftarrow v_{11}x+v_{10}$

7. return $D_1=(U_1,V_1)$, $invu$

【０１５２】

具体的に，この方法を上述のアルゴリズム１０［Ａｌｇｏｒｉｔｈｍ１０　ＨＥＣ＿ＨＬＶ　（ｈ２＝ｈ１＝１，ｆ４＝０）］に適用したアルゴリズムをアルゴリズム１１［Ａｌｇｏｒｉｔｈｍ１１　ＨＥＣ＿ＨＬＶ（ｈ２＝ｈ１＝１，ｆ４＝０，ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）］として、以下に示す。

【０１５３】

【数２０】

Algorithm11 $HEC\_HLV(h_2=h_1=1, f_4=0,$ with table-lookup)

$Input: D_2=(U_2,V_2),\ invu=1/u_{21}\ , t_0, t_1$

$Output: D_1=(U_1,V_1)=[1/2]D_2,\ invu=1/u_{11}$

$U_i(x)=x^2+u_{i1}x+u_{i0}, V_i(x)=v_{i1}x+v_{i0}, \gcd(h,U_1)=1$

1. Solve $k_1+k_1^2u_{21}+1=0$ and select correct k_1 via t_1

$\alpha \leftarrow invu,\ k_1 \leftarrow \mathrm{H}(u_{21})\alpha$

if $t_1=0$ then $k_1 \leftarrow k_1+\alpha$

2. Solving $k_1h_0+k_0+k_0^2u_{21}+c_1=0$ and select correct k_0 via t_0

$c_1 \leftarrow f_3+v_{21}+u_{20}+u_{21}^2$

$c_0 \leftarrow f_2+v_{20}+v_{21}+v_{21}^2+u_{21}(u_{20}+c_1)$

$\gamma \leftarrow (c_1+k_1h_0)u_{21}$

$k_0 \leftarrow \mathrm{H}(\gamma)\alpha$

if $t_1=0$ then $k_0 \leftarrow k_0+\alpha$

3. Compute U_1

$w_0 \leftarrow k_1^2,\ w_1 \leftarrow w_0u_{20}+k_1+u_{21}$

$w_2 \leftarrow k_0+\sqrt{w_1+k_0}$

$w_1 \leftarrow 1/(w_2k_1),\ w_4 \leftarrow w_1w_2$

$u_{11} \leftarrow w_2w_4$

$w_0 \leftarrow w_0w_1,\ w_1 \leftarrow k_0u_{20},\ w_5 \leftarrow k_1u_{21},\ w_6 \leftarrow (k_0+k_1)(u_{20}+u_{21})$

$u_{10} \leftarrow w_4\sqrt{k_0(w_1+h_0)+c_0}$

5. *Compute* $V_1=V_2+h+kU_2 \bmod U_1$

$w_4 \leftarrow w_5+k_0+1,\ w_5 \leftarrow w_1+w_5+w_6+v_{21}+1,\ w_6 \leftarrow w_1+v_{20}+h_0$

$invu \leftarrow w_0$

$w_0 \leftarrow w_2+w_4,\ w_1 \leftarrow w_0u_{10},\ w_3 \leftarrow (k_0+k_1)(u_{10}+u_{11})$

$v_{10} \leftarrow w_1+w_2+w_3+w_5$

$v_{11} \leftarrow w_1+w_6$

6. $U_1(x) \leftarrow x^2+u_{11}x+u_{10}, V_1(x) \leftarrow v_{11}x+v_{10}$

7. *return* $D_1=(U_1,V_1)\ , invu=1/u_{11}$

【０１５４】

上記アルゴリズム１１［Ａｌｇｏｒｉｔｈｍ１１　ＨＥＣ＿ＨＬＶ（ｈ$_2$＝ｈ$_1$＝１，ｆ$_4$＝０，ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）］における計算量は、

１８Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈとなり、さらにｈ$_0$＝１とすることにより、１Ｍの削減ができ、この場合の計算量は、

１７Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈとなる。

【０１５５】

さらにこの方法を、前述のアルゴリズム１２［Ａｌｇｏｒｉｔｈｍ１２　ＨＥＣ＿ＨＬ

Ｖ（ｈ（ｘ）＝ｘ，ｆ（ｘ）＝ｘ^{5}＋ｆ$_{1}$ｘ＋ｆ$_{0}$］に適用したアルゴリズムを、アルゴリズム１３［Ａｌｇｏｒｉｔｈｍ１３　ＨＥＣ＿ＨＬＶ（ｈ（ｘ）＝ｘ，ｆ（ｘ）＝ｘ^{5}＋ｆ$_{1}$ｘ＋ｆ$_{0}$，　ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）として、以下に示す。

【０１５６】

【数２１】

Algorithm13 $HEC\_HLV(h(x)=x, f(x)=x^5+f_1x+f_0,$ with table-lookup)

$Input: D_2=(U_2,V_2),\ t_0$

$Output: D_1=(U_1,V_1)=[1/2]D_2$

$U_i(x)=x^2+u_{i1}x+u_{i0},\ V_i(x)=v_{i1}x+v_{i0},\ \gcd(h,U_1)=1$

1. Solve $k_1^2u_{21}+1=0$

   $w_0 \leftarrow 1/u_{21},\ k_1 \leftarrow \sqrt{w_0}$

2. Solve $k_0+k_0^2u_{21}+c_1=0$

   $c_1 \leftarrow u_{20}+u_{21}^2,\ w_1 \leftarrow c_1u_{21}$

   $c_0 \leftarrow v_{21}+v_{21}^2+u_{21}u_{20}+w_1$

   $invk_1 \leftarrow \sqrt{u_{21}},\ w_2 \leftarrow \mathrm{H}(w_1)$

   if $t_0=0$ then

   $k_0 \leftarrow w_0w_2$

   else

   $k_0 \leftarrow w_0w_2+w_0,\ w_2 \leftarrow w_2+1$

3. *Compute* U_1

   $u_{11} \leftarrow \sqrt{invk_1+k_0},\ u_{10} \leftarrow \sqrt{(k_0+c_1)u_{20}+c_0u_{21}}$

4. *Compute* $V_1=V_2+h+kU_2 \bmod U_1$

   $w_1 \leftarrow k_1(u_{21}+u_{11})+k_0$

   $v_{11} \leftarrow k_1(u_{20}+u_{10})+w_2+v_{21}+1+u_{11}w_1$

   $v_{10} \leftarrow k_0u_{20}+v_{20}+u_{10}w_1$

5. $U_1(x) \leftarrow x^2+u_{11}x+u_{10},\ V_1(x) \leftarrow v_{11}x+v_{10}$

6. *return* $D_1=(U_1,V_1)$

【０１５７】

上記アルゴリズムにおける計算量は、

９．５Ｍ＋２Ｓ＋１Ｉ＋３．５ＳＲ＋１Ｈ

であり、さらに高速化が実現される。

【０１５８】

［処理例６（提案法Ｄ１）］

処理例６（提案法Ｄ１）は、上記処理例１～５に示す因子の１／２倍算の計算方法を用いて、因子のスカラー倍算を計算する方法である。

【０１５９】

楕円曲線での有理点の１／２倍算を用いたスカラー倍算の計算方法は、文献［E.Knudsen. Elliptic Scalar Multiplication Using Point Halving. ASIACRYPTO '99, LNCS 1716, pp.135-149, Springer-Verlag, 1999.］にある。超楕円曲線の因子の１／２倍算を用い

たスカラー倍算の計算方法は，この文献にあるスカラー倍算に基づいて実行される。ただし、［１／２^{i}］Ｄを計算し、加算していくｒｉｇｈｔ－ｔｏ－ｌｅｆｔ法を用いる。このアルゴリズムを、アルゴリズム１４［Ａｌｇｏｒｉｔｈｍ１４　Ｓｃａｌａｒ　Ｍｕｌｔｉｐｌｉｃａｔｉｏｎ］として以下に示す。

【数２２】

Algorithm14 *Scalar Multiplication*

Input : $D \in \mathbf{J}(\mathrm{F}_{2^n})$ *such that* $2D \neq O$, $d \in \mathrm{Z}$, r : order of D, $m = \lfloor \log_2 r \rfloor$

Output : *scalar multiplication* dD

1. $\sum_{i=0}^{m} \hat{d}_i 2^i \leftarrow 2^m d \mod r$, $\hat{d}_i \in \{0,1\}$
2. $\sum_{i=0}^{m} d_i / 2^i \leftarrow \sum_{i=0}^{m} \hat{d}_i 2^{i-m}$, $d_i \in \{0,1\}$
3. $Q \leftarrow O$, $R \leftarrow D$, $invu \leftarrow 1/u_1$
4. for i from 0 to m do:
   if $d_i = 1$ *then* $Q \leftarrow Q + R$
   $(R, invu) \leftarrow HEC\_HLV(R, invu)$
5. return Q

【０１６０】

上記アルゴリズム１４［Ａｌｇｏｒｉｔｈｍ１４　Ｓｃａｌａｒ　Ｍｕｌｔｉｐｌｉｃａｔｉｏｎ］のステップ４に出てくるＨＥＣ＿ＨＬＶは、ランダムカーブを用いた前述のアルゴリズム５［Ａｌｇｏｒｉｔｈｍ５］のＨＥＣ＿ＨＬＶでも良いし、アルゴリズム８［Ａｌｇｏｒｉｔｈｍ　８］の曲線パラメータにｈ$_{2}$＝１，ｆ$_{4}$＝０の制約を設けたＨＥＣ＿ＨＬＶでも良いし、アルゴリズム８［Ａｌｇｏｒｉｔｈｍ　８］の曲線パラメータにテーブル参照法を適用したＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０［Ａｌｇｏｒｉｔｈｍ　１０］の曲線パラメータにｈ$_{2}$＝ｈ$_{1}$＝１，ｆ$_{4}$＝０の制約を設けたＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０［Ａｌｇｏｒｉｔｈｍ　１０］の曲線パラメータにｈ$_{2}$＝ｈ$_{1}$＝ｈ$_{0}$＝１，ｆ$_{4}$＝０の制約を設けたＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０［Ａｌｇｏｒｉｔｈｍ　１０］に対してテーブル参照法を適用したＨＥＣ＿ＨＬＶでもよい。またアルゴリズム１２［Ａｌｇｏｒｉｔｈｍ　１２］の曲線パラメータのＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１２［Ａｌｇｏｒｉｔｈｍ　１２］に対してテーブル参照法を適用したＨＥＣ＿ＨＬＶでもよい。

【０１６１】

［演算高速化の検証］

次に、上述した処理例１～６を適用した演算における計算量を求め、演算の高速化について検証する。

【０１６２】

ＨＥＣ＿ＨＬＶ（ｈ$_{2}$＝１，ｆ$_{4}$＝０）では、計算量は、平均で、

２５．５Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

である。

【０１６３】

まず、有限体が正規基底で定義されている場合について考える。前述のとおり、正規基底を用いた場合、Ｍ，Ｉのみの計算量を考慮すればよい。文献［A.Menezes. Elliptic Cu

rve Public Key Cryptosystems. Kluwer Academic Publishers, 1993.］によると、有限体をＦｑ，ｑ＝２ｎとした場合、逆元計算一回は、下式によって算出される回数、すなわち、

【数２３】

$$\lfloor \log_2(n-1) \rfloor + w(n-1) - 1$$

【０１６４】

上記式によって算出される回数の乗算に匹敵する。ここで、ｗ（ｎ－１）はｎ－１を２進数で表示した際の１の個数を表す。例えば、ｎ＝８３，８９，１１３の場合、Ｉ＝８Ｍとなり、ｎ＝１０３の場合、Ｉ＝９Ｍとなる。

【０１６５】

ここで、Ｉ＝８Ｍを仮定すると、

ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０）

の計算量は、

２５．５Ｍ＋１Ｉ＝３３．５Ｍとなる。

【０１６６】

一方、ＨａｒｌｅｙＤＢＬでは、

２１Ｍ＋１Ｉ＝２９Ｍとなり、ＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも１３％ほど高速となる。また、テーブル参照法を用いた場合では、

２２Ｍ＋１Ｉ＝３０Ｍ

となり、ＨａｒｌｅｙＤＢＬは、ＨＥＣ＿ＨＬＶよりも３％ほど高速となる。

【０１６７】

また、ＨＥＣ＿ＨＬＶ（h_2＝h_1＝１，f_4＝０）

の計算量は、平均で、

１９．５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

となる。この場合は、

１９．５Ｍ＋１Ｉ＝２７．５Ｍ

となる。

【０１６８】

一方、ＨａｒｌｅｙＤＢＬでは、

１８Ｍ＋１Ｉ＝２６Ｍとなり、ＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも５％ほど高速となる。また、テーブル参照法を用いた場合では、
１８Ｍ＋１Ｉ＝２６Ｍ
となり、ＨａｒｌｅｙＤＢＬとＨＥＣ＿ＨＬＶは同等の計算量となる。

【０１６９】
また、ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）
の計算量は、平均で、
１８．５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ
となる。この場合は、
１８．５Ｍ＋１Ｉ＝２６．５Ｍ
となる。

【０１７０】
一方、ＨａｒｌｅｙＤＢＬでは、
１５Ｍ＋１Ｉ＝２３Ｍとなり、ＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも１３％ほど高速となる。また、テーブル参照法を用いた場合では、
１７Ｍ＋１Ｉ＝２５Ｍ
となり、ＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも８％ほど高速となる。

【０１７１】
また、多項式基底の場合をソフトウェア実装を行い、速度の比較を行った。
ＣＰＵは　ＰｅｎｔｉｕｍＩＩ　３００ＭＨｚ、
ＯＳはＲｅｄＨａｔ７．３、
コンパイラはｇｃｃ２．９６
の環境でソフトウェア実装を行った。

【０１７２】
Ｍ（乗算）、Ｓ（２乗算）は、文献［D.Hankerson, J.Hernandez, and A.Menezes. Software Implementation of Elliptic Curve Cryptography over Binary Fields. CHES 2000, LNCS 1965, pp.1-24, 2000.，Ａｌｇｏｒｉｔｈｍ４．６，４．７］、Ｉ（逆元演算）は、文献［S.Shantz. From Euclid's GCD to Montgomery Multiplication to the Great Divide. TR-2001-95, Sun Microsystems, Inc., 2001.］、ＳＲ（平方根演算）、Ｔ（トレース（二次方程式の根があるか否かの判定））は、文献［K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18.pdf］、Ｈ（ハーフトレース（二次方程式の根を求める演算））は、文献［K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18.pdf　Ａｌｇｏｒｉｔｈｍ　４．７］　に従った。

【０１７３】
ｎ＝８３，８９，１１３の３つの有限体に対して，Ｍ，Ｓ，Ｉ，ＳＲ，Ｈ，Ｔを実装し、Ｍとの比を求めた。ここで、
ｎ＝８３の既約多項式は、
$z^{83}+z^7+z^4+z^2+1=0$
ｎ＝８９の既約多項式は、
$z^{89}+z^{38}+1=0$
ｎ＝１１３の既約多項式は、
$z^{113}+z^9+1=0$
を用いた。
計算量は次の通りとなった。
ｎ＝８３：Ｓ／Ｍ＝０．１２，Ｉ／Ｍ＝７．９６，ＳＲ／Ｍ＝０．５７，Ｈ／Ｍ＝０．５８
ｎ＝８９：Ｓ／Ｍ＝０．０５，Ｉ／Ｍ＝８．７４，ＳＲ／Ｍ＝０．１４，Ｈ／Ｍ＝０．

