ActionScript による η_T ペアリング演算ライブラリ

伴拓也[†1]　毛利公美[†2]　白石善明[†1]　野口亮司[†3]

サービス利用者が自身のデータを預けるようなサービスでは，SSL/TLS による暗号化通信だけではサービス提供者やサービスサーバへの侵入者による盗聴や改ざん，情報漏洩などの脅威を防ぐことはできない．このようなサービス提供側で生じる脅威に対して，サービス提供者に送信する前に Web ブラウザ上でデータを暗号化することが考えられる．そのような暗号技術の中でも，受信者側の ID 情報のみで暗号化を実現できるペアリングに基づく暗号技術が注目されている．ペアリング演算ライブラリがあれば，Web ブラウザ上でペアリングによる暗号化などを行うような高度な暗号アプリケーションを効率的に開発できる．本稿では，Web アプリケーション開発のための ActionScript で利用できるペアリング演算ライブラリの実装について述べる．実装したライブラリは JavaScript のライブラリと同等の演算性能であることと，Web ブラウザ上で暗号化／復号する ID ベース暗号の実装例を示している．

η_T Pairing Library by ActionScript

Takuya Ban[†1] Masami Mohri[†2] Yoshiaki Shiraishi[†1] Ryoji Noguchi[†3]

Threats such as eavesdropping, falsification, and leak of information by the service administrator or the intruder to the service server cannot be prevented only by encrypted communication by SSL/TLS. Data encryption on user's web browser is one of the protecting methods against the threat caused on service providing side. Pairing-based cryptography is paid to attention for developing cryptographic or secure application. Pairing computation library helps web service developer to implement secure web application because the implementation of pairing computation is not so easy. In this paper, we show our η_T pairing computation library, the result of computation performance, and an implementation example of ID-based encryption.

[†1] 名古屋工業大学
Nagoya Institute of Technology
[†2] 岐阜大学
Gifu University
[†3] (株) 豊通シスコム
Toyotsu Syscom Corp.

1. はじめに

SSL/TLS による暗号化通信は一般的な Web サービスのセキュリティ確保の手段として用いられている．しかしながら，サービス利用者が自身のデータを預けるようなサービスでは，通信路暗号化だけではサービス提供者やサービスサーバへの侵入者による利用者データの盗聴や改ざん，情報漏洩などの脅威を防ぐことができない．このようなサービス提供側で生じる利用者への脅威に対して，Web ブラウザ上で暗号化／復号を行うことが考えられる．本稿では，Web アプリケーション上でやり取りを行う情報をクライアント側の Web ブラウザ上で暗号化することにより盗聴や改ざんから保護することができるアプリケーションを“セキュアな Web アプリケーション”と呼ぶ．

ファイルを暗号化して送信する場合，受信者の暗号化鍵を渡されていなくても，受信者の ID 情報を用いてファイルを暗号化することができることや，電子署名を従来よりも短くできるなどの従来にない特徴を持つ，または，従来よりも短い鍵長のプロトコルの設計が可能なペアリングという暗号技術が注目されている．ペアリング演算ライブラリがあれば，セキュアな Web アプリケーションの開発者は高度な理論的知識を意識することなく効率的に開発ができる．

本稿では，多様なコンピューティング環境で使用でき，インストールされているランタイム環境のバージョンに大きな違いがない Flash Platform に着目し，ActionScript で使用できる標数 3 の η_T ペアリング演算ライブラリについて述べる．

以下，2 章では，楕円曲線上のペアリングと演算ライブラリの必要性について述べる．3 章では，ライブラリで適用したアルゴリズムとプログラムでの実装方法を示す．4 章では，実装したライブラリの演算性能について，Flash Platform と同様に広く使われている JavaScript でのペアリング演算との比較により述べる．5 章では，BF 方式の ID ベース暗号を例にあげ，本ライブラリによる Web ブラウザ上で暗号化／復号するプログラムの実装方法について述べる．ID ベース暗号における受信者の復号鍵を発行する主体となる PKG を REST スタイルの Web サービスで実装し，Web ブラウザ上の暗号化／復号クライアントからアクセスできることを 6 章では述べる．最後に 7 章で本稿をまとめる．

2. 楕円曲線上のペアリングとライブラリ

2.1 楕円曲線上のペアリング

ペアリングとは，以下のような双線形性を持つ 2 入力 1 出力の写像関数 e である．

$$e:\ G_1 \times G_2 \to G_T,\ \ (P,Q) \mapsto e(P,Q)$$
$$e(P_1 + P_2,\ Q) = e(P_1,Q)\, e(P_2,Q)$$
$$e(P,\ Q_1 + Q_2) = e(P,Q_1)\, e(P,Q_2)$$

ここで，G_1,G_2は加法群，G_Tは乗法群を表し，いずれも位数 r の有限群である．

有限体 F_q 上の楕円曲線を用いたペアリングでは，G_1は E(F_q)の位数 r の部分群，G_2は E(F_{q^k})の位数 r の部分群，G_Tは F_{q^k} の乗法群に含まれる 1 の r 乗根からなる部分群である．ここで整数 k は $r \mid (q^k - 1)$ を満たす最小の k で与えられ，埋め込み次数と呼ばれる．例えば，q=3^mかつ楕円曲線が $E: y^2 = x^3 - x + b,\ \ b \in \{1, -1\}$ である場合，k=6であることが知られている[1].