６１
ｎ＝１１３：Ｓ／Ｍ＝０．０６，Ｉ／Ｍ＝８．５６，ＳＲ／Ｍ＝０．１０，Ｈ／Ｍ＝０．５０
【０１７４】
これらを、ＨａｒｌｅｙＤＢＬの計算量２１Ｍ＋５Ｓ＋１Ｉに適用すると次のようになる。
ｎ＝８３：ＨａｒｌｅｙＤＢＬ２９．５６Ｍ
ｎ＝８９：ＨａｒｌｅｙＤＢＬ２９．９９Ｍ
ｎ＝１１３：ＨａｒｌｅｙＤＢＬ２９．８６Ｍ
【０１７５】
これらを、ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０）の計算量、２５．５Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０）３６．５７Ｍ
ｎ＝８９：ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０）３５．９８Ｍ
ｎ＝１１３：ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０）３５．４８Ｍ
【０１７６】
この場合、ｎ＝８３，８９，１１３で、それぞれＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも２０％、１７％、１６％程度、高速である。
【０１７７】
さらに、ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０）にテーブル参照法を適用した場合の計算量、２２Ｍ＋２Ｓ＋１Ｉ＋２ＳＲ＋２Ｈに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０　ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）３２．５Ｍ
ｎ＝８９：ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０　ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）３２．３４Ｍ
ｎ＝１１３：ＨＥＣ＿ＨＬＶ（h_2＝１，f_4＝０　ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）３１．８８Ｍ
【０１７８】
この場合、ｎ＝８３，８９，１１３で、それぞれＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも９％、７％、６％程度、高速である。
【０１７９】
また、h_2＝h_1＝１，f_4＝０の場合は、次の通りとなる。
ＨａｒｌｅｙＤＢＬの計算量１８Ｍ＋７Ｓ＋１Ｉに適用すると次のようになる。
ｎ＝８３：ＨａｒｌｅｙＤＢＬ２７．４Ｍ
ｎ＝８９：ＨａｒｌｅｙＤＢＬ２７．０９Ｍ
ｎ＝１１３：ＨａｒｌｅｙＤＢＬ２６．９８Ｍ
【０１８０】
次に、ＨＥＣ＿ＨＬＶ（h_2＝h_1＝１，f_4＝０）の計算量、１９．５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（h_2＝h_1＝１，f_4＝０）３０．６９Ｍ
ｎ＝８９：ＨＥＣ＿ＨＬＶ（h_2＝h_1＝１，f_4＝０）３０．０３Ｍ
ｎ＝１１３：ＨＥＣ＿ＨＬＶ（h_2＝h_1＝１，f_4＝０）２９．５４Ｍ
【０１８１】
この場合、ｎ＝８３，８９，１１３で、それぞれＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも１３％、１２％、９％程度、高速である。
【０１８２】
さらに、ＨＥＣ＿ＨＬＶ（h_2＝h_1＝１，f_4＝０）にテーブル参照法を適用した場合の計算量、１８Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（h_2＝h_1＝１，f_4＝０　ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）２８．６２Ｍ

n＝８９：ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$ ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）２８．３９Ｍ
n＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$ ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）２７．９４Ｍ
【０１８３】
この場合、ｎ＝８３，８９，１１３で、それぞれＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも４％、５％、３％程度、高速である。
【０１８４】
また、$h_2=h_1=h_0=1$，$f_4=0$の場合は、次の通りとなる。
ＨａｒｌｅｙＤＢＬの計算量１５Ｍ＋７Ｓ＋１Ｉに適用すると次のようになる。
n＝８３：ＨａｒｌｅｙＤＢＬ２３．８Ｍ
n＝８９：ＨａｒｌｅｙＤＢＬ２４．０９Ｍ
n＝１１３：ＨａｒｌｅｙＤＢＬ２３．９８Ｍ
【０１８５】
次に、ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）の計算量、１８．５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔに適用すると次のようになる。
n＝８３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）２９．６９Ｍ
n＝８９：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）２９．０３Ｍ
n＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）２８．５４Ｍ
【０１８６】
この場合、ｎ＝８３，８９，１１３で、それぞれＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも２０％、１７％、１６％程度、高速である。
【０１８７】
さらに、ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）にテーブル参照法を適用した場合の計算量、１７Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈに適用すると次のようになる。
n＝８３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$ ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）２７．６２Ｍ
n＝８９：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$ ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）２７．３９Ｍ
n＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$ ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）２６．９４Ｍ
【０１８８】
この場合、ｎ＝８３，８９，１１３で、それぞれＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも１４％、１２％、１１％程度、高速である。
【０１８９】
次に、上述のアルゴリズム１２［Ａｌｇｏｒｉｔｈｍ１２］、すなわち、［Ａｌｇｏｒｉｔｈｍ１２ ＨＥＣ＿ＨＬＶ（$h(x)=x$，$f(x)=x^5+f_1x+f_0$）］と、テーブル参照法を用いたアルゴリズム１３［Ａｌｇｏｒｉｔｈｍ１３］、すなわち、［Ａｌｇｏｒｉｔｈｍ１３ ＨＥＣ＿ＨＬＶ（$h(x)=x$，$f(x)=x^5+f_1x+f_0$，ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）］について、ＨａｒｌｅｙＤＢＬとの比較を行う。
【０１９０】
［Ａｌｇｏｒｉｔｈｍ１２ ＨＥＣ＿ＨＬＶ（$h(x)=x$，$f(x)=x^5+f_1x+f_0$）］の計算量は、１１．５Ｍ＋２Ｓ＋１Ｉ＋４．５ＳＲ＋１Ｈ＋１Ｔ、
［Ａｌｇｏｒｉｔｈｍ１３ ＨＥＣ＿ＨＬＶ（$h(x)=x$，$f(x)=x^5+f_1x+f_0$，ｗｉｔｈ ｔａｂｌｅ－ｌｏｏｋｕｐ）］の計算量は、９．５Ｍ＋２Ｓ＋１Ｉ＋３．５ＳＲ＋１Ｈである。ＨａｒｌｅｙＤＢＬの計算量は、文献［（非特許文献１９：T. Lange. Efficient Doubling on Genus Two Curves over Binary Fields, SAC 2004, pre-proceedings, pp.189-202, 2004.）］によると、６Ｍ＋５Ｓ＋１Ｉである。前述のように、有限体が正規基底で定義されている場合、Ｓ（２乗算），ＳＲ（平方根演算），Ｈ（

ハーフトレース（二次方程式の根を求める演算）），Ｔ（トレース（二次方程式の根があるか否かの判定））の計算量は無視することができ、Ｍ（乗算），Ｉ（逆元演算）の計算量のみ考慮すればよい。

【０１９１】

従って、アルゴリズム１２［Ａｌｇｏｒｉｔｈｍ１２　ＨＥＣ＿ＨＬＶ（ｈ（ｘ）＝ｘ，ｆ（ｘ）＝$ｘ^{5}$＋$ｆ_{1}$ｘ＋$ｆ_{0}$）］の計算量は、

１１．５Ｍ＋２Ｓ＋１Ｉ＋４．５ＳＲ＋１Ｈ＋１Ｔ、

＝１１．５Ｍ＋１Ｉ

となり、

アルゴリズム１３［Ａｌｇｏｒｉｔｈｍ１３　ＨＥＣ＿ＨＬＶ（ｈ（ｘ）＝ｘ，ｆ（ｘ）＝$ｘ^{5}$＋$ｆ_{1}$ｘ＋$ｆ_{0}$，ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）］の計算量は、

９．５Ｍ＋２Ｓ＋１Ｉ＋３．５ＳＲ＋１Ｈ

＝９．５Ｍ＋１Ｉ

となる。

【０１９２】

ＨａｒｌｅｙＤＢＬの計算量は、

６Ｍ＋１Ｉ

となる。従って、ＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも高速となる。

【０１９３】

以上、説明したように、曲線パラメータ$ｈ_{2}$＝$ｈ_{1}$＝１、$ｆ_{4}$＝０の場合のＨＥＣ＿ＨＬＶにテーブル参照法を適用した場合の計算量１８Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈが、ＨａｒｌｅｙＤＢＬの計算量１８Ｍ＋７Ｓ＋１Ｉにほぼ等しく、同等の条件でＨＥＣ＿ＨＬＶとＨａｒｌｅｙＤＢＬとで比較した場合、最も高速なアルゴリズムである。

【０１９４】

次に、アルゴリズム１４［Ａｌｇｏｒｉｔｈｍ　１４　ｓｃａｌａｒ　ｍｕｌｔｉｐｌｉｃａｔｉｏｎ］を用いたスカラー倍算の計算量を考える。曲線パラメータは、同等な条件でＨａｒｌｅｙＤＢＬと比較して、最も高速な$ｈ_{2}$＝$ｈ_{1}$＝１、$ｆ_{4}$＝０を用い、さらに、ＨＥＣ＿ＨＶＬにはテーブル参照法を用いた方法でスカラー倍算の計算量を考える。

【０１９５】

アルゴリズム１４におけるステップ１、２がスカラー倍算全体に占める割合は非常に小さいため、これらの計算量は無視する。ここで、正規基底および多項式基底ともにｎ＝８３、８９、１１３の場合で計算量を考えてみる。また、ベースポイントの位数は、ｎ＝８３、８９、１１３に対してそれぞれ１６５ビット、１７７ビット、２２５ビットと仮定する。また、ステップ４の繰り返し部分では、ベースポイントの位数のビット数だけ繰り返す。因子の加算は文献［T.Lange, Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae. Cryptology ePrint Archive, 2002/121, IACR, 2002］を用いる。ただし、曲線パラメータは$ｈ_{2}$＝$ｈ_{1}$＝１、$ｆ_{4}$＝０とする。この場合の因子の加算に必要な計算量は２１Ｍ＋３Ｓ＋１Ｉである。スカラー値は２進数で表現した場合、０、１が同程度の割合で現れると仮定する。計算量は、（（加算の計算量）／２＋（１／２倍算ｏｒ２倍残の計算量））ｘ（ベースポイントの位数のビット数）で計算する。まずは、正規基底の場合から見てみる。Ｉ＝８Ｍと仮定する。

【０１９６】

$ｈ_{2}$＝$ｈ_{1}$＝１，$ｆ_{4}$＝０の場合、

ｎ＝８３：加算・２倍算：６６８２．５Ｍ

ｎ＝８９：加算・２倍算：７１６８．５Ｍ

ｎ＝１１３：加算・２倍算：９１１２．５Ｍ

【０１９７】

$ｈ_{2}$＝$ｈ_{1}$＝１，$ｆ_{4}$＝０の場合、

ｎ＝８３：加算・１／２倍算：６９３０Ｍ

ｎ＝８９：加算・１／２倍算：７４３４Ｍ

ｎ＝１１３：加算・１／２倍算：９４５０Ｍ
【０１９８】
$h_2=h_1=1$，$f_4=0$＋テーブル参照法の場合（加算・２倍算の場合と等しい計算量）
ｎ＝８３：加算・２倍算：６６８２．５Ｍ
ｎ＝８９：加算・２倍算：７１６８．５Ｍ
ｎ＝１１３：加算・２倍算：９１１２．５Ｍ
【０１９９】
次に、多項式の場合を考える。
$h_2=h_1=1$，$f_4=0$の場合、
ｎ＝８３：加算・２倍算：６９１３．５Ｍ
ｎ＝８９：加算・２倍算：７３６１．３Ｍ
ｎ＝１１３：加算・２倍算：９３３３Ｍ
【０２００】
$h_2=h_1=1$，$f_4=0$の場合、
ｎ＝８３：加算・１／２倍算：７４５６．３５Ｍ
ｎ＝８９：加算・１／２倍算：７８８１．８Ｍ
ｎ＝１１３：加算・１／２倍算：９９０９Ｍ
【０２０１】
$h_2=h_1=1$，$f_4=0$＋テーブル参照法の場合（加算・２倍算の場合と等しい計算量）
ｎ＝８３：加算・２倍算：７１１４．８Ｍ
ｎ＝８９：加算・２倍算：７５９１．５３Ｍ
ｎ＝１１３：加算・２倍算：９５４０Ｍ
【０２０２】
以上、説明したように、上述した本発明の処理例によれば、楕円曲線暗号の１／２倍算を、超楕円曲線暗号に拡張し、高速演算が実現される。超楕円曲線上の因子の演算を使う暗号処理演算において、処理の重い演算は因子のスカラー倍算である。上述した本発明の処理によりスカラー倍算を従来と同程度の速度で演算することができ、その結果、１／２倍算を用いても超楕円曲線暗号の処理を従来と同程度の速度で処理することがきる。
【０２０３】
次に、上述した各種処理例、すなわち、
（処理例１：提案法Ａ１）
（処理例２：提案法Ｆ１）
（処理例３：提案法Ｂ１）
（処理例５：提案法Ｃ１）
（処理例６：提案法Ｄ１）
をさらに高速化した処理例７～１１、すなわち、以下の処理例について説明する。
（処理例７：提案法Ａ２）：上記処理例（処理例１：提案法Ａ１）のさらなる高速化手法であり、種数２、標数２のランダムパラメータを持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法。
（処理例８：提案法Ｆ２）：上記処理例（処理例２：提案法Ｆ１）のさらなる高速化手法であり、種数２，標数２の$h(x)=x^2+h_1x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において因子Ｄの１／２倍算を計算する方法。
（処理例９：提案法Ｂ２）：上記処理例（処理例３：提案法Ｂ１）のさらなる高速化手法であり、種数２、標数２の$h(x)=x^2+x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法。
（処理例１０：提案法Ｃ２）：上記処理例（処理例５：提案法Ｃ１）のさらなる高速化手法であり、種数２、標数２のランダムパラメータを持つような超楕円曲線、$h(x)=x^2+h_1x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線、および$h(x)=x^2+$

$x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において、因子Dの１／２倍算を計算する際に、１／２倍したものの候補が２つ出てくる。２つの候補のうち、正しい値を選ぶ必要があるが、選ぶ際に、有限体のトレース、乗算、平方根を計算する必要がある。どちらが正しいかは、因子Dに依存する。そのため、因子Dが固定されている場合、事前に２つの候補のうち、どちらが正しいかをテーブルに保持し、正しい値を選ぶ際に、このテーブルを参照することにより、上記の余計な計算を省略する方法。

（処理例１１：提案法Ｄ２）：上記処理例（処理例６：提案法Ｄ１）のさらなる高速化手法であり、上記処理例７～１０に示す因子の１／２倍算の計算方法を用いて、因子のスカラー倍算を計算する方法。

【０２０４】

以下、各処理例について、順次、その詳細を説明する。

【０２０５】

［処理例７：提案法Ａ２］

処理例７（提案法Ａ２）は、前述した処理例（処理例１：提案法Ａ１）のさらなる高速化手法であり、種数２、標数２のランダムパラメータを持つような超楕円曲線において、因子Dの１／２倍算を計算する方法である。

【０２０６】

以下の各処理例においても、扱う因子は位数がｒのものとする。つまり、扱う因子は分岐点を持たない。入力因子を、

$D_2=(U_2, V_2)$、

$U_2=u_{22}x^2+u_{21}x+u_{20}$、

$V_2=v_{21}x+v_{20}$、

ここで、D_2のｗｅｉｇｈｔが２の場合は$u_{22}=1$、

D_2のｗｅｉｇｈｔが１の場合は$u_{22}=0$、$u_{21}=1$、$v_{21}=0$とする。

分岐点を持たないため、１／２倍算として、

$ExHarDBL^{1+1\to 2}$、$ExHarDBL^{2+2\to 1}$、$ExHarDBL^{2+2\to 2}$、および、ＨａｒｌｅｙＤＢＬの４つの逆演算を考えればよい。ＨａｒｌｅｙＤＢＬ以外は例外ケースである。

【０２０７】

ここで$ExHarDBL^{2+2\to 1}$は入力因子のｗｅｉｇｈｔが２で、出力因子のｗｅｉｇｈｔが１の場合を計算する方法である。また、$ExHarDBL^{2+2\to 2}$は入力因子のｗｅｉｇｈｔが２で、かつU_2の１次の項の係数が$u_{21}=0$を満たし、出力因子のｗｅｉｇｈｔが２の場合を計算する方法である。ただし、$ExHarDBL^{2+2\to 2}$はＨａｒｌｅｙＤＢＬで計算できるが、これの逆演算である１／２倍算は、例外ケースとなるため、ここでは、$ExHarDBL^{2+2\to 2}$を例外ケースとして扱う。

【０２０８】

これら、$ExHarDBL^{1+1\to 2}$、$ExHarDBL^{2+2\to 1}$、$ExHarDBL^{2+2\to 2}$、および、ＨａｒｌｅｙＤＢＬに対応する１／２倍算をそれぞれ、$ExHEC\_HLV^{2\to 1+1}$、$ExHEC\_HLV^{1\to 2+2}$、$ExHEC\_HLV^{2\to 2+2}$、およびＨＥＣ＿ＨＬＶとする。

【０２０９】

因子の１／２倍算を行なう場合、先に（処理例１：提案法Ａ１）において説明したように、まず、入力因子により、図２に示す通りに場合分けを行なう。入力因子のｗｅｉｇｈｔが２で、かつU_2の１次の項の係数が$u_{21}\neq 0$を満たす場合は、ＨＥＣ＿ＨＬＶで計算を行なう。また、入力因子のｗｅｉｇｈｔが２で、かつU_2の項の係数が$u_{21}=0$を満たす場合は、$ExHEC\_HLV^{2\to 2+2}$、もしくは、$ExHEC_HLV^{2\to 1+1}$で計算を行なう。また、入力因子のｗｅｉｇｈｔが１の場合は$ExHEC_HLV^{1\to 2+2}$で計算を行う。ＨＥＣ＿ＨＬＶのアルゴリズムについては、先に図３を参照して説明した通りである。