$\mathrm{E}(F_q)[r] = \{P \in \mathrm{E}(F_q) \mid rP = O\},\ \ \mu_r = \{x \in F_{q^k} \mid x^r = 1\}$ とするとき，以下の式により定義されるペアリングを Tate ペアリングと呼ぶ．

$$e_\gamma:\ \ \mathrm{E}(F_q)[r] \times \mathrm{E}(F_{q^k}) / r\mathrm{E}(F_{q^k}) \to r\mathrm{E}(F_{q^k}) \to \mu_r$$

標数が小さい有限体上の超特異な楕円曲線上で Tate ペアリングを計算する際に，Frobenius 写像を利用することで高速に計算可能であることが Duursma らにより示されている[2]．Barreto らはこれをさらに改良して，η_Tペアリングと呼ばれるペアリング関数を示した[3].

$\mathrm{E}(F_{3^m})[r] = \{P \in \mathrm{E}(F_{3^m}) \mid rP = O\},\ \ \mu_r = F_{3^{6m}} / (F_{3^{6m}})^r$ とするとき，以下の式により定義されるペアリングを標数 3 における η_Tペアリングと呼ぶ．

$$\eta T:\ \ \mathrm{E}(F_{3^m})[r] \times \mathrm{E}(F_{3^m})[r] \to \mu_r$$

拡大体 F_{3^m} に対して定義される超特異な楕円曲線 $E: y^2 = x^3 - x + b,\ \ b \in \{1, -1\}$ 上での η_Tペアリングの計算手順を図 1 に示す．なお，σ は $\sigma^2 + 1 = 0$ を満たす $F_{3^{2m}}$ の元，ρ は $\rho^3 - \rho - 1 = 0$ を満たす $F_{3^{3m}}$ の元を表す．

図 1 のアルゴリズムにおける計算は大きく分けてループ部分（3 行目~9 行目）と最終冪（10 行目）から構成されている．標数 3 の η_Tペアリングでは，ループ部分の繰り返し回数が通常の Tate ペアリングの約半分に削減される．

2.2 ペアリング演算ライブラリ

ペアリング演算を行うためには，ペアリング関数の引数となる楕円曲線上の点や戻り値となる拡大体を扱えなければならない．さらに，楕円曲線や拡大体を扱うためには素体となる有限体が必要となる．その演算に関わるコードをセキュアな Web アプリケーション開発に利用するためには，再利用が容易な形でまとめたライブラリとすることが望ましい．

ペアリング演算に必要な関数をまとめてライブラリとすることで，開発者は有限

Input: $P = (x_p, y_p),\ \ Q = (x_q, y_q) \in E(F_{3^m})$
Output: $\eta T\,(P,Q) \in F_{3^{mk}}$

1. *if* $b = 1$ *then* $y_p \leftarrow -y_p$
2. $f \leftarrow \sigma y_q + y_p(\rho - x_p - x_q - 1)$
3. *for* $j \leftarrow 0$ *to* $(m-1)/2$ *do*
4. $u \leftarrow x_p + x_q + b$
5. $g \leftarrow \sigma y_p y_q - u^2 - \rho u - \rho^2$
6. $f \leftarrow fg$
7. $x_p \leftarrow \sqrt[3]{x_p},\ y_p \leftarrow \sqrt[3]{y_p}$
8. $x_q \leftarrow x_q^3,\ y_q \leftarrow y_q^3$
9. *end for*
10. *return* $f^{(3^{3m}-1)(3^m+1)(3^m - b3^{\frac{m+1}{2}}+1)}$

図 1　η_Tペアリングの計算[3]

体や楕円曲線，ペアリング演算のための複雑なアルゴリズムなど，高度な理論的知識を意識することなく，プロトコルや方式を容易に Web アプリケーションへ適用することができる．

C 言語，C++言語によるペアリング演算ライブラリが公開されている[4][5]が，一般的な Web アプリケーションではこれらの言語は使用されていない．Java 言語のライブラリ[6]は，C，C++に比べて Web アプリケーション開発に適しているが，Java では Java 実行環境（JRE）のバージョンによりアプリケーションの動作が異なる．Web アプリケーション開発ではスクリプト言語がよく利用されており，その代表的な言語に JavaScript がある．JavaScript は Web ブラウザがソースコードを解釈して実行することから，JavaScript の解釈が Web ブラウザごとに異なる部分があるので，任意の Web ブラウザで同一の動作をするアプリケーションを開発するのは容易ではない．なお，JavaScript のペアリング演算ライブラリは存在するものの，著者らの知る限り公開されているものはない．

スクリプト言語の一つに Flash コンテンツのコントロールに用いられる ActionScript がある．この言語は SWF 形式でコンパイルされ，クライアント側の仮想マシン上で実行される．この仮想マシン ActionScript Virtual Machine 2（AVM2）は Adobe Flash Player に組み込まれている．ActionScript は仮想マシンのバージョンが同じ，つまり，インストールされている Adobe Flash Player のバージョンが同じならば，クライアントの環境に影響されずに同じ動作をするアプリケーションを容易に開発できる．Adobe Flash Player は Java と比較して広く普及している[7]ことから，セキュ