【０２１０】

因子の１／２倍算は、因子の２倍算を行うアルゴリズム、すなわち、以下に示す［Ａｌｇｏｒｉｔｈｍ　１　ＨａｒｌｅｙＤＢＬ］

【数２４】

Algorithm 1 *HarleyDBL*

Input : $D_1 = (U_1, V_1)$

Output : $D_2 = (U_2, V_2)$

$U_i(x) = x^2 + u_{i1}x + u_{i0}, V_i(x) = v_{i1}x + v_{i0}, \gcd(h, U_1) = 1$

1. $U_1' \leftarrow U_1^2$
2. $S \leftarrow h^{-1}(f + hV_1 + V_1^2)/U_1 \bmod U_1$
3. $V_1' \leftarrow SU_1 + V_1$
4. $U_2' \leftarrow (f + hV_1' + V_1'^2)/U'$
5. $U_2 \leftarrow \mathrm{MakeMonic}(U_2')$
6. $V_2 \leftarrow V_1' + h \bmod U_2$
7. return $D_2 = (U_2, V_2)$

【０２１１】

上記アルゴリズム１［Ａｌｇｏｒｉｔｈｍ　１　ＨａｒｌｅｙＤＢＬ］を逆から計算することにより実現する。Ａｌｇｏｒｉｔｈｍ　１のステップ６では、

$V_1' + h = (k_1 x + k_0) U_2 + V_2$

を満たすような多項式、

$k(x) = k_1 x + k_0$

が唯一存在する。

【０２１２】

これを、

$V_1' = h + (k_1 x + k_0) U_2 + V_2$

と式変形し、ステップ４に現れる式、

$(f + hV_1' + V_1'^2)$

に代入すると、ステップ４は、次のようになる。

$U_2' U_1' = f + h(kU_2 + V_2) + k^2 U_2^2 + V_2^2$ ・・・式（１）

上記式において、

（U_2，V_2）は分かっているので、式（１）から、ｋとU_1'との関係式が得られる。

ここで、

$U_2' = k_1^2 U_2$

であることに注意する。

【０２１３】

上記式（１）を展開して整理すると、次のようになる。

$U_1' = x^4 + ((k_1 h_2 + k_1^2 u_{21} + 1)/k_1^2) x^3 + ((k_1 h_1 + k_0 h_2 + k_1^2 u_{20} + k_0^2 + c_2)/k_1^2) x^2 + ((k_1 h_0 + k_0 h_1 + k_0^2 u_{21} + c_1)/k_1^2) x + (k_0 h_0 + k_0^2 u_{20} + c_0)/k_1^2$

・・・式（２）
ここで，
$c_2 = f_4 + u_{21}$
$c_1 = f_3 + h_2 v_{21} + u_{20} + c_2 u_{21}$
$c_0 = f_2 + h_2 v_{20} + h_1 v_{21} + v_{21}^2 + c_2 u_{20} + c_1 u_{21}$
を満たす。
【０２１４】
また、ステップ１より、
$U_1' = U_1^2$、
つまり、
$U_1' = x^4 + u_{11}^2 x^2 + u_{10}^2$
・・・式（３）
が成り立つ。
【０２１５】
上記式（２）、（３）の各係数を比較することにより、関係式を導き、この関係式を解くことにより、１／２倍算が計算できる。上記の処理手順を示すアルゴリズムを以下に、アルゴリズム４［Ａｌｇｏｒｉｔｈｍ　４　Ｓｋｅｔｃｈ　ＨＥＣ＿ＨＬＶ］として示す。
【数２５】

Algorithm 4 *Sketch HEC _ HLV*

Input : $D_2 = (U_2, V_2)$
Output : $D_1 = (U_1, V_1) = [1/2]D_2$
$U_i(x) = x^2 + u_{i1}x + u_{i0},\ V_i(x) = v_{i1}x + v_{i0},\ \gcd(h, U_i) = 1,\ i = 1, 2$

1. reconstruct k_0, k_1
   1.1 $V_1' \leftarrow V_2 + h + kU_2,\ k(x) = k_1 x + k_0$
   1.2 $U_1' \leftarrow (f + hV_1' + V_1'^2)/(k_1^2 U_2)$
   1.3 derive k_0, k_1 from $\mathrm{coeff}(U_1', 3) = 0, \mathrm{coeff}(U_1', 1) = 0$
2. compute u_{11} by substituting k_0, k_1 in $\mathrm{coeff}(U_1', 2)$
3. compute u_{10} by substituting k_0, k_1 in $\mathrm{coeff}(U_1', 0)$
4. $U_1 \leftarrow x^2 + u_{11}x + u_{10}$
5. $V_1 \leftarrow V_2 + h + kU_2 \bmod U_1$
6. return $D_1 = (U_1, V_1)$

【０２１６】
具体的には，次の関係式が得られる。

【数２６】

$$k_1 h_2 + k_1^2 u_{21} + 1 = 0 \qquad \cdots \text{式（４）}$$

$$k_1 h_0 + k_0 h_1 + k_0^2 u_{21} + c_1 = 0 \qquad \cdots \text{式（５）}$$

$$u_{11} = \sqrt{k_1 h_1 + k_0 h_2 + k_1^2 u_{20} + k_0^2 + c_2} / k_1 \qquad \cdots \text{式（６）}$$

$$u_{10} = \sqrt{k_0 h_0 + k_0^2 u_{20} + c_0} / k_1 \qquad \cdots \text{式（７）}$$

【０２１７】

これらの関係式から，正しいｋ$_0$，ｋ$_1$を計算する必要があるが、これは、次の補題によって計算できる。

［補題１］

ｈ（ｘ）を既約多項式であると仮定する。このとき、式（４），（５）を満たすｋ$_1$は唯一である。また、式（５）は、正しいｋ$_1$に対してのみ、根を持つ。また、Ａｌｇｏｒｉｔｈｍ４でのＩ／２倍された因子Ｄ$_1$を計算できるｋ$_0$は唯一存在する。また、以下に示す式、

ｘｈ$_2$＋ｘ^{2}ｕ$_{11}$＋１＝０

は、正しいｋ$_0$に対してのみ根を持つ。

【０２１８】

Ａｌｇｏｒｉｔｈｍ４に対して上記補題１を適用した。詳細な１／２倍算の計算方法を以下のアルゴリズム５ａ［Ａｌｇｏｒｉｔｈｍ　５ａ　ＨＥＣ＿ＨＬＶ］として示す。

【数２７】

Algorithm 5*aHEC_HLV*

Input : $D_2 = (U_2, V_2)$

Output : $D_1 = (U_1, V_1) = [1/2]D_2$

$U_i(x) = x^2 + u_{i1}x + u_{i0}$, $V_i(x) = v_{i1}x + v_{i0}$, $\gcd(h, U_i) = 1, i = 1, 2$

1. Solve $k_1 h_2 + k_1^2 u_{21} + 1 = 0$

   $invu \leftarrow 1/u_{21}$, $invh \leftarrow 1/h_2^2$, $\alpha \leftarrow h_2 invu$, $\gamma \leftarrow u_{21} invh$

   $k_1 \leftarrow \mathrm{H}(\gamma)\alpha$, $k_1' \leftarrow k_1 + \alpha$

2. Select correct k_1 by solving $k_1 h_0 + k_0 h_1 + k_0^2 u_{21} + c_1 = 0$

   $c_2 \leftarrow f_4 + u_{21}$, $c_1 \leftarrow f_3 + h_2 v_{21} + u_{20} + c_2 u_{21}$

   $c_0 \leftarrow f_2 + h_2 v_{20} + h_1 v_{21} + v_{21}^2 + c_2 u_{20} + c_1 u_{21}$

   $\alpha \leftarrow h_1 invu$, $w \leftarrow u_{21}/h_1^2$, $\gamma \leftarrow (c_1 + k_1 h_0)w$

   if $\mathrm{Tr}(\gamma) = 1$ then $k_1 \leftarrow k_1'$, $\gamma \leftarrow (c_1 + k_1 h_0)w$

   $k_0 \leftarrow \mathrm{H}(\gamma)\alpha$, $k_0' \leftarrow k_0 + \alpha$

3. Select correct k_0 by checking trace of $xh_2 + x^2 u_{11} + 1 = 0$

   $invk \leftarrow 1/k_1$, $u_{11} \leftarrow invk\sqrt{k_1(h_1 + k_1 u_{20}) + k_0(h_2 + k_0) + c_2}$, $\gamma \leftarrow u_{11} invh$

   if $\mathrm{Tr}(\gamma) = 1$ then

   $k_0 \leftarrow k_0'$, $u_{11} \leftarrow u_{11} + invk\sqrt{\alpha(h_2 + \alpha)}$

4. *Compute* U_1

   $u_{10} \leftarrow invk\sqrt{k_0(h_0 + k_0 u_{20}) + c_0}$

5. *Compute* $V_1 = V_2 + h + kU_2 \bmod U_1$

   $w \leftarrow h_2 + k_1(u_{11} + u_{21}) + k_0$

   $v_{11} \leftarrow v_{21} + h_1 + k_1(u_{10} + u_{20}) + k_0 u_{21} + u_{11} w$

   $v_{10} \leftarrow v_{20} + h_0 + k_0 u_{20} + w$

6. $U_1(x) \leftarrow x^2 + u_{11}x + u_{10}$, $V_1(x) \leftarrow v_{11}x + v_{10}$

7. *return* $D_1 = (U_1, V_1)$

【０２１９】

上記アルゴリズム５ａ［Ａｌｇｏｒｉｔｈｍ５ａ］では、ｋ$_1$'，ｋ$_0$'が正しい値（言い換えるとｋ$_1$，ｋ$_0$が正しくない値）である場合、計算量は、

２９Ｍ＋１Ｓ＋４Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

となる。ここでＭ，Ｓ，Ｉ，ＳＲ，Ｈ，Ｔはそれぞれ有限体上の乗算、２乗算、逆元演算、平方根演算、ハーフトレース（二次方程式の根を求める演算）、トレース（二次方程式の根があるか否かの判定）を意味する。このｋ$_1$'，ｋ$_0$'が正しい値である場合、計算量は最も多くなる。

【０２２０】

次にｋ$_1$，ｋ$_0$が正しい値（言い換えるとｋ$_1$'，ｋ$_0$'が正しくない値）である場合、計算量は最も少なくなり、ステップ２では２Ｍの計算量が削減でき、ステップ３では、２

Ｍ＋１ＳＲの計算量が削減できる。つまり、この場合の計算量は、
２５Ｍ＋１Ｓ＋４Ｉ＋２ＳＲ＋２Ｈ＋２Ｔとなり、計算量は最も少なくなる。
【０２２１】
次にk_1，k_0'が正しい値（言い換えるとk_1'，k_0が正しくない値）である場合，ステップ３では、２Ｍ＋１ＳＲの計算量が削減できる。つまり、この場合の計算量は、
２７Ｍ＋１Ｓ＋４Ｉ＋２ＳＲ＋２Ｈ＋２Ｔとなる。
【０２２２】
最後にk_1'，k_0が正しい値（言い換えるとk_1，k_0'が正しくない値）である場合、ステップ２では，２Ｍの計算量が削減できる。つまり、この場合の計算量は、
２７Ｍ＋１Ｓ＋４Ｉ＋３ＳＲ＋２Ｈ＋２Ｔとなる。
【０２２３】
上記の４つの場合が発生する確率を、計算機実験によって確認した結果、ほぼ同じ割合で発生していることが確認された。これ以降は、上記４つの場合が生じる確率は等しいとする。上記４つの場合の計算量の平均をとると、
２７Ｍ＋１Ｓ＋４Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ
となる。
【０２２４】
次に、例外ケースの、
$ExHEC\_HLV^{2\to 1+1}$、
$ExHEC\_HLV^{1\to 2+2}$、
$ExHEC\_HLV^{2\to 2+2}$、
について考える。これらが発生する確率は無視できる程度であるので、計算量の評価は行なわない。
【０２２５】
なお、これらの例外ケースの計算アルゴリズムについては、先の（処理例１：提案法Ａ１）の項目において説明した図４～図６に示すフローチャート、すなわち、
$ExHEC_HLV^{2\to 1+1}$のアルゴリズムについては図４に示すフローチャート、
$ExHEC_HLV^{1\to 2+2}$のアルゴリズムについては図５に示すフローチャート、
$ExHEC_HLV^{2\to 2+2}$のアルゴリズムについては図６に示すフローチャート、
それぞれにおいて説明した処理と同様の処理となる。
【０２２６】
また、$ExHEC\_HLV^{2\to 2+2}$、の計算手順についても、先に説明した（処理例１：提案法Ａ１）の項目において説明したアルゴリズム６［Ａｌｇｏｒｉｔｈｍ６］と同様の処理であり、$ExHEC_HLV^{1\to 2+2}$の計算手順についても、先に説明した（処理例１：提案法Ａ１）の項目において説明したアルゴリズム７［Ａｌｇｏｒｉｔｈｍ７］と同様の処理となる。
【０２２７】
［処理例８：提案法Ｆ２］
処理例８（提案法Ｆ２）は、前述した処理例（処理例２：提案法Ｆ１）のさらなる高速化手法であり、種数２，標数２の$h(x)=x^2+h_1x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において因子Ｄの１／２倍算を計算する方法である。
【０２２８】
前述の処理例（処理例７：提案法Ａ２）において適用したＡｌｇｏｒｉｔｈｍ５ａを良く見ると、ｈ（ｘ）の係数との乗算、ｈ（ｘ）の係数の逆元演算が多くある。つまり、ｈ（ｘ）の係数を工夫することにより、乗算および逆元演算の計算量を削減することができる。なお、文献（非特許文献１９：T. Lange. Efficient Doubling on Genus Two Curves over Binary Fields, SAC 2004, pre-proceedings, pp.189-202, 2004.）では、高速化のため、$h_2=1$、$f_4=0$を用いている。このパラメータを用いた場合のＨａｒｌｅｙＤＢＬの計算量は、２１Ｍ＋５Ｓ＋１Ｉである。
【０２２９】

ここで説明する処理例８（提案法Ｆ２）もこれに条件を合わせることにするが、補題１より、ｈ（ｘ）は既約多項式を仮定しているため、

$h(x)=x^2+h_1x+h_0$、

$Tr(h_0/h_1^2)=1$[0]

とする（二次方程式$ax^2+bx+c=0$が既約多項式であるための必要十分条件は$Tr(ac/b^2)=1$である）。この場合の計算法をアルゴリズム８ａ［Ａｌｇｏｒｉｔｈｍ　８ａ］ＨＥＣ＿ＨＬＶ（$h_2=1$、$f_4=0$）に示す。

【０２３０】

【数２８】

Algorithm 8 $aHEC\_HLV(h_2=1, f_4=0)$

$Input: D_2=(U_2,V_2), 1/h_1^2$

$Output: D_1=(U_1,V_1)=[1/2]D_2$

$U_i(x)=x^2+u_{i1}x+u_{i0}, V_i(x)=v_{i1}x+v_{i0}, \gcd(h,U_i)=1, i=1,2$

1. Solve $k_1+k_1^2u_{21}+1=0$

   $\alpha \leftarrow 1/u_{21},\ k_1 \leftarrow \mathrm{H}(u_{21})\alpha,\ k_1' \leftarrow k_1+\alpha$

2. Select correct k_1 by solving $k_1h_0+k_0h_1+k_0^2u_{21}+c_1=0$

   $c_1 \leftarrow f_3+v_{21}+u_{20}+u_{21}^2$

   $c_0 \leftarrow f_2+v_{20}+v_{21}(h_1+v_{21})+u_{21}(u_{20}+c_1)$

   $w_0 \leftarrow u_{21}/h_1^2,\ \alpha \leftarrow h_1\alpha,\ \gamma \leftarrow (c_1+k_1h_0)w_0$

   if $\mathrm{Tr}(\gamma)=1$ then $k_1 \leftarrow k_1',\ \gamma \leftarrow (c_1+k_1h_0)w_0$