アな Web アプリケーションを開発する言語として ActionScript に注目する．著者らの知る限り ActionScript によるペアリング演算ライブラリは提供されていない．

3. 開発したライブラリ

3.1 有限体の実装

標数 3 の有限体における η_T ペアリングは，入力を楕円曲線上の点，出力を拡大体の元とする写像関数である．ペアリングを実装するために，まず有限体，拡大体の実装を次のように行う．

3.1.1 拡大体 F_{3^m} の表現

拡大体 F_{3^m} の元 α は，次のような多項式で表すことができる．

$$\alpha = \alpha_{m-1}x^{m-1} + \alpha_{m-2}x^{m-2} + \cdots + \alpha_1 x + \alpha_0$$

拡大体 F_{3^m} の元のプログラム上での表現方法として，次の 3 通りが考えられる．

1. 多項式の係数を 2 ビットで表現し，配列の各要素に格納する
2. 配列を 2 つ用意し，上位ビットと下位ビットに分けて格納する
3. 係数成分 α_i のビット表現を最適化する

1 の表現方法では，多項式の各係数を連続する 2 ビットで表現することから，有限体上の演算を行う際に係数ごとに 2 ビットずつ切り出して演算を行う．さらに，演算結果に対して 3 の剰余を求めるので，処理に時間がかかる．

2 の表現方法では，拡大体 F_{3^m} 上の加算をビット演算のみで実行可能である．拡大体 F_{3^m} 上の加算は，他の演算や拡大体，楕円曲線，ペアリングの演算の速度に影響を与える．2 の表現方法での加算のアルゴリズムには文献[8]によると高速化の余地が残されている．

3 の表現方法は，2 の表現方法での加算と比較してビット演算を 1 回削減したアルゴリズム[8]が与えられている．本ライブラリでは 3 の表現方法を採用する．最適化した係数成分のビット表現を表 1 に示す．また，加算のアルゴリズム[8]を図 2 に示す．

3 の表現方法を用いて実装したクラスである F3_fund の定義を以下に示す．

表 1　係数成分 α_i のビット表現[8]

α_i	α_i^H	α_i^L
0	0	0
1	0	1
2	1	1

Input: $A, B \in F_{3^m}$
Output: $C = A + B (C \in F_{3^m})$

1. $T \leftarrow A^H \; XOR \; B^L$
2. $C^L \leftarrow (A^L XOR \; B^L) \; OR \; (T \; XOR \; B^H)$
3. $C^H \leftarrow (A^L \; XOR \; B^H) \; AND \; T$
4. $return \; C$

図 2　加算のアルゴリズム[8]

【F3_fund：　標数 3 の有限体の元を表現する】

メソッドの使用方法：F3_fund のオブジェクト名.メソッド名(引数)を実行

readString(str:String):void　元を表す{0,1,2}表記の文字列を読み込む
toString():String　元の値を{0,1,2}表記の文字列(String)として出力する
degree():int　標数 3 の有限体の元である多項式の次数を出力する
shift_l(shift:int):void　元の値を左に shift だけシフトする
shift_r(shift:int):void　元の値を右に shift だけシフトする
rawGet(i:int):int　下位の位から i 番目の値を整数型で返す
rawSet(i:int, value:int):void　下位の位から i 番目に値 value をセットする
rand():void　元をランダムに生成する

3.1.2 拡大体 $F_{3^{3m}}$ の表現

拡大体 $F_{3^{3m}}$ 上の元 U は次のような多項式で表現することができる．

$$U = u_2\rho^2 + u_1\rho + u_0$$

ここで，多項式の係数 u_i は拡大体 F_{3^m} の元であり，拡大体 $F_{3^{3m}}$ の法となる既約多項式は $f(\rho) = \rho^3 - \rho - b$ である．

拡大体 $F_{3^{3m}}$ をプログラム上で表現するのに，係数成分を拡大体 F_{3^m} の元（F3_fund）で表し，既約多項式を表現する整数 b も保持する．拡大体 $F_{3^{3m}}$ の元の操作は，係数成分である拡大体 F_{3^m} の元（F3_fund）について操作を行うことで実現する．

拡大体 $F_{3^{3m}}$ を表現するクラスである F3_3_fund の定義を以下に示す．

【F3_3_fund：　標数 3 の有限体における 3 次拡大体の元を表現する】

メソッドの使用方法：F3_3_fund のオブジェクト名.メソッド名(引数)を実行

toString():String　元の値を{0,1,2}表記の文字列(String)として出力する
rand():void　元をランダムに生成する

3.1.3 拡大体 $F_{3^{6m}}$ の表現

拡大体 $F_{3^{6m}}$ の元は $F_{3^{3m}}$ の元 U_0，U_1 を係数に持つ多項式であるが，元を分かりやすく表現するために，拡大体 F_{3^m} の元 $u_0, u_1, u_2, u_3, u_4, u_5$ を係数成分の表現に用いる．また，法となる既約多項式は $f(\sigma) = \sigma^2 + 1$ である．