   $k_0 \leftarrow \mathrm{H}(\gamma)\alpha,\ k_0' \leftarrow k_0+\alpha$

3. Select correct k_0 by checking trace of $x+x^2u_{11}+1=0$

   $w_0 \leftarrow k_1^2,\ w_1 \leftarrow w_0u_{20}+k_1h_1+u_{21}$

   $w_2 \leftarrow k_0+\sqrt{w_1+k_0},\ \ w_4 \leftarrow k_1u_{21}+1,\ u_{11} \leftarrow w_2w_4$

   if $\mathrm{Tr}(u_{11})=1$ then

   $k_0 \leftarrow k_0',\ w_2 \leftarrow k_0+\sqrt{w_1+k_0},\ u_{11} \leftarrow w_2w_4$

4. Compute U_1

   $w_1 \leftarrow k_0u_{20},\ w_5 \leftarrow w_4+1,\ w_6 \leftarrow (k_0+k_1)(u_{20}+u_{21})$

   $u_{10} \leftarrow w_4\sqrt{k_0(w_1+h_0)+c_0}$

5. Compute $V_1=V_2+h+kU_2 \bmod U_1$

   $w_4 \leftarrow w_5+k_0+1,\ w_5 \leftarrow w_1+w_5+w_6+v_{21}+h_1$

   $w_6 \leftarrow w_1+v_{20}+h_0,\ w_7 \leftarrow w_2+w_4$

   $w_1 \leftarrow w_7u_{10},\ w_3 \leftarrow (k_1+w_7)(u_{10}+u_{11})$

   $v_{11} \leftarrow w_1+w_2+w_3+w_5$

   $v_{10} \leftarrow w_1+w_6$

6. $U_1(x) \leftarrow x^2+u_{11}x+u_{10},\ V_1(x) \leftarrow v_{11}x+v_{10}$

7. return $D_1=(U_1,V_1)$

【０２３１】

ここで、逆元演算の回数を削減する方法を考える。前述したアルゴリズム５ａ［Ａｌｇｏｒｉｔｈｍ　５ａ］では、１／ｕ$_{21}$，１／ｈ$_2{}^2$，１／ｈ$_1{}^2$，１／ｋ$_1$の４回の逆元演算が必要であるが、ここで、

ｈ$_2$＝１

と設定することにより、逆元演算は３回とすることができる。さらに、ｈ$_1$は曲線パラ

メータであるため、事前に$1/h_1^2$を計算しておき、入力として与えることにより、逆元演算は$1/u_{21}$と$1/k_1$の２回に削減することができる。また、$1/k_1$に関しては，前述の［処理例７：提案法Ａ２］において説明した式（４），すなわち、

$1/k_1 = h_2 + k_1 u_{21}$

より，乗算１回と加算１回で求めることができる。これらの演算により、上記のアルゴリズム８ａ［Ａｌｇｏｒｉｔｈｍ　８ａ］ＨＥＣ＿ＨＬＶ（$h_2=1$、$f_4=0$）で必要となる逆元演算は$1/u_{21}$の１回のみとなる。

【０２３２】

結果として、上記のアルゴリズム８ａ［Ａｌｇｏｒｉｔｈｍ　８ａ］ＨＥＣ＿ＨＬＶ（$h_2=1$、$f_4=0$）における計算量は次の通りとなる。

（ａ）k_1，k_0が正しい値である場合：１８Ｍ＋２Ｓ＋１Ｉ＋２ＳＲ＋２Ｈ＋２Ｔ

（ｂ）k_1，k_0'が正しい値である場合：１９Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

（ｃ）k_1'，k_0が正しい値である場合：２０Ｍ＋２Ｓ＋１Ｉ＋２ＳＲ＋２Ｈ＋２Ｔ

（ｄ）k_1'，k_0'が正しい値である場合：２１Ｍ＋２Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

上記（ａ）～（ｄ）のすべてについて平均すると、１９．５Ｍ＋２Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔとなる。

【０２３３】

ＨａｒｌｅｙＤＢＬの計算量は、２１Ｍ＋５Ｓ＋１Ｉであった。ここで、文献［（非特許文献１５）　E.Knudsen. Elliptic Scalar Multiplication Using Point Halving. ASIACRYPTO '99, LNCS 1716, pp.135-149, Springer-Verlag, 1999.］によると、有限体が正規基底で定義されている場合、Ｓ（２乗算），ＳＲ（平方根演算），Ｈ（ハーフトレース（二次方程式の根を求める演算）），Ｔ（トレース（二次方程式の根があるか否かの判定））の計算量は無視することができ、Ｍ（乗算），Ｉ（逆元演算）の計算量のみ考慮すればよいことが知られている。そのため、正規基底を用いた場合、Ａｌｇｏｒｉｔｈｍ８ａはＨａｒｌｅｙＤＢＬよりも１．５Ｍ高速となる。

【０２３４】

また、有限体が多項式基底で定義されている場合、

文献［（非特許文献１６）　K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/ corr2003-18.pdf１８］によると、通常ＳＲ（平方根演算），Ｈ（ハーフトレース（二次方程式の根を求める演算））の計算量は、Ｍ（乗算）との対比において、ＳＲ＝Ｈ＝０．５Ｍ程度であることが知られている。また、Ｔ（トレース（二次方程式の根があるか否かの判定））の計算量は無視できる。さらにＳ（２乗算）の計算量は、Ｍ（乗算）の数分の１程度であることが知られている。しかしながら、ＳＲの計算量は、多項式基底の選び方によっては、０．５Ｍよりも少なくなることも、同時に知られている。なお、例外ケースは、前述の処理例７（提案法Ａ２）の例外ケースに基いて計算できる。

【０２３５】

［処理例９（提案法Ｂ２）］

処理例９（提案法Ｂ２）は、前述した処理例（処理例３：提案法Ｂ１）のさらなる高速化手法であり、種数２、標数２の$h(x)=x^2+x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する方法である。

【０２３６】

前述の処理例８（提案法Ｆ２）において説明したように、処理例７（提案法Ａ２）において説明した１／２倍算の計算アルゴリズム、すなわち、アルゴリズム５ａ［Ａｌｇｏｒｉｔｈｍ　５ａ　ＨＥＣ＿ＨＬＶ］を良く見ると、$h(x)$の係数との乗算、$h(x)$の係数の逆元演算が多くある。つまり、$h(x)$の係数を工夫することにより、乗算および逆元演算の計算量を削減することができる。文献［J.Pelzl, T.Wollinger, and C.Paar. High Performance Arithmetic for Hyperelliptic Curve Cryptosystems of Genus Two. Cryptology ePrint Archive, 2003/212, IACR, 2003］では、高速化のために、h_2，h

$_1 \in \{0, 1\}$，$f_4 = 0$を用いた例を示している。

【０２３７】

このパラメータを用いた場合の［ＨａｒｌｅｙＤＢＬ］の計算量は、

１８Ｍ＋７Ｓ＋１Ｉ

である。

【０２３８】

処理例９（提案法Ｂ２）もこれに条件を合わせることにするが、前述した［補題１］より、ｈ（ｘ）は既約多項式を仮定しているため、

$h(x) = x^2 + x + h_0$，

$Tr(h_0) = 1$

とする（二次方程式$ax^2 + bx + c = 0$が既約多項式であるための必要十分条件はＴｒ（ａｃ／ｂ^2）＝１である）。

【０２３９】

この場合の計算法示すアルゴリズムをアルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ１０ａ ＨＥＣ＿ＨＬＶ（$h_2 = h_1 = 1$，$f_4 = 0$）］として以下に示す。

【数２９】

$$\text{Algorithm 10}\ aHEC\_HLV(h_2=h_1=1, f_4=0)$$

$$Input: D_2=(U_2,V_2)$$

$$Output: D_1=(U_1,V_1)=[1/2]D_2$$

$$U_i(x)=x^2+u_{i1}x+u_{i0},\ V_i(x)=v_{i1}x+v_{i0},\ \gcd(h,U_i)=1, i=1,2$$

1. Solve $k_1+k_1^2u_{21}+1=0$

   $\alpha \leftarrow 1/u_{21},\ k_1 \leftarrow \mathrm{H}(u_{21})\alpha,\ k_1' \leftarrow k_1+\alpha$

2. Select correct k_1 by solving $k_1h_0+k_0+k_0^2u_{21}+c_1=0$

   $c_1 \leftarrow f_3+v_{21}+u_{20}+u_{21}^2$

   $c_0 \leftarrow f_2+v_{20}+v_{21}+v_{21}^2+u_{21}(u_{20}+c_1)$

   $\gamma \leftarrow (c_1+k_1h_0)u_{21}$

   if $\mathrm{Tr}(\gamma)=1$ then $k_1 \leftarrow k_1',\ \gamma \leftarrow \gamma+h_0$

   $k_0 \leftarrow \mathrm{H}(\gamma)\alpha,\ k_0' \leftarrow k_0+\alpha$

3. Select correct k_0 by checking trace of $x+x^2u_{11}+1=0$

   $w_0 \leftarrow k_1^2,\ w_1 \leftarrow w_0u_{20}+k_1+u_{21}$

   $w_2 \leftarrow k_0+\sqrt{w_1+k_0},\ w_4 \leftarrow k_1u_{21}+1,\ u_{11} \leftarrow w_2w_4$

   if $\mathrm{Tr}(u_{11})=1$ then

   $k_0 \leftarrow k_0',\ w_2 \leftarrow k_0+\sqrt{w_1+k_0},\ u_{11} \leftarrow w_2w_4$

4. *Compute* U_1

   $w_1 \leftarrow k_0u_{20},\ w_5 \leftarrow w_4+1,\ w_6 \leftarrow (k_0+k_1)(u_{20}+u_{21})$

   $u_{10} \leftarrow w_4\sqrt{k_0(w_1+h_0)+c_0}$

5. *Compute* $V_1=V_2+h+kU_2 \bmod U_1$

   $w_4 \leftarrow w_5+k_0+1,\ w_5 \leftarrow w_1+w_5+w_6+v_{21}+1$

   $w_6 \leftarrow w_1+v_{20}+h_0,\ w_7 \leftarrow w_2+w_4$

   $w_1 \leftarrow w_7u_{10},\ w_3 \leftarrow (k_0+w_7)(u_{10}+u_{11})$

   $v_{11} \leftarrow w_1+w_2+w_3+w_5$

   $v_{10} \leftarrow w_1+w_6$

6. $U_1(x) \leftarrow x^2+u_{11}x+u_{10},\ V_1(x) \leftarrow v_{11}x+v_{10}$

7. *return* $D_1=(U_1,V_1)$

【０２４０】

上記のアルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ１０ａ　ＨＥＣ＿ＨＬＶ（ｈ$_2$＝ｈ$_1$＝１，ｆ$_4$＝０）］における計算量は次の通りである。

（ａ）ｋ$_1$，ｋ$_0$が正しい値である場合：１４Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈ＋２Ｔ

（ｂ）ｋ$_1$，ｋ$_0$'が正しい値である場合：１５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

（ｃ）ｋ$_1$'，ｋ$_0$が正しい値である場合：１４Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈ＋２Ｔ

（ｄ）ｋ$_1$'，ｋ$_0$'が正しい値である場合：１５Ｍ＋３Ｓ＋１Ｉ＋３ＳＲ＋２Ｈ＋２Ｔ

【０２４１】

上記（ａ）～（ｄ）のすべての場合について平均すると、

１４．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ

となる。ＨａｒｌｅｙＤＢＬの計算量は、１８Ｍ＋７Ｓ＋１Ｉであった。ここで、前述したように、文献［E.Knudsen. Elliptic Scalar Multiplication Using Point Halving. ASIACRYPTO '99, LNCS 1716, pp.135-149, Springer-Verlag, 1999.］によると、有限体が正規基底で定義されている場合、Ｓ（２乗算），ＳＲ（平方根演算），Ｈ（ハーフトレース（二次方程式の根を求める演算）），Ｔ（トレース（二次方程式の根があるか否かの判定））の計算量は無視することができ、Ｍ（乗算），Ｉ（逆元演算）の計算量のみ考慮すればよいことが知られている。

【０２４２】

そのため、正規基底を用いた場合、上述したアルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ１０ａ］は、従来のアルゴリズム［ＨａｒｌｅｙＤＢＬ］よりも、３．５Ｍ高速となる。また、有限体が多項式基底で定義されている場合、

文献［K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18.pdf］によると、通常ＳＲ，Ｈの計算量は、ＳＲ＝Ｈ＝０．５Ｍ程度であることが知られている。また、Ｔの計算量は無視できる。さらにＳの計算量は、Ｍの数分の１程度であることが知られている。しかしながら、ＳＲの計算量は、多項式基底の選び方によっては、０．５Ｍよりも少なくなることも同時に知られている。

【０２４３】

さて、前述のアルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ１０ａ］の曲線にさらに、h_0＝１の制約を設ける。前述のアルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ１０ａ］は、h_0の乗算が１回あるので、h_0＝１とすることにより、１Ｍの削減ができ、上記（ａ）～（ｄ）のすべての場合についての平均の計算量は、

１３．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ

となる。一方、ＨａｒｌｅｙＤＢＬは、計算量は、

１５Ｍ＋７Ｓ＋１Ｉ

である。有限体が正規規定で定義されている場合、Ｓ，ＳＲ，Ｈ，Ｔの計算量は無視することができ、正規規定を用いた場合、アルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ１０ａ］は、従来のアルゴリズム［ＨａｒｌｅｙＤＢＬ］よりも、１．５Ｍ高速となる。なお、例外ケースは、前述の処理例７（提案法Ａ２）の例外ケースに基いて計算できる。

【０２４４】

［処理例１０（提案法Ｃ２）］

処理例１０（提案法Ｃ２）は、前述した処理例（処理例５：提案法Ｃ１）のさらなる高速化手法であり、種数２、標数２のランダムパラメータを持つような超楕円曲線、$h(x)=x^2+h_1x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線、および$h(x)=x^2+x+h_0$、$f_4=0$をパラメータに持つような超楕円曲線において、因子Ｄの１／２倍算を計算する際に、１／２倍したものの候補が２つ出てくる。２つの候補のうち、正しい値を選ぶ必要があるが、選ぶ際に、有限体のトレース、乗算、平方根を計算する必要がある。どちらが正しいかは、因子Ｄに依存する。そのため、因子Ｄが固定されている場合、事前に２つの候補のうち、どちらが正しいかをテーブルに保持し、正しい値を選ぶ際に、このテーブルを参照することにより、上記の余計な計算を省略する方法である。

【０２４５】

k_1，k_1'（k_0，k_0'）のどちらが正しいかは、入力因子Ｄに依存している。したがって、Ｄが固定されている場合、例えばＥＣＤＨ型鍵交換のフェーズ１や、ＥＣＤＳＡ型署名生成、検証など、ベースポイントがあらかじめ決まっている場合、事前に、［１／2^i］Ｄを計算し、k_1，k_1'（k_0，k_0'）のどちらが正しい値であるかを、テーブルに記録しておく。

【０２４６】

例えば、ベースポイントの位数と同じビットサイズのテーブルをT_1，T_0と二つ用意し、これらを２進数表現したものを、

$T_1 = (t_{1,r-1}, \cdots, t_{1,0})$
$T_0 = (t_{0,r-1}, \cdots, t_{0,0})$
とする。

【０２４７】

［１／２^i］Ｄを求める際に、
k_1が正しければ、$t_{1,i}=0$、そうではなくて、k_1'が正しければ$t_{1,i}=1$
k_1が正しければ$t_{0,i}=0$、そうではなくて、k_0'が正しければ$t_{0,i}=1$などとテーブルに記録すれば、テーブルサイズは、わずかベースポイントの位数のサイズの２倍程度のビット列で済む。このテーブルを参照することにより、１／２倍算の計算量を削減することができる。

【０２４８】

この方法を前述のアルゴリズム８ａ［Ａｌｇｏｒｉｔｈｍ　８ａ］　ＨＥＣ＿ＨＬＶ（$h_2=1, f_4=0$）に適用したものを、アルゴリズム９ａ［Ａｌｇｏｒｉｔｈｍ　９ａ］　ＨＥＣ＿ＨＬＶ（$h_2=1, f_4=0$，ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）として示す。計算量は、１８Ｍ＋２Ｓ＋１Ｉ＋２ＳＲ＋２Ｈとなる。

【０２４９】

【数３０】

Algorithm 9$aHEC_HLV(h_2=1, f_4=0,$ table-lookup)

Input : $D_2=(U_2,V_2)$, $1/h_1^2$, t_0, t_1

Output : $D_1=(U_1,V_1)=[1/2]D_2$

$U_i(x)=x^2+u_{i1}x+u_{i0}$, $V_i(x)=v_{i1}x+v_{i0}$, $\gcd(h,U_i)=1, i=1,2$

1. Solve $k_1+k_1^2u_{21}+1=0$

   $\alpha \leftarrow 1/u_{21}$

   if $t_1=0$ then $k_1 \leftarrow \mathrm{H}(u_{21})\alpha$ else $k_1 \leftarrow (\mathrm{H}(u_{21})+1)\alpha$