拡大体 $F_{3^{6m}}$ を表現するクラスである F3_6_fund の定義を以下に示す．

【F3_6_fund：　標数 3 の有限体における 6 次拡大体の元を表現する】

メソッドの使用方法：F3_6_fund のオブジェクト名.メソッド名(引数)を実行

toString():String　元(F3_6)の値を{0,1,2}表記の文字列(String)として出力する
rand():void　元をランダムに生成する

3.2 楕円曲線と MapToPoint

楕円曲線上の点は x 成分，y 成分の 2 つの要素を持つことから，プログラム上では 2 つの拡大体 F_{3^m} の元を属性として持たせる．η_T ペアリングには楕円曲線 $y^2 = x^3 - x + b,\ b \in \{1, -1\}$ を使うことから，b の値はプログラム実行中に変更されない

ように保持する．

拡大体 F_{3^m} の元から楕円曲線上の点を算出する MapToPoint のアルゴリズム[9]を図 3 に示す．このアルゴリズムを用いて，ビット列から楕円曲線上の点を求める．

楕円曲線上の点を表すクラス EC_fund の定義を以下に示す．

【EC_fund： 楕円曲線上の点を表現する】

メソッドの使用方法：EC のオブジェクト名.メソッド名(引数)を実行

toString():String 楕円曲線上の点を{0,1,2}表記の文字列(String)で出力する

isValid():Boolean 楕円曲線上の点として正しい値かどうかを判断する

MapToPoint(a:F3_fund):void 拡大体の元(F3_fund)から楕円曲線上の点を生成する

rand():void 楕円曲線上の点をランダムに生成する

3.3 η_T ペアリング

標数 3 の有限体における η_T ペアリングの演算は，文献[1]をはじめとして様々な高速化手法が提案されているが，文献[1]のアルゴリズムを基に実装を行った．

η_T ペアリングのアルゴリズムを図 4 に示す．なお，拡大次数 m について，m mod 12 = 1 ならば $(\mu,\lambda)=(1,-1)$，m mod 12 = 5 ならば $(\mu,\lambda)=(-1,-1)$，m mod 12 = 7 ならば $(\mu,\lambda)=(-1,1)$，m mod 12 = 11 ならば $(\mu,\lambda)=(1,1)$ とする．

処理 13 は最終冪と呼び，文献[1]に高速なアルゴリズムが示されている．また，最終冪で必要となる冪乗算，$U^{3^{3m}-1}$，U^{3^m+1} の高速なアルゴリズムも同様に文献[1]に示されている．

本ライブラリでは，図 4 の η_T ペアリングの演算，図 5 の最終冪，図 6，図 7 の冪乗算の実装を行った．以下にクラス定義を示す．

Input: $y \in F_{3^m}$
Output: $P=(x,y) \in E(F_{3^m})$

1. $s \leftarrow t \leftarrow c \leftarrow y^2 - b$
2. $r \leftarrow m \mod 3$
3. *for* i *from* 1 *to* $(m-r)/3$
4. $\quad t \leftarrow t^{3^3}$
5. $\quad s \leftarrow s + t$
6. *end for*
7. $s \leftarrow s^3 - s$
8. *if* $r=1$ *then*
9. $\quad s \leftarrow c - s$
10. *end if*
11. *return*(s,y)

図 3　MapToPoint のアルゴリズム[9]

Input: $P=(x_p,y_p),\quad Q=(x_q,y_q) \in E(F_{3^m})$
Output: $\eta_T(P,Q) \in F_{3^{6m}}$

1. $x_p \leftarrow x_p + b$
2. $y_p \leftarrow -\mu b y_p$
3. $x_q \leftarrow x_q^3,\ y_q \leftarrow y_q^3$
4. $t \leftarrow x_p + x_q$
5. $R \leftarrow (\lambda y_p t - \lambda y_q \sigma - \lambda y_p \rho)\cdot(-t^2 + y_p y_q \sigma - t\rho - \rho^2)$
6. *for* i *from* 1 *to* $\dfrac{m-1}{2}$ *do*
7. $\quad R \leftarrow R^3$
8. $\quad x_q \leftarrow x_q^9 - b,\ y \leftarrow -y_q^9$
9. $\quad t \leftarrow x_p + x_q,\ u \leftarrow y_p y_q$
10. $\quad S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$
11. $\quad R \leftarrow R \cdot S$
12. *end for*
13. *return* $R^{(3^{3m}-1)(3^m+1)(3^m-\mu b 3^{\frac{m+1}{2}}+1)}$

図 4　η_T ペアリングのアルゴリズム[1]

Input: $U \in F_{3^{6m}}$
Output: $U^{(3^{3m}-1)(3^m+1)(3^m-\mu b 3^{\frac{m+1}{2}}+1)} \in F_{3^{6m}}$

1. $V \leftarrow U^{3^{3m}-1}$
2. $V \leftarrow V^{3^m+1}$
3. $W \leftarrow V$
4. *for* $i \leftarrow 1$ *to* $\dfrac{m+1}{2}$ *do*
5. $\quad W \leftarrow W^3$
6. *end for*
7. $V \leftarrow V^{3^m+1}$
8. $W' \leftarrow W^{-\mu b}$
9. *return* $V \cdot W'$

図 5　最終冪計算アルゴリズム[1]