2. Solve $k_1h_0+k_0h_1+k_0^2u_{21}+c_1=0$

   $c_1 \leftarrow f_3+v_{21}+u_{20}+u_{21}^2$

   $c_0 \leftarrow f_2+v_{20}+v_{21}(h_1+v_{21})+u_{21}(u_{20}+c_1)$

   $w_0 \leftarrow u_{21}/h_1^2$, $\alpha \leftarrow h_1\alpha$, $\gamma \leftarrow (c_1+k_1h_0)w_0$

   if $t_0=0$ then $k_0 \leftarrow \mathrm{H}(\gamma)\alpha$ else $k_0 \leftarrow (\mathrm{H}(\gamma)+1)\alpha$

3. Compute U_1

   $w_0 \leftarrow k_1^2$, $w_1 \leftarrow w_0u_{20}+k_1h_1+u_{21}$, $w_2 \leftarrow k_0+\sqrt{w_1+k_0}$

   $w_4 \leftarrow k_1u_{21}+1$, $u_{11} \leftarrow w_2w_4$

   $w_1 \leftarrow k_0u_{20}$, $w_5 \leftarrow w_4+1$, $w_6 \leftarrow (k_0+k_1)(u_{20}+u_{21})$

   $u_{10} \leftarrow w_4\sqrt{k_0(w_1+h_0)+c_0}$

4. Compute $V_1=V_2+h+kU_2 \bmod U_1$

   $w_4 \leftarrow w_5+k_0+1$, $w_5 \leftarrow w_1+w_5+w_6+v_{21}+h_1$

   $w_6 \leftarrow w_1+v_{20}+h_0$, $w_7 \leftarrow w_2+w_4$

   $w_1 \leftarrow w_7u_{10}$, $w_3 \leftarrow (k_1+w_7)(u_{10}+u_{11})$

   $v_{11} \leftarrow w_1+w_2+w_3+w_5$

   $v_{10} \leftarrow w_1+w_6$

6. $U_1(x) \leftarrow x^2+u_{11}x+u_{10}$, $V_1(x) \leftarrow v_{11}x+v_{10}$

7. return $D_1=(U_1,V_1)$

【０２５０】

具体的に，この方法を上述のアルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ１０ａ　ＨＥＣ＿ＨＬＶ　（ｈ$_2$＝ｈ$_1$＝１，ｆ$_4$＝０）］に適用したアルゴリズムをアルゴリズム１１ａ［Ａｌｇｏｒｉｔｈｍ１１ａ　ＨＥＣ＿ＨＬＶ（ｈ$_2$＝ｈ$_1$＝１，ｆ$_4$＝０，ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）］として、以下に示す。

【０２５１】

【数３１】

Algorithm 11a $HEC\_HLV(h_2 = h_1 = 1, f_4 = 0,$ with table-lookup)

Input : $D_2 = (U_2, V_2), t_0, t_1$

Output : $D_1 = (U_1, V_1) = [1/2]D_2$

$U_i(x) = x^2 + u_{i1}x + u_{i0}$, $V_i(x) = v_{i1}x + v_{i0}$, $\gcd(h, U_1) = 1$

1. Solve $k_1 + k_1^2 u_{21} + 1 = 0$ and select correct k_1 via t_1

   $\alpha \leftarrow 1/u_{21}$

   if $t_1 = 0$ then $k_1 \leftarrow \mathrm{H}(u_{21})\alpha$ else $k_1 \leftarrow (\mathrm{H}(u_{21}) + 1)\alpha$

2. Solve $k_1 h_0 + k_0 + k_0^2 u_{21} + c_1 = 0$ and select correct k_0 via t_0

   $c_1 \leftarrow f_3 + v_{21} + u_{20} + u_{21}^2$

   $c_0 \leftarrow f_2 + v_{20} + v_{21} + v_{21}^2 + u_{21}(u_{20} + c_1)$

   $\gamma \leftarrow (c_1 + k_1 h_0)u_{21}$

   $k_0 \leftarrow \mathrm{H}(\gamma)\alpha$

   if $t_0 = 0$ then $k_0 \leftarrow \mathrm{H}(\gamma)\alpha$ else $k_0 \leftarrow (\mathrm{H}(\gamma) + 1)\alpha$

3. Compute U_1

   $w_0 \leftarrow k_1^2$, $w_1 \leftarrow w_0 u_{20} + k_1 + u_{21}$

   $w_2 \leftarrow k_0 + \sqrt{w_1 + k_0}$

   $w_4 \leftarrow k_1 u_{21} + 1$, $u_{11} \leftarrow w_2 w_4$

   $w_1 \leftarrow k_0 u_{20}$, $w_5 \leftarrow w_4 + 1$, $w_6 \leftarrow (k_0 + k_1)(u_{20} + u_{21})$

   $u_{10} \leftarrow w_4 \sqrt{k_0(w_1 + h_0) + c_0}$

5. *Compute* $V_1 = V_2 + h + kU_2 \bmod U_1$

   $w_4 \leftarrow w_5 + k_0 + 1$, $w_5 \leftarrow w_1 + w_5 + w_6 + v_{21} + 1$

   $w_6 \leftarrow w_1 + v_{20} + h_0$, $w_7 \leftarrow w_2 + w_4$

   $w_1 \leftarrow w_7 u_{10}$, $w_3 \leftarrow (k_0 + w_7)(u_{10} + u_{11})$

   $v_{11} \leftarrow w_1 + w_2 + w_3 + w_5$

   $v_{10} \leftarrow w_1 + w_6$

6. $U_1(x) \leftarrow x^2 + u_{11}x + u_{10}$, $V_1(x) \leftarrow v_{11}x + v_{10}$

7. *return* $D_1 = (U_1, V_1)$

【０２５２】

上記アルゴリズム１１ａ［Ａｌｇｏｒｉｔｈｍ１１ａ　ＨＥＣ＿ＨＬＶ（ｈ$_2$＝ｈ$_1$＝１，ｆ$_4$＝０，ｗｉｔｈ　ｔａｂｌｅ－ｌｏｏｋｕｐ）］における計算量は、

１４Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈとなり、さらにｈ$_0$＝１とすることにより、１Ｍの削減ができ、この場合の計算量は、

１３Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈとなる。

【０２５３】

［処理例１１（提案法Ｄ２）］

処理例１１（提案法Ｄ２）は、上記処理例７～１０に示す因子の１／２倍算の計算方法を用いて、因子のスカラー倍算を計算する方法である。

【０２５４】
楕円曲線での有理点の１／２倍算を用いたスカラー倍算の計算方法は、文献［E.Knudsen. Elliptic Scalar Multiplication Using Point Halving. ASIACRYPTO '99, LNCS 1716, pp.135-149, Springer-Verlag, 1999.］、［K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18,http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18 .pdf］にある。超楕円曲線の因子の１／２倍算を用いたスカラー倍算の計算方法は、この文献にあるスカラー倍算に基づいて実行される。ここで、スカラー倍の対象となる因子Ｄは、位数が大きな素数ｒであるとする。また，スカラー値を整数０＜ｄ＜ｒとする。１／２倍算を用いたスカラー倍算を行なうには、まず、２進数で表現されているスカラー値ｄを１／２進数で表現しなおす必要がある。
【０２５５】
ここで、
【数３２】

$$m = \lfloor \log_2 r \rfloor$$

とする。また、ｄを２ｍ倍しｒで割った余り、すなわち、
【数３３】

$$\sum_{i=0}^{m} \hat{d}_i 2^i \leftarrow 2^m d \pmod{r}$$

を求める。次にこれを２ｍで割ると、
【数３４】

$$\sum_{i=0}^{m} d_i / 2^i \leftarrow \sum_{i=0}^{m} \hat{d}_i 2^{i-m}$$

となる。上記式の、

【数３５】

$$\sum_{i=0}^{m} d_i / 2^i$$

で表現されたスカラー値を、１／２倍算を用いたスカラー倍算［ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ］法で用いる。ここで、

【数３６】

$$d_i, \hat{d}_i \in \{0,1\}$$

である。

【０２５６】

以下に、アルゴリズム１２ａ［Ａｌｇｏｒｉｔｈｍ　１２ａ］として、ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ法（ｒｉｇｈｔ－ｔｏ－ｌｅｆｔ）、アルゴリズム１３ａ［Ａｌｇｏｒｉｔｈｍ　１３ａ］として、ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ法（ｌｅｆｔ－ｔｏ－ｒｉｇｈｔ）をそれぞれ示す。

【数３７】

Algorithm 12a *Halve-and-add binary Method (right-to-left)*

Input: $D \in \mathbf{J}(\mathrm{F}_{2^n})$ *such that* $2D \neq O$, $d \in \mathrm{Z}$, r: order of D, $m = \lfloor \log_2 r \rfloor$

Output: *scalar multiplication* dD

1. $\sum_{i=0}^{m} \hat{d}_i 2^i \leftarrow 2^m d \mod r$, $\hat{d}_i \in \{0,1\}$
2. $\sum_{i=0}^{m} d_i / 2^i \leftarrow \sum_{i=0}^{m} \hat{d}_i 2^{i-m}$, $d_i \in \{0,1\}$
3. $Q \leftarrow O$, $R \leftarrow D$
4. for i from 0 to m do:
   if $d_i = 1$ then $Q \leftarrow Q + R$
   $R \leftarrow HEC\_HLV(R)$
5. return Q

【０２５７】
【数３８】

Algorithm 13a *Halve-and-add binary Method (left-to-right)*

Input : $D \in \mathbf{J}(\mathrm{F}_{2^n})$ *such that* $2D \neq O$, $d \in \mathrm{Z}$, r : order of D, $m = \lfloor \log_2 r \rfloor$

Output : *scalar multiplication* dD

1. $\sum_{i=0}^{m} \hat{d}_i 2^i \leftarrow 2^m d \mod r$, $\hat{d}_i \in \{0,1\}$
2. $\sum_{i=0}^{m} d_i / 2^i \leftarrow \sum_{i=0}^{m} \hat{d}_i 2^{i-m}$, $d_i \in \{0,1\}$
3. $Q \leftarrow O$
4. for i from m downto 0 do:
   $Q \leftarrow HEC\_HLV(Q)$
   if $d_i = 1$ then $Q \leftarrow Q + D$
5. return Q

【０２５８】
上記アルゴリズム１２ａ［Ａｌｇｏｒｉｔｈｍ　１２ａ］と、アルゴリズム１３ａ［Ａｌｇｏｒｉｔｈｍ　１３ａ］のステップ４に出てくるＨＥＣ＿ＨＬＶは、ランダムカーブを用いた前述のアルゴリズム５ａ［Ａｌｇｏｒｉｔｈｍ　５ａ］のＨＥＣ＿ＨＬＶでも良いし、アルゴリズム８ａ［Ａｌｇｏｒｉｔｈｍ　８ａ］の曲線パラメータに、$h_2 = 1$，$f_4 = 0$の制約を設けたＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ　１０ａ］の曲線パラメータに$h_2 = h_1 = 1$，$f_4 = 0$の制約を設けたＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ　１０ａ］の曲線パラメータに$h_2 = h_1 = h_0 = 1$，$f_4 = 0$の制約を設けたＨＥＣ＿ＨＬＶでも良い。
【０２５９】
前段の処理例１～６において説明した１／２倍算において、例えば前述の［処理例３（提案法Ｂ１）］に示すアルゴリズム１０や、［処理例５（提案法Ｃ１）］に示すアルゴリズム１１では、入力に$1/u_{21}$が、出力に$1/u_{11}$が必要であった。これにより、ベースポイントＤの$1/2^i$倍算、すなわち、
（１／２）Ｄ、（１／２^2）Ｄ、（１／２^3）Ｄ、（１／２^i）Ｄ・・・
を、前の１／２倍算の出力の$1/u_{11}$を次の１／２倍算の入力として与えることができるので、効率よく計算することができる。そのため、スカラー倍を　ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ法（ｒｉｇｈｔ－ｔｏ－ｌｅｆｔ）で行なう場合には、ベースポイントＤの$1/2^i$倍を適宜加えていけばよいので、ｒｉｇｈｔ－ｔｏ－ｌｅｆｔの場合には、効率よくスカラー倍算を行なうことができた。
【０２６０】
一方、ｌｅｆｔ－ｔｏ－ｒｉｇｈｔの場合には、前の１／２倍算の出力を次の１／２倍算の入力とすることができない場合がある。前述のアルゴリズム１３ａ［Ａｌｇｏｒｉｔｈｍ１３ａ］のＳｔｅｐ４では、まず途中の結果Ｑを１／２倍し（Ｑ←ＨＥＣ＿ＨＬＶ（Ｑ））、スカラー値のあるビットが１の場合は、ベースポイントを途中結果に加える（ｉｆ　ｄｉ＝１ｔｈｅｎＱ←Ｑ＋Ｄ）。このため、ビットが１の場合の、次のビットでは、前の１／２倍算の出力の$1/u_{11}$を次の１／２倍算の入力として与えることができない

ので、新たに入力値１／u_{21}を計算する必要がある。有限体の逆元演算は乗算に比べて非常に計算量がかかる。そのため、先の処理例１～６において説明した１／２倍算法では、ｌｅｆｔ－ｔｏ－ｒｉｇｈｔを適用した場合、１／２倍算の入力値を生成するための余計な逆元演算を必要とするため、計算の効率が悪くなる。しかしながら、処理例７以降において説明した提案法では、入力に１／u_{21}を必要としない設定としているため、ｌｅｆｔ－ｔｏ－ｒｉｇｈｔでもｒｉｇｈｔ－ｔｏ－ｌｅｆｔと同等の計算量で計算することができる。

【０２６１】

また、前述のアルゴリズム１２ａ［Ａｌｇｏｒｉｔｈｍ　１２ａ］のｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ法（ｒｉｇｈｔ－ｔｏ－ｌｅｆｔ）のステップ４のＨＥＣ＿ＨＬＶは、ランダムカーブを用いたアルゴリズム５ａ［Ａｌｇｏｒｉｔｈｍ　５ａ］にテーブル参照法を適用したＨＥＣ＿ＨＬＶでも良いし、アルゴリズム８ａ［Ａｌｇｏｒｉｔｈｍ　８ａ］の曲線パラメータに$h_2=1$，$f_4=0$の制約を設け、テーブル参照法を適用したＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ　１０ａ］の曲線パラメータに$h_2=h_1=1$，$f_4=0$の制約を設け、テーブル参照法を適用したＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ　１０ａ］の曲線パラメータに$h_2=h_1=h_0=1$，$f_4=0$の制約を設け，テーブル参照法を適用したＨＥＣ＿ＨＬＶでも良い。

【０２６２】

また，ｂｉｎａｒｙ法のほかに、ｗｉｎｄｏｗ法も適用することができる。入力因子をＤとしｗｉｎｄｏｗ幅をｗとする。途中結果を代入する因子をＱ←Ｏとする。

整数ｉ＝$(i_{w-1}, i_{w-2}\cdots i_0)_2 \in \{0, 1, \cdots, 2^w-1\}$

に対して事前計算、すなわち、

【数３９】

$$D_i \leftarrow \sum_{j=0}^{w-1} \frac{i_j}{2^j} D$$

を行い2^w個の因子からなるテーブルを計算しておく。

【０２６３】

また、スカラー値ｄを、１／2^w進数展開する。

【数４０】

$$\sum_{i=0}^{l} c_i / (2^w)^i \leftarrow \sum_{i=0}^{m} d_i / 2^i$$

はじめにＱに１／２倍算をｗ回適用し、

Ｑ←（１／2^w）Ｑ

とする。次にｄの上位ビットからｗｉｎｄｏｗ幅ｗで、スカラー値c_1を走査し、対応

するテーブルの因子の値を参照し、結果に足し込む。
Q←Q＋D$_{c\ 1}$
これをc$_0$まで繰り返す。
【０２６４】
この計算手法［ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｗｉｎｄｏｗ　ｍｅｔｈｏｄ］を、アルゴリズム１４ａ［Ａｌｇｏｒｉｔｈｍ　１４ａ］として以下に示す。
【数４１】

Algorithm 14a *Halve – and – add Window Method*

Input : $D \in \mathbf{J}(\mathrm{F}_{2^n})$ *such that* $2D \neq O$, $d \in \mathrm{Z}$, r : order of D, $m = \lfloor \log_2 r \rfloor$