Input: $U = u_0 + u_1\sigma + u_2\rho + u_3\sigma\rho + u_4\rho^2 + u_5\sigma\rho^2 \in F_{3^{6m}}$
Output: $V = U^{(3^{3m}-1)} \in F_{3^{6m}}$

1. $m_0 \leftarrow (u_0 + u_2\rho + u_4\rho^2)^2$
2. $m_1 \leftarrow (u_1 + u_3\rho + u_5\rho^2)^2$
3. $m_2 \leftarrow (u_0 + u_2\rho + u_4\rho^2)\cdot(u_1 + u_3\rho + u_5\rho^2)$
4. $a_0 \leftarrow m_0 - m_1,\ a_1 \leftarrow m_0 + m_1$
5. $i \leftarrow a_1^{-1}$
7. $V_0 \leftarrow a_0 \cdot i$
8. $V_1 \leftarrow m_2 \cdot i$
9. *return* $V_0 + V_1\sigma$

図 6　$F_{3^{6m}}$ 上の $U^{3^{3m}-1}$ の計算[1]

Input: $U = u_0 + u_1\sigma + u_2\rho + u_3\sigma\rho + u_4\rho^2 + u_5\sigma\rho^2 \in F_{3^{6m}}$
Output: $V = U^{3^m+1} \in F_{3^{6m}}$

1. $a_0 \leftarrow u_0 + u_1,\ a_1 \leftarrow u_2 + u_3,\ a_2 \leftarrow u_4 + u_5$
2. $m_0 \leftarrow u_0 \cdot u_4, m_1 \leftarrow u_1 \cdot u_5,\ m_2 \leftarrow u_2 \cdot u_4$
3. $m_3 \leftarrow u_3 \cdot u_5,\ m_4 \leftarrow a_0 \cdot a_2\ \ m_5 \leftarrow u_1 \cdot u_2$
4. $m_6 \leftarrow u_0 \cdot u_3,\ m_7 \leftarrow a_0 \cdot a_1,\ m_8 \leftarrow a_1 \cdot a_2$
5. $a_3 \leftarrow m_5 + m_6 - m_7,\ a_4 \leftarrow -m_2 - m_3$
6. $a_5 \leftarrow -m_2 + m_3,\ a_6 \leftarrow -m_0 + m_1 + m_4$
7. *if* $m \mod 6 = 1$ *then*
8. $\quad v_0 \leftarrow 1 + m_0 + m_1 + ba_4$
9. $\quad v_1 \leftarrow bm_5 - bm_6 + a_6$
10. $\quad v_2 \leftarrow -a_3 + a_4$
11. $\quad v_3 \leftarrow m_8 + a_5 - ba_6$
12. $\quad v_4 \leftarrow -ba_3 - ba_4$
13. $\quad v_5 \leftarrow bm_8 + ba_5$
14. *else if* $m \mod 6 = 5$ *then*
15. $\quad v_0 \leftarrow 1 + m_0 + m_1 - ba_4$
16. $\quad v_1 \leftarrow -bm_5 + bm_6 + a_6$
17. $\quad v_2 \leftarrow a_3$
18. $\quad v_3 \leftarrow m_8 + a_5 + ba_6$
19. $\quad v_4 \leftarrow -ba_3 - ba_4$
20. $\quad v_5 \leftarrow -bm_8 - ba_5$
21. *end if*
22. *return* $v_0 + v_1\sigma + v_2\rho + v_3\sigma\rho + v_4\rho^2 + v_5\sigma\rho^2$

図 7　$F_{3^{6m}}$ 上の U^{3^m+1} の計算[1]

【EtatPairing： 標数 3 の η_T ペアリングに関わる計算を行う】

メソッドの使用方法：EtatPairing.メソッド名(引数)を実行

FinalExponentiation(A:F3_6_fund):F3_6_fund η_Tペアリングの最終冪計算を行う

power3_3m_1(X:F3_6_fund):F3_6_fund $F_{3^{6m}}$ 上における $X^{3^{3m}-1}$を計算する

power3m_1(X:F3_6_fund):F3_6_fund $F_{3^{6m}}$ 上における $X^{3^{m}+1}$を計算する

CalcEtatPairing(P:EC_fund, Q:EC_fund) : F3_6_fund 楕円曲線上の 2 点 *P,Q* を引数としてペアリング演算を行い，結果を 6 次拡大体の元で返す

3.4 拡大体と楕円曲線での演算と操作

3.2 節，3.3 節でのアルゴリズムは拡大体と楕円曲線の上での演算を行っている．その演算と必要な操作をするクラスを次のように定義する．

3.4.1 F3_calc

F3_fund の元を演算・操作するクラス F3_calc を次のように定義する．なお，加算，減算では文献[8]のアルゴリズムを利用し，3 乗算では文献[1]のアルゴリズムを利用している．