Output : *scalar multiplication* dD

1. $\sum_{i=0}^{m} \hat{d}_i 2^i \leftarrow 2^m d \mod r,\ \hat{d}_i \in \{0,1\}$
2. $\sum_{i=0}^{m} d_i / 2^i \leftarrow \sum_{i=0}^{m} \hat{d}_i 2^{i-m},\ d_i \in \{0,1\}$
3. $\sum_{i=0}^{l} c_i /(2^w)^i \leftarrow \sum_{i=0}^{m} d_i / 2^i,\ c_i \in \{0,1,\cdots,2^w-1\}$
4. $D_i \leftarrow \frac{i_{w-1}}{2^{w-1}} D + \cdots + \frac{i_0}{2^0} D$, for $i = (i_{w-1}\cdots i_0)_2 \in \{0,1,\cdots,2^w-1\}$
5. $Q \leftarrow O$
6. for i from l downto 0 do:
   for j from 1 to w do: $Q \leftarrow HEC\_HLV(Q)$
   $Q \leftarrow Q + D_{c_i}$
7. return Q

【０２６５】
また、
因子Ｄ＝（Ｕ，Ｖ），
Ｕ＝ｘ^{2}＋ｕ$_{1}$ｘ＋ｕ$_{0}$，
Ｖ＝ｖ$_{1}$ｘ＋ｖ$_{0}$，
の逆元は、
－Ｄ＝（Ｕ，Ｖ＋ｈｍｏｄＵ）＝（Ｕ，（ｖ$_{1}$＋ｈ$_{2}$ｕ$_{1}$＋ｈ$_{1}$）ｘ＋（ｖ$_{0}$＋ｈ$_{2}$ｕ$_{0}$＋ｈ$_{0}$））
で表すことが出来る。特にｈ$_{2}$＝１の場合は、有限体の乗算は必要なく、４回の有限体の加算でＤから－Ｄを求めることが出来る。因子Ｄの減算は因子－Ｄの加算により計算できる。つまり、因子の加算と減算は同等の計算量で求めることができる。
【０２６６】
そこで、スカラー値を負の値も使って表現し、それを用いてスカラー倍算を行なうことが出来る。まず、非隣接形式ＮＡＦ（Ｎｏｎ－Ａｄｊａｃｅｎｔ　Ｆｏｒｍ）を使い、ある整数ｓを｛－１，０，１｝で表現する。ＮＡＦでは２進数表現されたスカラー値ｓを下位のビットから見ていく。１が隣接するところがあれば、例えば、

（１１）であれば、（１０－１）、つまり３＝２^{2}－１，
（１１１）であれば、（１００－１）、つまり７＝２^{3}－１
などと表現する。

【０２６７】

ＮＡＦの計算手法を、アルゴリズム１５ａ［Ａｌｇｏｒｉｔｈｍ　１５ａ］として以下に示す。

【数４２】

Algorithm 15a *Conversion to NAF*

Input : $s=\sum_{j=0}^{l-1} s_j 2^j \in \mathrm{Z}$, $s_j \in \{0,1\}$

Output : $NAF(s)=\sum_{j=0}^{l} s'_j 2^j$, $s'_j \in \{-1,0,1\}$

1. $c_0 \leftarrow 0, s_l \leftarrow 0$
2. For $j=0$ to l do:
   $c_{j+1} \leftarrow \lfloor (s_j + s_{j+1} + c_j)/2 \rfloor$
   $s'_j \leftarrow s_j + c_j - 2c_{j+1}$
3. return $NAF(S)=(s'_l s'_{l-1} \cdots s'_0)$

【０２６８】

ＮＡＦは，非零のビットの数が最も少ない表現法である。非零のビットの箇所では、因子の加算もしくは減算を行なうため、非零のビットの数が少ない方が、スカラー倍算を高速に計算することが出来る。ＮＡＦを用いたスカラー値表現を、ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ法，ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｗｉｎｄｏｗ法に適用することが出来る。ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ法，ｈａｌｖｅ－ａｎｄ－ａｄｄ　ｗｉｎｄｏｗ法で用いる　ＨＥＣ＿ＨＬＶ　はランダムカーブを用いたアルゴリズム５ａ［Ａｌｇｏｒｉｔｈｍ　５ａ］のＨＥＣ＿ＨＬＶでも良いし、アルゴリズム８ａ［Ａｌｇｏｒｉｔｈｍ　８ａ］の曲線パラメータにｈ$_{2}$＝１，ｆ$_{4}$＝０の制約を設けたＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ　１０ａ］の曲線パラメータにｈ$_{2}$＝ｈ$_{1}$＝１，ｆ$_{4}$＝０の制約を設けたＨＥＣ＿ＨＬＶでも良いし、アルゴリズム１０ａ［Ａｌｇｏｒｉｔｈｍ　１０ａ］の曲線パラメータにｈ$_{2}$＝ｈ$_{1}$＝ｈ$_{0}$＝１，ｆ$_{4}$＝０の制約を設けたＨＥＣ＿ＨＬＶでも良い。ＮＡＦを用いたｈａｌｖｅ－ａｎｄ－ａｄｄ　ｂｉｎａｒｙ方を　アルゴリズム１６ａ［Ａｌｇｏｒｉｔｈｍ　１６ａ］として以下に示す。

【数４３】

Algorithm16a *Halve – and – add NAF Binary Method*

Input : $D \in \mathbf{J}(\mathrm{F}_{2^n})$ *such that* $2D \neq O$, $d \in \mathrm{Z}$, r : order of D, $m = \lfloor \log_2 r \rfloor$

Output : *scalar multiplication* dD

1. $\sum_{i=0}^{m} \hat{d}_i 2^i \leftarrow NAF(2^m d \mod r)$, $\hat{d}_i \in \{-1, 0, 1\}$
2. $\sum_{i=0}^{m} d_i / 2^i \leftarrow \sum_{i=0}^{m} \hat{d}_i 2^{i-m}$, $d_i \in \{-1, 0, 1\}$
3. $Q \leftarrow O$
4. for i from m downto 0 do:
   $Q \leftarrow HEC\_HLV(Q)$
   if $d_i > 0$ then $Q \leftarrow Q + D$
   if $d_i < 0$ then $Q \leftarrow Q - D$
5. return Q

【０２６９】

［演算高速化の検証］

次に、上述した処理例７～１１を適用した演算における計算量を求め、演算の高速化について検証する。

【０２７０】

ＨＥＣ＿ＨＬＶ（ｈ$_2$＝１，ｆ$_4$＝０）では、計算量は、平均で、

１９．５Ｍ＋２Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ

である。

【０２７１】

まず、有限体が正規基底で定義されている場合について考える。前述のとおり、正規基底を用いた場合、Ｍ，Ｉのみの計算量を考慮すればよい。文献［A.Menezes. Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers, 1993.］によると、有限体をＦｑ，ｑ＝２^{n}とした場合、逆元計算一回は、下式によって算出される回数、すなわち、

【数４４】

$$\lfloor \log_2(n-1) \rfloor + w(n-1) - 1$$

【０２７２】
上記式によって算出される回数の乗算に匹敵する。ここで、ｗ（ｎ－１）はｎ－１を２進数で表示した際の１の個数を表す。例えば、ｎ＝８３，８９，１１３の場合、Ｉ＝８Ｍとなり、ｎ＝１０３の場合、Ｉ＝９Ｍとなる。

【０２７３】
ここで、Ｉ＝８Ｍを仮定すると、
ＨＥＣ＿ＨＬＶ（ｈ$_2$＝１，ｆ$_4$＝０）
の計算量は、
１９．５Ｍ＋１Ｉ＝２７．５Ｍとなる。

【０２７４】
一方、ＨａｒｌｅｙＤＢＬでは、
２１Ｍ＋１Ｉ＝２９Ｍとなり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも５％ほど高速となる。また、テーブル参照法を用いた場合では、
１８Ｍ＋１Ｉ＝２６Ｍ
となり、ＨＥＣ＿ＨＬＶは、ＨａｒｌｅｙＤＢＬよりも１０％ほど高速となる。

【０２７５】
また、ＨＥＣ＿ＨＬＶ（ｈ$_2$＝ｈ$_1$＝１，ｆ$_4$＝０）
の計算量は、平均で、
１４．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ
となる。この場合は、
１４．５Ｍ＋１Ｉ＝２２．５Ｍ
となる。

【０２７６】
一方、ＨａｒｌｅｙＤＢＬでは、
１８Ｍ＋１Ｉ＝２６Ｍとなり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも１３％ほど高速となる。また、テーブル参照法を用いた場合では、
１４Ｍ＋１Ｉ＝２２Ｍ
となり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも１５％ほど高速となる。

【０２７７】
また、ＨＥＣ＿ＨＬＶ（ｈ$_2$＝ｈ$_1$＝ｈ$_0$＝１，ｆ$_4$＝０）
の計算量は、平均で、
１３．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ
となる。この場合は、
１３．５Ｍ＋１Ｉ＝２１．５Ｍ
となる。

【０２７８】
一方、ＨａｒｌｅｙＤＢＬでは、
１５Ｍ＋１Ｉ＝２３Ｍとなり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも６％ほど高速となる。また、テーブル参照法を用いた場合では、
１７Ｍ＋１Ｉ＝２５Ｍ
となり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも１４％ほど高速となる。
【０２７９】
次に、多項式基底の場合の計算量を評価する。Ｓ，Ｉ，ＳＲ，Ｈ，Ｔの計算量をそれぞれＳ＝０．１Ｍ，Ｉ＝８Ｍ，ＳＲ＝０．５Ｍ，Ｈ＝０．５Ｍ，Ｔ＝０Ｍと仮定する。ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）では、計算量は平均で、
１９．５Ｍ＋２Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ＝２９．９５Ｍ
となる。
一方、ＨａｒｌｅｙＤＢＬでは、
２１Ｍ＋５Ｓ＋１Ｉ＝２９．５Ｍ
となり、ＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも１％程度高速となる。
【０２８０】
また、テーブル参照法を用いた場合は、
１８Ｍ＋２Ｓ＋１Ｉ＋２ＳＲ＋２Ｈ＝２８．２Ｍ
となり、ＨＥＣ＿ＨＬＶはＨａｒｌｅｙＤＢＬよりも４％程度高速となる。
また、ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$）では、計算量は平均で、
１４．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ＝２５．０５Ｍ
となる。一方、ＨａｒｌｅｙＤＢＬでは、
１８Ｍ＋７Ｓ＋１Ｉ＝２６．７Ｍ
となり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも６％程度高速となる。
【０２８１】
また、テーブル参照法を用いた場合は、
１４Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈ＝２４．３Ｍ
となり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも９％程度高速となる。
【０２８２】
また、ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）の計算量は、
１３．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔ＝２４．０５Ｍ
となる。一方、ＨａｒｌｅｙＤＢＬでは、
１５Ｍ＋７Ｓ＋１Ｉ＝２３．７Ｍ
となり、ＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも１％程度高速となる。また、テーブル参照法を用いた場合は、
１３Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈ＝２３．３Ｍ
となり、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも２％程度高速となる。
【０２８３】
また、多項式基底の場合をソフトウェア実装を行い、速度の比較を行った。
ＣＰＵは　ＰｅｎｔｉｕｍＩＩ　３００ＭＨｚ、
ＯＳはＲｅｄＨａｔ７．３、
コンパイラはｇｃｃ２．９６
の環境でソフトウェア実装を行った。
【０２８４】
Ｍ（乗算）、Ｓ（２乗算）は、文献［D.Hankerson, J.Hernandez, and A.Menezes. Software Implementation of Elliptic Curve Cryptography over Binary Fields. CHES 2000, LNCS 1965, pp.1-24, 2000.，Ａｌｇｏｒｉｔｈｍ４．６，４．７］、Ｉ（逆元演算）は、文献［S.Shantz. From Euclid's GCD to Montgomery Multiplication to the Great Divide. TR-2001-95, Sun Microsystems, Inc., 2001.］、ＳＲ（平方根演算）、Ｔ（トレース（二次方程式の根があるか否かの判定））は、文献［K.Fong, D.Hankerson, J.Lop

ez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18.pdf］、H（ハーフトレース（二次方程式の根を求める演算））は、文献［K.Fong, D.Hankerson, J.Lopez, and A.Menezes. Field inversion and point halving revised. Technical Report CORR2003-18, http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-18.pdf Ａｌｇｏｒｉｔｈｍ ４．７］に従った。

【０２８５】

ｎ＝８３，８９，１１３の３つの有限体に対して，Ｍ，Ｓ，Ｉ，ＳＲ，Ｈ，Ｔを実装し、Ｍとの比を求めた。ここで、

ｎ＝８３の既約多項式は、

$z^{83}+z^{7}+z^{4}+z^{2}+1=0$

ｎ＝８９の既約多項式は、

$z^{89}+z^{38}+1=0$

ｎ＝１１３の既約多項式は、

$z^{113}+z^{9}+1=0$

を用いた。

計算量は次の通りとなった。

ｎ＝８３：Ｓ／Ｍ＝０．１２，Ｉ／Ｍ＝７．９６，ＳＲ／Ｍ＝０．５７，Ｈ／Ｍ＝０．５８

ｎ＝８９：Ｓ／Ｍ＝０．０５，Ｉ／Ｍ＝８．７４，ＳＲ／Ｍ＝０．１４，Ｈ／Ｍ＝０．６１

ｎ＝１１３：Ｓ／Ｍ＝０．０６，Ｉ／Ｍ＝８．５６，ＳＲ／Ｍ＝０．１０，Ｈ／Ｍ＝０．５０

【０２８６】

これらを、ＨａｒｌｅｙＤＢＬの計算量２１Ｍ＋５Ｓ＋１Ｉに適用すると次のようになる。

ｎ＝８３：ＨａｒｌｅｙＤＢＬ２９．５６Ｍ

ｎ＝８９：ＨａｒｌｅｙＤＢＬ２９．９９Ｍ

ｎ＝１１３：ＨａｒｌｅｙＤＢＬ２９．８６Ｍ

【０２８７】

これらを、ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）の計算量、１９．５Ｍ＋２Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔに適用すると次のようになる。

ｎ＝８３：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）３０．２８５Ｍ

ｎ＝８９：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）２９．９１Ｍ

ｎ＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）２９．４３Ｍ

【０２８８】

この場合、ｎ＝８３でＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも２％程度高速である。また、ｎ＝８９，１１３で，それぞれＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりもそれぞれ、０．３％，１．５％程度高速である。

【０２８９】

さらに、ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）にテーブル参照法を適用した場合の計算量、１８Ｍ＋２Ｓ＋１Ｉ＋２ＳＲ＋２Ｈに適用すると次のようになる。

ｎ＝８３：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２８．５Ｍ

ｎ＝８９：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２８．３４Ｍ

ｎ＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２７．８８Ｍ

【０２９０】

この場合、ｎ＝８３，８９，１１３で，それぞれＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤ

ＢＬよりも，４％，５％，６％程度高速である。
【０２９１】
また、$h_2=h_1=1$，$f_4=0$の場合は次の通りとなる。
ＨａｒｌｅｙＤＢＬの計算量１８Ｍ＋７Ｓ＋１Ｉに適用すると次のようになる。
ｎ＝８３：ＨａｒｌｅｙＤＢＬ２７．４Ｍ
ｎ＝８９：ＨａｒｌｅｙＤＢＬ２７．０９Ｍ
ｎ＝１１３：ＨａｒｌｅｙＤＢＬ２６．９８Ｍ
【０２９２】
次に、ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$）の計算量、１４．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）２５．４０５Ｍ
ｎ＝８９：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）２４．９６Ｍ
ｎ＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=1$，$f_4=0$）２４．４９Ｍ
この場合、ｎ＝８３，８９，１１３で、それぞれＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも、７％，８％，１０％程度高速である。
【０２９３】
さらに、ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$）にテーブル参照法を適用した場合の計算量、１４Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２４．６２Ｍ
ｎ＝８９：ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２４．３９Ｍ
ｎ＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２３．９４Ｍ
この場合、ｎ＝８３，８９，１１３で、それぞれＨＥＣ＿ＨＶＬの方がＨａｒｌｅｙＤＢＬよりも，１０％，８％，１１％程度高速である。
【０２９４】
また、$h_2=h_1=h_0=1$，$f_4=0$の場合は次の通りとなる。
ＨａｒｌｅｙＤＢＬの計算量１５Ｍ＋７Ｓ＋１Ｉに適用すると次のようになる。
ｎ＝８３：ＨａｒｌｅｙＤＢＬ２３．８Ｍ
ｎ＝８９：ＨａｒｌｅｙＤＢＬ２４．０９Ｍ
ｎ＝１１３：ＨａｒｌｅｙＤＢＬ２３．９８Ｍ
【０２９５】
次に、ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）の計算量１３．５Ｍ＋３Ｓ＋１Ｉ＋２．５ＳＲ＋２Ｈ＋２Ｔに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）２４．４０５Ｍ
ｎ＝８９：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）２３．９６Ｍ
ｎ＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）２３．４９Ｍ
この場合、ｎ＝８３でＨａｒｌｅｙＤＢＬの方がＨＥＣ＿ＨＬＶよりも、２％程度高速である。また、ｎ＝８９，１１３では、それぞれＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも０．５％，２％程度高速である。
【０２９６】
さらに、ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$）にテーブル参照法を適用した場合の計算量１３Ｍ＋３Ｓ＋１Ｉ＋２ＳＲ＋２Ｈに適用すると次のようになる。
ｎ＝８３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２３．６２Ｍ
ｎ＝８９：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２３．３９Ｍ
ｎ＝１１３：ＨＥＣ＿ＨＬＶ（$h_2=h_1=h_0=1$，$f_4=0$，ｗｉｔｈｔａｂｌｅ－ｌｏｏｋｕｐ）２２．９４Ｍ

この場合、ｎ＝８３，８９，１１３で，それぞれＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも、１％，３％，４％程度高速である。

【０２９７】

以上より、ほとんどの場合で、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも高速であることが言える。特に、曲線パラメータが、$h_2=h_1=1$，$f_4=0$の場合は、全ての場合においてＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも高速である。

【０２９８】

次に、スカラー倍算における計算量を考える。上記の例で、ＨＥＣ＿ＨＬＶの方がＨａｒｌｅｙＤＢＬよりも高速な場合は、加算と１／２倍算の組み合わせによるスカラー倍算の方が、加算と２倍算の組み合わせによるスカラー倍算よりも高速である。次に具体的にスカラー倍算の計算量を比較してみる。曲線は，$h_2=h_1=1$，$f_4=0$を用いる。また、スカラー倍算アルゴリズムは、前述のＮＡＦ＋ｂｉｎａｒｙ法（アルゴリズム１６ａ［Ａｌｇｏｒｉｔｈｍ１６ａ］）を用いる。このアルゴリズムにおいて、ステップ１，２がスカラー倍算全体に占める割合は非常に小さいため、これらの計算量は無視する。ここで、正規基底および多項式基底ともにｎ＝８３，８９，１１３の場合で計算量を考えてみる。また、ベースポイントの位数は、ｎ＝８３，８９，１１３に対してそれぞれ１６５ビット、１７７ビット、２２５ビットと仮定する。また、ステップ４の繰り返し部分では、ベースポイントの位数のビット数だけ繰り返す。因子の加算は文献［T.Lange, Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae. Cryptology ePrint Archive, 2002/121, IACR, 2002.］を用いる。ただし、曲線パラメータは$h_2=h_1=1$，$f_4=0$とする。

【０２９９】

この場合の因子の加算に必要な計算量は２１Ｍ＋３Ｓ＋１Ｉである。スカラー値はＮＡＦを使い｛－１，０，１｝で表現する。スカラー値をｍとした場合、非零のビットは、おおよそｍ／３個ある。よってＮＡＦ＋ｂｉｎａｒｙ法の計算量は、（（加算・減算の計算量）／３＋（１／２倍算ｏｒ２倍算の計算量））ｘ（ベースポイントの位数のビット数）で計算する。

【０３００】

まずは、正規基底の場合から見てみる。Ｉ＝８Ｍと仮定する。

$h_2=h_1=1$，$f_4=0$の場合、

ｎ＝８３：加算・２倍算　５８８５Ｍ

ｎ＝８９：加算・２倍算　６３１３Ｍ

ｎ＝１１３：加算・２倍算　８０２５Ｍ

【０３０１】

$h_2=h_1=1$，$f_4=0$の場合、

ｎ＝８３：加算・１／２倍算　５３０７．５Ｍ

ｎ＝８９：　加算・１／２倍算　５６９３．５Ｍ

ｎ＝１１３：　加算・１／２倍算　７２３７．５Ｍ

【０３０２】

$h_2=h_1=1$，$f_4=0$＋テーブル参照法の場合、

ｎ＝８３：加算・１／２倍算　５２２５Ｍ

ｎ＝８９：加算・１／２倍算　５６０５Ｍ

ｎ＝１１３：加算・１／２倍算　７１２５Ｍ

【０３０３】

次に、多項式基底の場合を考える。

$h_2=h_1=1$，$f_4=0$の場合、

ｎ＝８３：加算・２倍算　６１１６Ｍ

ｎ＝８９：加算・２倍算　６５０５．９３Ｍ

ｎ＝１１３：加算・２倍算　８２４５．５Ｍ

【０３０４】

$h_2=h_1=1$，$f_4=0$の場合、
n＝８３：加算・１／２倍算　５７８６．８２Ｍ
n＝８９：加算・１／２倍算　６１２８．９２Ｍ
n＝１１３：加算・１／２倍算　７６８５．２５Ｍ
【０３０５】
$h_2=h_1=1$，$f_4=0$＋テーブル参照法の場合、
n＝８３：加算・１／２倍算　５６５７．３Ｍ
n＝８９：加算・１／２倍算　６０２８．０３Ｍ
n＝１１３：加算・１／２倍算　７５６１．５Ｍ
【０３０６】
正規基底の場合は１０～１１％程度、多項式基底の場合は５～８％程度（加算・１／２倍算のスカラー倍）の方が（加算・２倍算のスカラー倍）よりも高速であることが言える。
【０３０７】
以上、説明したように、本発明の処理によれば、楕円曲線暗号の１／２倍算を、超楕円曲線暗号に拡張し、高速演算が実現される。
【０３０８】
超楕円曲線上の因子の演算を使う暗号処理演算において、処理の重い演算は因子のスカラー倍算である。上述した本発明の処理によりスカラー倍算を高速化することで超楕円曲線暗号の処理を大幅に改善することができる。
【０３０９】
前述したように、超楕円曲線暗号（ＨＥＣＣ）は楕円曲線暗号（ＥＣＣ）を一般化した概念であり、現在、様々な分野で適用されている楕円曲線暗号（ＥＣＣ）を使った暗号処理、具体的には署名処理、暗号データの生成、復号処理、暗号鍵共有処理、認証処理などに、本発明を適用することが可能である。楕円曲線暗号（ＥＣＣ）における演算処理中のスカラー倍算の部分を、上述したスカラー倍算に置き換えることで、演算の高速化が可能となる。
【０３１０】
［暗号処理装置の機能構成について］
図７に、本発明の暗号処理装置の機能構成を示すブロック図を示す。暗号処理装置１００は、超楕円曲線暗号に基づく暗号処理演算を実行する暗号処理装置１００であり、因子Ｄをベースポイントとして生成するベースポイント生成部１０１、前述の処理例５において説明したテーブル、すなわち、予め固定された因子Ｄについての［$1/2^i D$］計算値に基づいて、k_1，k_1'，（k_0，k_0'）のいずれが正しいかを記録したテーブルを格納した記憶部１０２と、演算実行部１０３を有する。
【０３１１】
演算実行部１０３は、超楕円曲線の因子Ｄに対するスカラー倍算の演算において、演算処理として、１／２倍算を含む演算を実行する。具体的には、種数２、標数２のランダムパラメータを持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する。例えば、種数２、標数２の$h(x)=x^2+x+h_0$，$f_4=0$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算、あるいは、種数２、標数２の$h(x)=x$をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行する。
【０３１２】
［電子署名生成および検証アルゴリズムにおける本発明の適用例］
以下に本発明の処理を適用可能な具体的な暗号処理アルゴリズムの例として、楕円曲線暗号を適用した電子署名生成および検証アルゴリズムであるＥＣＤＳＡ（ＥＣ－Ｄｉｇｉｔａｌ Ｓｉｇｎａｔｕｒｅ Ａｌｇｏｒｉｔｈｍ）のスカラー倍算に本発明の演算手法を適用した場合について説明する。ＩＥＥＥ１３６３によるとＥＣＤＳＡによる署名生成、検証は以下のシーケンスによって実行される。

【０３１３】
（１）入力
（１－１）楕円曲線のドメインパラメータ及びベースポイントＧ（位数ｒ）
（１－２）署名者の秘密鍵ｓ
（１－３）平文Ｍ
（２）鍵生成
（２－１）秘密鍵ｓに対してＷ＝ｓＧを公開鍵とする
（３）署名生成
（３－１）ランダムな整数０＜ｕ＜ｒを生成
（３－２）Ｖ＝ｕＧ＝（x_v、y_v）を計算
（３－３）x_vを整数に変換し、ｉとする
（３－４）ｃ＝ｉ　ｍｏｄ　ｒを計算。ｃ＝０ならｓｔｅｐ３－１へ
（３－５）ｆ＝ｈ（Ｍ）、ｈはｈａｓｈ関数
（３－６）$d=u^{-1}(f+sc) \bmod r$を計算。ｄ＝０ならｓｔｅｐ３－１へ
（３－７）（ｃ、ｄ）を平文Ｍに対する署名にする。
（４）署名検証
（４－１）０＜ｃ＜ｒ、０＜ｄ＜ｒをチェック。該当しない場合ｉｎｖａｌｉｄを出力
（４－２）$h=d^{-1} \bmod r$、$h_1=fh \bmod r$、$h_2=ch \bmod r$を計算。
（４－３）Ｐ＝（x_p、y_p）＝h_1Ｇ＋h_2Ｗを計算。Ｐ＝Ｏならばｉｎｖａｌｉｄを出力
（４－４）x_pを整数に変換し、ｉとする。
（４－５）ｃ'＝ｉ　ｍｏｄ　ｒを計算する。
（４－６）ｃ'＝ｃならばｖａｌｉｄを出力。そうでないならばｉｎｖａｌｉｄを出力。

【０３１４】
上記アルゴリズムにおいて下記の処理が超楕円曲線を用いた本提案法が適用できる箇所である。
（２－１）秘密鍵ｓに対してＷ＝ｓＧを公開鍵とする
（３－２）Ｖ＝ｕＧ＝（x_v、y_v）を計算
（４－３）Ｐ＝（x_p、y_p）＝h_1Ｇ＋h_2Ｗを計算。Ｐ＝Ｏならばｉｎｖａｌｉｄを出力

【０３１５】
これらの各ステップ（２－１），（３－２），（４－３）における演算処理、Ｗ＝ｓＧ、Ｖ＝ｕＧ、Ｐ＝（x_p、y_p）＝h_1Ｇ＋h_2Ｗで、ｓＧ、ｕＧ、h_1Ｇ、h_2Ｗの演算処理は、因子のスカラー倍算処理であり、本発明の適用による高速化が可能となる。さらにｓＧ、ｕＧ、h_1Ｇの演算処理は、固定因子のスカラー倍算処理であり、本発明のテーブル参照法の適用による高速化が可能となる。

【０３１６】
［暗号処理装置のハードウェア構成例］
最後に、上述の暗号処理を実行するデバイスとしてのＩＣモジュール２００の構成例を図８に示す。上述の処理は、例えばＰＣ、ＩＣカード、リーダライタ、その他、様々な情報処理装置において実行可能であり、図８に示すＩＣモジュール２００は、これら様々な機器に構成することが可能である。

【０３１７】
図８に示すＣＰＵ(Central processing Unit)２０１は、暗号処理の開始や、終了、データの送受信の制御、各構成部間のデータ転送制御、その他の各種プログラムを実行するプロセッサである。メモリ２０２は、ＣＰＵ２０１が実行するプログラム、あるいは演算パラメータとしての固定データを格納するＲＯＭ（Read-Only-Memory）、ＣＰＵ２０１の処理において実行されるプログラム、およびプログラム処理において適宜変化するパラメ

ータの格納エリア、ワーク領域として使用されるＲＡＭ（Random Access Memory）等からなる。

【０３１８】

なお、メモリ２０２に格納する演算実行プログラムは、上述したベースポイントの設定処理、スカラー倍算としての加算、２倍算の実行シーケンスを含むプログラムとして設定される。また、メモリ２０２は暗号処理に必要な鍵データ等の格納領域として使用可能である。データ等の格納領域は、耐タンパ構造を持つメモリとして構成されることが好ましい。

【０３１９】

暗号処理手部２０３は、上述したスカラー倍算処理を含む暗号処理、復号処理等を実行する。なお、ここでは、暗号処理手段を個別モジュールとした例を示したが、このような独立した暗号処理モジュールを設けず、例えば暗号処理プログラムをＲＯＭに格納し、ＣＰＵ２０１がＲＯＭ格納プログラムを読み出して実行するように構成してもよい。

【０３２０】

乱数発生器２０４は、暗号処理に必要となる鍵の生成などにおいて必要となる乱数の発生処理を実行する。

【０３２１】

送受信部２０５は、外部とのデータ通信を実行するデータ通信処理部であり、例えばリーダライタ等、ＩＣモジュールとのデータ通信を実行し、ＩＣモジュール内で生成した暗号文の出力、あるいは外部のリーダライタ等の機器からのデータ入力などを実行する。

【０３２２】

以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、特許請求の範囲の欄を参酌すべきである。

【０３２３】

なお、明細書中において説明した一連の処理はハードウェア、またはソフトウェア、あるいは両者の複合構成によって実行することが可能である。ソフトウェアによる処理を実行する場合は、処理シーケンスを記録したプログラムを、専用のハードウェアに組み込まれたコンピュータ内のメモリにインストールして実行させるか、あるいは、各種処理が実行可能な汎用コンピュータにプログラムをインストールして実行させることが可能である。

【０３２４】

例えば、プログラムは記録媒体としてのハードディスクやＲＯＭ（Read Only Memory）に予め記録しておくことができる。あるいは、プログラムはフレキシブルディスク、ＣＤ－ＲＯＭ(Compact Disc Read Only Memory)，ＭＯ(Magneto optical)ディスク，ＤＶＤ(Digital Versatile Disc)、磁気ディスク、半導体メモリなどのリムーバブル記録媒体に、一時的あるいは永続的に格納（記録）しておくことができる。このようなリムーバブル記録媒体は、いわゆるパッケージソフトウエアとして提供することができる。

【０３２５】

なお、プログラムは、上述したようなリムーバブル記録媒体からコンピュータにインストールする他、ダウンロードサイトから、コンピュータに無線転送したり、ＬＡＮ(Local Area Network)、インターネットといったネットワークを介して、コンピュータに有線で転送し、コンピュータでは、そのようにして転送されてくるプログラムを受信し、内蔵するハードディスク等の記録媒体にインストールすることができる。

【０３２６】

なお、明細書に記載された各種の処理は、記載に従って時系列に実行されるのみならず、処理を実行する装置の処理能力あるいは必要に応じて並列的にあるいは個別に実行されてもよい。また、本明細書においてシステムとは、複数の装置の論理的集合構成であり、

各構成の装置が同一筐体内にあるものには限らない。

【産業上の利用可能性】

【０３２７】

本発明の構成によれば、楕円曲線暗号の１／２倍算を、超楕円曲線暗号に拡張し、高速演算が実現される。超楕円曲線上の因子の演算を使う暗号処理演算において、処理の重い演算は因子のスカラー倍算であり、本発明の処理によりスカラー倍算を高速化することで超楕円曲線暗号の処理を大幅に改善することができ、ＩＣカードなど高速性、安全性の要求される暗号処理演算を行う機器、デバイスなどにおいて本発明の適用が可能である。

【０３２８】

本発明の構成によれば、超楕円曲線暗号の因子Ｄに対するスカラー倍算において、演算処理として、１／２倍算を含む演算を実行することでスカラー倍算を高速化することができる。例えば、種数２、標数２のｈ（ｘ）＝ｘ^{2}＋ｘ＋ｈ$_{0}$，ｆ$_{4}$＝０をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算、あるいは、種数２、標数２のｈ（ｘ）＝ｘ^{2}＋ｈ$_{1}$ｘ＋ｈ$_{0}$，ｆ$_{4}$＝０をパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算、あるいは、種数２、標数２のｈ（ｘ）＝ｘをパラメータに持つ超楕円曲線の因子Ｄに対するスカラー倍算において１／２倍算を含む演算を実行することで、高速演算が実現され、ＩＣカードなど高速性、安全性の要求される暗号処理演算を行う機器、デバイスなどにおいて本発明の適用が可能である。