【F3_calc： 標数 3 の有限体の元による演算関数の実装】

メソッドの使用方法：F3_calc.メソッド名(引数)を実行

copy(a:F3_fund):F3_fund 元 a を複製する

e_judge(a:F3_fund, b:F3_fund):Boolean 元 a, b を比較し，同値かを判断する

digit(a:F3_fund):F3_fund 元 a の次数を法となる既約多項式の次数に合わせる

add(a:F3_fund, b:F3_fund):F3_fund $a+b$ を計算する

neg(a:F3_fund):F3_fund $-a$ を計算する

sub(a:F3_fund, b:F3_fund):F3_fund $a-b$を計算する

mul(a:F3_fund, b:F3_fund):F3_fund $a \cdot b$ を計算する

cube(a:F3_fund):F3_fund a^3を計算する

cubeN(a:F3_fund, n:int):F3_fund a^{3^n} を計算する

inverse(a:F3_fund):F3_fund a^{-1}を計算する

div(a:F3_fund, b:F3_fund):F3_fund a/b を計算する

sqrt(a:F3_fund):F3_fund $\sqrt{a}$ を計算する

cbrt(a:F3_fund):F3_fund $\sqrt[3]{a}$ を計算する

3.4.2 F3_3_calc

F3_3_fund の元を演算・操作するクラス F3_3_calc を次のように定義する．なお，加算，減算，3 乗算では文献[5]のアルゴリズムを利用し，乗算，除算，逆元，2 乗算では文献[1]のアルゴリズムを利用している．

【F3_3_calc： 標数 3 の有限体における 3 次拡大体の元による演算関数の実装】

メソッドの使用方法：F3_3_calc.メソッド名(引数)を実行

copy(a:F3_3_fund):F3_3_fund 元 a を複製する

e_judge(a:F3_3_fund, b:F3_3_fund):Boolean 元 a, b を比較し，同値かを判断する

add(a:F3_3_fund, b:F3_3_fund):F3_3_fund $a+b$ を計算する

neg(a:F3_3_fund):F3_3_fund $-a$ を計算する

sub(a:F3_3_fund, b:F3_3_fund):F3_3_fund $a-b$を計算する

mul(a:F3_3_fund, b:F3_3_fund):F3_3_fund $a \cdot b$ を計算する

inverse(a:F3_3_fund):F3_3_fund a^{-1}を計算する

div(a:F3_3_fund, b:F3_3_fund):F3_3_fund a/b を計算する

cube(a:F3_3_fund):F3_3_fund a^3を計算する

cubeN(a:F3_3_fund, n:int):F3_3_fund a^{3^n} を計算する

sqrt(a:F3_3_fund):F3_3_fund $\sqrt{a}$ の解を求める

square(a:F3_3_fund):F3_3_fund a^2 の解を求める

3.4.3 F3_6_calc

F3_6_fund の元を演算・操作するクラス F3_6_calc を次のように定義する．なお，加算，減算では文献[5]のアルゴリズムを利用し，乗算，3 乗算では文献[1]のアルゴリズムを利用している．

【F3_6_calc： 標数 3 の有限体における 6 次拡大体の元による演算関数の実装】

メソッドの使用方法：F3_6_calc.メソッド名(引数)を実行

copy(a:F3_6_fund):F3_6_fund 元 a を複製する

e_judge(a:F3_6_fund, b:F3_6_fund):Boolean 元 a, b を比較し，同値かを判断する

add(a:F3_6_fund, b:F3_6_fund):F3_6_fund $a+b$ を計算する

neg(a:F3_6_fund):F3_6_fund $-a$ を計算する

sub(a:F3_6_fund, b:F3_6_fund):F3_6_fund $a-b$を計算する

mul(a:F3_6_fund, b:F3_6_fund):F3_6_fund $a \cdot b$ を計算する

cube(a:F3_6_fund):F3_6_fund a^3を計算する

cbrt(a:F3_6_fund):F3_6_fund $\sqrt[3]{a}$ を計算する

3.4.4 EC_calc

EC_fund の演算を行うクラス EC_calc を次のように定義する．なお，加算，減算，3 倍算では文献[5]のアルゴリズムを利用している．

【EC_calc： 楕円曲線上の点による演算関数の実装】

メソッドの使用方法：EC_calc.メソッド名(引数)を実行

copy(P:EC_fund):EC_fund 点 P の複製である点 Q を生成する

e_judge(P:EC_fund, Q:EC_fund):Boolean 点 P, Q を比較し，同値かを判断する

add(P:EC_fund, Q:EC_fund):EC $P+Q$ を計算する

neg(P:EC_fund):EC_fund $-P$ を計算する

sub(P:EC_fund, Q:EC_fund):EC_fund $P-Q$ を計算する

cube(P:EC_fund):EC_fund $3P$ を計算する

cube(P:EC_fund):EC_fund　$3P$ を計算する
exp(n:BigInteger, P:EC_fund):EC_fund　nP を計算する

3.4.5 演算関数のまとめ

ライブラリで実装した演算関数一覧を示す．表 2 に拡大体上の演算関数を示し，表 3 に楕円曲線上の演算関数を示す. なお, 表 2 における F_{3^m} 上の乗算は shift-and-add 法を用いて実装し, 表 3 における楕円曲線上の n 倍算は window 法を用いて実装した．