る。

【０３２９】

さらに、本発明の構成によれば、予め固定された因子Ｄについての［１／２^{i}Ｄ］計算値に基づいて、ｋ$_{1}$，ｋ$_{1}$'，（ｋ$_{0}$，ｋ$_{0}$'）のいずれが正しいかを記録したテーブルを適用することで、因子のスカラー倍算における計算量をさらに削減することが可能となり、さらなる高速化が実現され、ＩＣカードなど高速性、安全性の要求される暗号処理演算を行う機器、デバイスなどにおいて本発明の適用が可能である。

【０３３０】

さらに、本発明の構成によれば、超楕円曲線暗号の因子Ｄに対するスカラー倍算において、演算処理として、１／２倍算を含む演算を実行する構成とするとともに、１／２倍算の演算処理において逆元演算の実行回数を減少させるアルゴリズムを適用したので、因子のスカラー倍算における計算量をさらに削減することが可能となり、さらなる高速化が実現される。

【図面の簡単な説明】

【０３３１】

【図１】種数２の超楕円曲線暗号のスカラー倍算における加算処理、２倍算処理のアルゴリズムについて説明する図である。

【図２】種数２の超楕円曲線暗号の１／２倍算の場合分け処理について説明する図である。

【図３】ＨＥＣ＿ＨＬＶアルゴリズムについて説明するフローチャートを示す図である。

【図４】ＨＥＣ＿ＨＬＶ$^{2\rightarrow 1+1}$アルゴリズムについて説明するフローチャートを示す図である。

【図５】ＨＥＣ＿ＨＬＶ$^{2\rightarrow 2+2}$アルゴリズムについて説明するフローチャートを示す図である。

【図６】ＨＥＣ＿ＨＬＶ$^{1\rightarrow 2+2}$アルゴリズムについて説明するフローチャートを示す図である。

【図７】本発明の暗号処理演算を実行する暗号処理装置の機能構成を示すブロック図である。

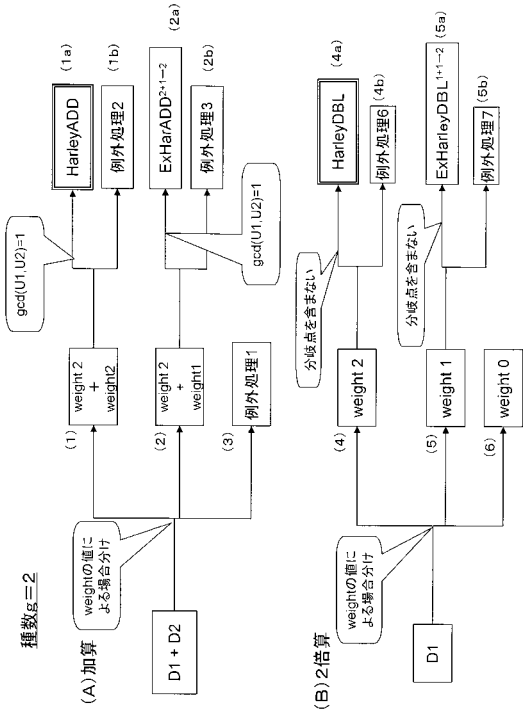
【図８】本発明の暗号処理演算を実行する暗号処理実行デバイス例としてのＩＣモジュールの構成例を示す図である。

【符号の説明】

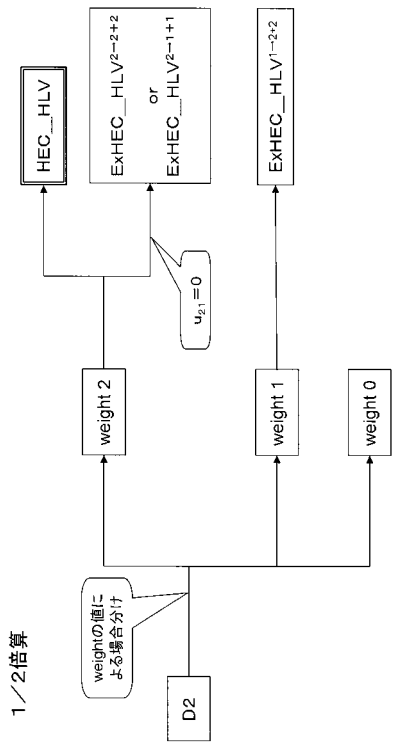
【０３３２】

１００　暗号処理装置
１０１　ベースポイント生成部
１０２　記憶部
１０３　演算実行部
２００　ＩＣモジュール
２０１　ＣＰＵ(Central Processing Unit)
２０２　メモリ
２０３　暗号処理部
２０４　乱数発生器
２０５　送受信部

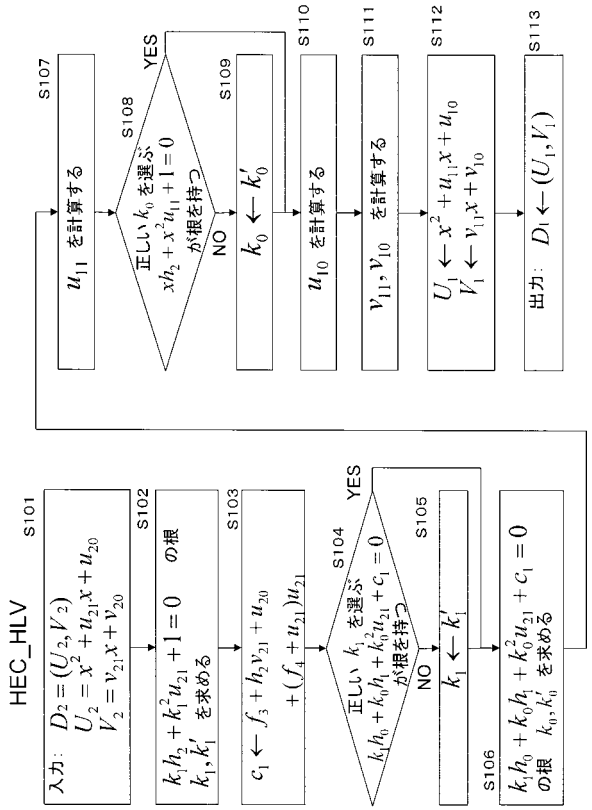
【図１】



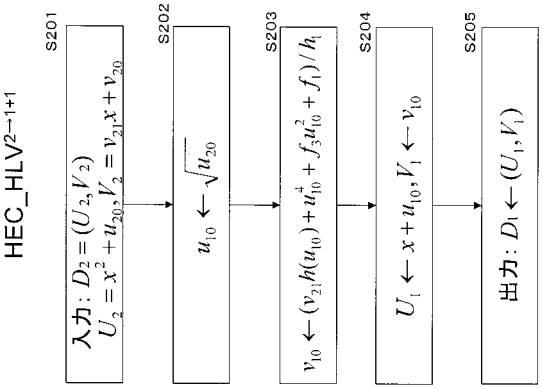
【図２】



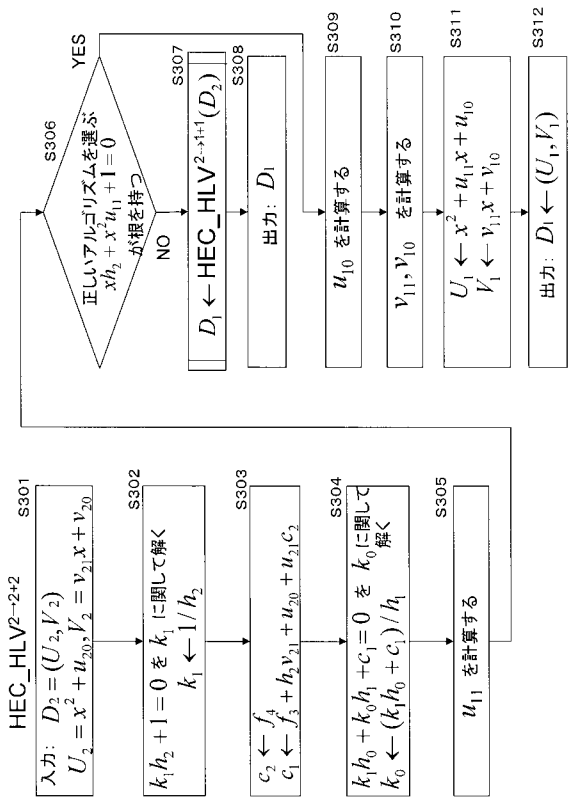
【図３】


HEC_HLV
S101
入力： $D_2=(U_2,V_2)$ $U_2=x^2+u_{21}x+u_{20}$ $V_2=v_{21}x+v_{20}$
S102
$k_1h_2+k_1^2u_{21}+1=0$ の根 k_1,k_1' を求める
S103
$c_1 \leftarrow f_3+h_2v_{21}+u_{20}+(f_4+u_{21})u_{21}$
S104
正しい k_1 を選ぶ $k_1h_0+k_0h_1+k_0^2u_{21}+c_1=0$ が根を持つ
YES
NO
S105
$k_1 \leftarrow k_1'$
S106
$k_1h_0+k_0h_1+k_0^2u_{21}+c_1=0$ の根 k_0,k_0' を求める
S107
u_{11} を計算する
S108
正しい k_0 を選ぶ $xh_2+x^2u_{11}+1=0$ が根を持つ
YES
NO
S109
$k_0 \leftarrow k_0'$
S110
u_{10} を計算する
S111
v_{11},v_{10} を計算する
S112
$U_1 \leftarrow x^2+u_{11}x+u_{10}$ $V_1 \leftarrow v_{11}x+v_{10}$
S113
出力： $D_1 \leftarrow (U_1,V_1)$

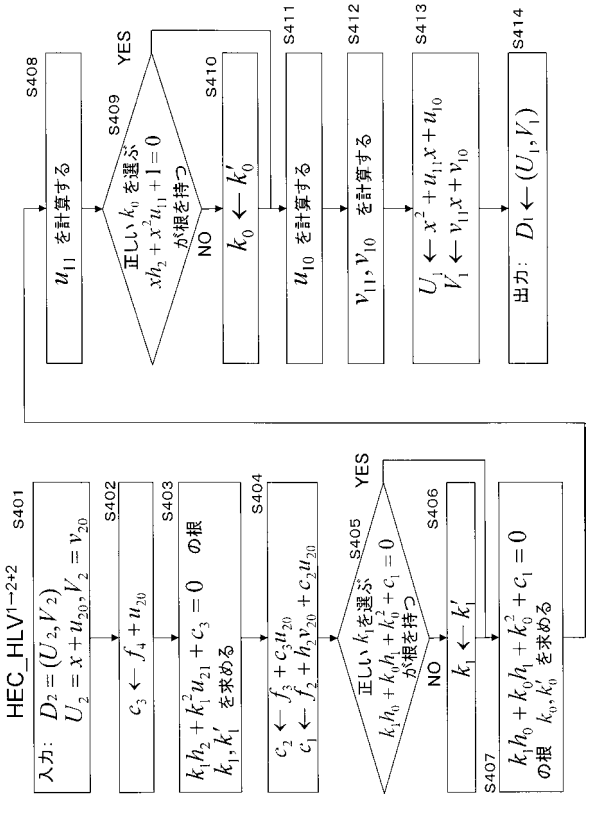
【図４】


HEC_HLV$^{2\rightarrow 1+1}$
S201
入力： $D_2=(U_2,V_2)$ $U_2=x^2+u_{20},V_2=v_{21}x+v_{20}$
S202
$u_{10} \leftarrow \sqrt{u_{20}}$
S203
$v_{10} \leftarrow (v_2h(u_{10})+u_{10}^4+f_3u_{10}^2+f_1)/h_1$
S204
$U_1 \leftarrow x+u_{10},V_1 \leftarrow v_{10}$
S205
出力： $D_1 \leftarrow (U_1,V_1)$

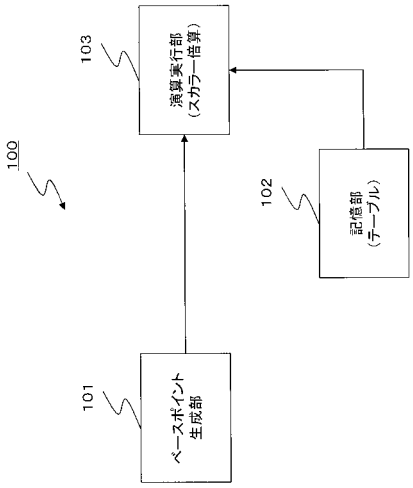
【図５】


HEC_HLV$^{2\rightarrow 2+2}$
S301
入力： $D_2=(U_2,V_2)$ $U_2=x^2+u_{20},V_2=v_{21}x+v_{20}$
S302
$k_1h_2+1=0$ を k_1 に関して解く $k_1 \leftarrow 1/h_2$
S303
$c_2 \leftarrow f_4$ $c_1 \leftarrow f_3+h_2v_{21}+u_{20}+u_2c_2$
S304
$k_1h_0+k_0h_1+c_1=0$ を k_0 に関して解く $k_0 \leftarrow (k_1h_0+c_1)/h_1$
S305
u_{11} を計算する
S306
正しいアルゴリズムを選ぶ $xh_2+x^2u_{11}+1=0$ が根を持つ
YES
NO
S307
$D_1 \leftarrow$ HEC_HLV$^{2\rightarrow 1+1}(D_2)$
S308
出力： D_1
S309
u_{10} を計算する
S310
v_{11},v_{10} を計算する
S311
$U_1 \leftarrow x^2+u_{11}x+u_{10}$ $V_1 \leftarrow v_{11}x+v_{10}$
S312
出力： $D_1 \leftarrow (U_1,V_1)$

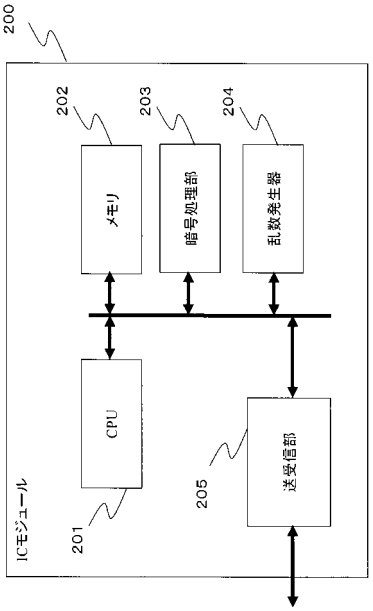
【図６】


HEC_HLV$^{1\rightarrow 2+2}$
S401
入力： $D_2=(U_2,V_2)$ $U_2=x+u_{20},V_2=v_{20}$
S402
$c_3 \leftarrow f_4+u_{20}$
S403
$k_1h_2+k_1^2u_{21}+c_3=0$ の根 k_1,k_1' を求める
S404
$c_2 \leftarrow f_3+c_3u_{20}$ $c_1 \leftarrow f_2+h_2v_{20}+c_2u_{20}$
S405
正しい k_1 を選ぶ $k_1h_0+k_0h_1+k_0^2+c_1=0$ が根を持つ
YES
NO
S406
$k_1 \leftarrow k_1'$
S407
$k_1h_0+k_0h_1+k_0^2+c_1=0$ の根 k_0,k_0' を求める
S408
u_{11} を計算する
S409
正しい k_0 を選ぶ $xh_2+x^2u_{11}+1=0$ が根を持つ
YES
NO
S410
$k_0 \leftarrow k_0'$
S411
u_{10} を計算する
S412
v_{11},v_{10} を計算する
S413
$U_1 \leftarrow x^2+u_{11}x+u_{10}$ $V_1 \leftarrow v_{11}x+v_{10}$
S414
出力： $D_1 \leftarrow (U_1,V_1)$

【図７】


100
101
ベースポイント
生成部
102
記憶部
（テーブル）
103
演算実行部
（スカラー倍算）

【図８】


200
ICモジュール
201
CPU
202
メモリ
203
暗号処理部
204
乱数発生器
205
送受信部

フロントページの続き

(72)発明者　高木　剛
ドイツ連邦共和国，６４２８９，ダルムシュタット，カロリーネンプラッツ５

審査官　石田　信行

(56)参考文献　特開２００４－２０５８７０（ＪＰ，Ａ）
特表２００３－５０４６９５（ＪＰ，Ａ）
特開２００４－２０５８６９（ＪＰ，Ａ）
特開２００４－２０５８６８（ＪＰ，Ａ）
特開２００３－２１６０２８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０９Ｃ　　１／００