3.5 ID ベース暗号

η_T ペアリングを用いた ID ベース暗号の実装をサポートすることを目的として, ID ベース暗号に必要となる機能を備えたクラス IDBase を実装した．ID ベース暗号に必要な機能として，公開ハッシュ関数となる拡大体 $F_{3^{6m}}$ の元の SHA-1 のハッシュ値を多倍長整数で算出する機能と, 任意の文字列から EC_fund の MapToPoint 関数を用いて楕円曲線上の点を計算する機能，RFC5408[14]に準拠した PKG へのリクエスト生成機能，Base64 エンコード・デコード機能を実装した．ハッシュ計算の機能と多倍長整数，Base64 のエンコード・デコードを扱う機能は，公開ライブラリである AsCrypto[10]を利用している．IDBase のクラス定義を以下に示す．

【IDBase：　ID ベース暗号で必要なハッシュ関数や MapToPoint 関数，PKG へのリクエストデータの生成関数の実装】

メソッドの使用方法：IDBase.メソッド名(引数)を実行
Hash_SHA1_F3_6(X:F3_6_fund):BigInteger　$F_{3^{6m}}$ の元のハッシュ値(SHA-1)を計算し，多倍長整数で返す
MapToGroup(str:String):EC_fund　String 型文字列から楕円曲線上の点を計算する
Base64enc(str:String):String　String 型文字列を Base64 エンコードする
Base64dec(str:String):String　Base64 エンコードされた文字列をデコードする
Request(algo:String, ID:String):String　ID ベース暗号のアルゴリズム名と ID 情報から RFC5408 に準拠した PKG への復号鍵のリクエストを文字列で生成する

3.6 パラメータの自由度

既存のライブラリ[4][5][6]はいずれも拡大体の拡大次数や既約多項式のパラメータを変更することができない．本ライブラリでは，システム開発の際に活用の幅を持たせるために，拡大体 F_{3^m} における拡大次数 m および既約多項式 $f(x)=x^m+\alpha x^a+\beta$ を容易に変更できるように実装した．F3_fund の変数定義部分に図 8 の記述があり，各パラメータの変更が可能である．η_T ペアリングでは，F_{3^m} について $m=79, 97, 163, 193, 239, 353$ で有用な関数を構成可能であることが知られている[11]ことから，パラメータの自由度を残すことでペアリングの入力点の数に幅を持たせることができる．

3.7 ライブラリの構成と外部ライブラリ

本ライブラリの動作には，As3Crypto[10]という外部ライブラリが必要となる．

表 2　拡大体上の演算関数

	F3_calc	F3_3_calc	F3_6_calc
add（加算）	○[8]	○[5]	○[5]
sub（減算）	○[8]	○[5]	○[5]
mul（乗算）	○*1	○[1]	○[1]
div（除算）	◎	○[1]	－
inverse（逆元）	◎	○[1]	－
square（2 乗算）	－	○[1]	－
cube（3 乗算）	○[1]	○[5]	○[1]
sqrt（平方根）	◎	◎	－
cbrt（立方根）	◎	－	◎

表 3　楕円曲線上の演算関数

	EC_calc
add（加算）	○ [5]
sub（減算）	○ [5]
cube（3 倍算）	○ [5]
exp（n 倍算）	○ *2
MapToPoint	○ [9]

○：既存のアルゴリズムで実装した関数
◎：独自のアルゴリズムで実装した関数
－：η_T ペアリングの演算には必要ない関数
*1：shift-and-add 法
*2：window 法

As3Crypto は ActionScript3.0 で実装された暗号化ライブラリで，多倍長整数（BigInteger）を扱う機能や Base64 エンコード・デコード機能，MD5 や SHA-1 などのハッシュ関数，共通鍵暗号，公開鍵暗号などが実装されている．本ライブラリの EC_calc では，As3Crypto の多倍長整数，IDBase では多倍長整数とハッシュ計算の機能，Base64 エンコード・デコード機能を使用する．

クラスの依存関係を図 9 に示す. 拡大体 $F_{3^{3m}}$，$F_{3^{6m}}$ を表すクラスである F3_3_fund, F3_6_fund と，楕円曲線上の点を表すクラスである EC_fund は，拡大体 F_{3^m} を表すクラス F3_fund を基にしている．さらに，各拡大体の元，楕円曲線上の点を使った演算を行うクラスとして，F3_calc，F3_3_calc，F3_6_calc，EC_calc を実装している．η_T

```
public const field:int = 97;        //  拡大次数m
public const pow:int = 12;          //  α
public const a1:int = 2;            //  α
public const a0:int = 1;            //  β
```

図 8　プログラム上のパラメータの設定

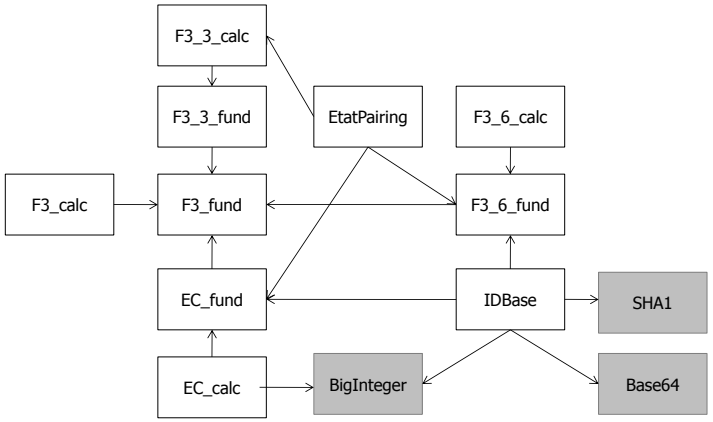


図 9　クラスの依存関係

ペアリングの演算機能を実装したクラス EtatPairing では，楕円曲線上の点 EC_fund からペアリング F3_6_fund を算出する．また，冪乗計算で F3_3_fund を利用している．ID ベース暗号に必要な機能を実装したクラス IDBase は，文字列から楕円曲線上の点 EC_fund を算出するのと，F3_6_fund から AsCrypto のハッシュ関数と多倍長整数を表す BigInteger を利用して，多倍長整数のハッシュ値の計算，Base64 を利用して，Base64 のエンコード・デコードを行う．

3.8 ライブラリの使用方法

本ライブラリは As3Pairing パッケージの中に，合計 10 個の ActionScript ファイル（拡張子.as）が含まれる構成であり，一つの ActionScript ファイルが一つのクラスを表す．本ライブラリを使用する際には，まず，As3Pairing パッケージをプロジェクトに加え，さらに，本ライブラリの動作に必要な As3Crypto ライブラリの全機能が入ったパッケージである，com.hurlant パッケージをプロジェクトに加える．

そして，本ライブラリを利用するプログラムのファイルの先頭に，

import As3Pairing.*;

と記述し，インポートすることで本ライブラリの全機能が利用可能となる．また，本ライブラリで利用している As3Crypto ライブラリの多倍長整数の機能を利用するには，com.hurlant.math.BigInteger をインポートすることで利用可能であり，ハッシュ計算の機能を利用するには，com.hurlant.crypto.Crypto と com.hurlant.crypto.hash.* をインポートすることで利用可能である．

4. 本ライブラリの性能

本ライブラリの速度の性能を示す．まず，性能を比較する対象となるライブラリは次のようにした．本ライブラリと同様にペアリング演算を実装したライブラリとして，PBC[4]，jPBC[6]，光成のライブラリ[5]がある．これらのうち，Web ブラウザ上での動作に適するのは，Java で開発された jPBC である．しかし，jPBC には，本ライブラリで実装した標数 3 の η_T ペアリングの演算が用意されていない．そこで，安全性の基準を同一にした上で，それぞれのペアリング演算の速度を測り，ライブラリの性能を比較する．本ライブラリを利用した標数 3 の η_T ペアリングと jPBC を利用した Tate ペアリングの演算速度を測定する．

安全性の評価指標に MOV security[12]がある．MOV security とは，ペアリングに用いられる有限体の要素数と拡大体の拡大次数により決定される安全性の評価値である．次の MOV security の値における，1 回のペアリング演算の実行時間を測定する．測定には表 4 に示す環境を用いた．

1. MOV security が 1000 程度
2. MOV security が 2000 程度

測定を行った結果を表 5，表 6 に示す．2 つの表より，本ライブラリの演算速度は実行速度に関して既存のライブラリと同程度の性能であることが確認された．

表 4　実行環境

	本ライブラリ	jPBC[6]
CPU	Intel® Core™ 2 Duo CPU E4600 2.40GHz 1.20GHz	
メモリ	2.00GB	
OS	Windows Vista Business SP2	
Web ブラウザ	Firefox3.6.15	
仮想マシン	Adobe Flash Player 10.2.152.26	Java(TM) 6.22

表 5　MOV security が 1000 程度の場合

	本ライブラリ	jPBC[6]
ペアリングの種類	η_Tペアリング	Tate ペアリング
有限体 F_q の要素数 q	3^{97}	512 ビットの素数
拡大次数 k	6	2
MOV security	922	1024
ペアリング演算の実行時間[msec]	288	157

表 6　MOV security が 2000 程度の場合

	本ライブラリ	jPBC[6]
ペアリングの種類	η_Tペアリング	Tate ペアリング
有限体 F_q の要素数 q	3^{193}	1024 ビットの素数
拡大次数 k	6	2
MOV security	1836	2048
ペアリング演算の実行時間 [msec]	1216	3187

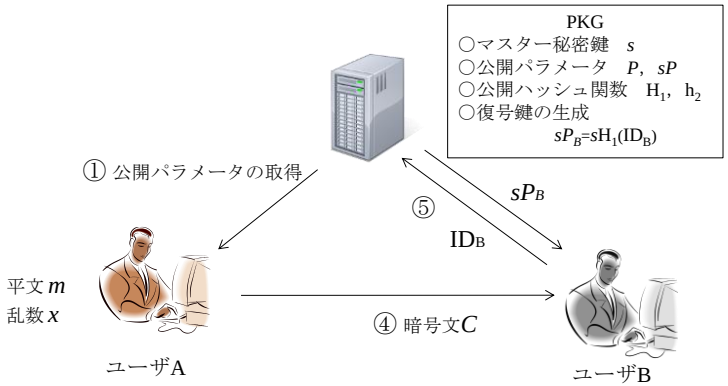
5.　使用例：BF 方式の ID ベース暗号

Web ブラウザ上で暗号化/復号する ID ベース暗号の実装例として BF（Boneh-Franklin）方式の ID ベース暗号[13]を実装する．ID ベース暗号を実現するには，PKG（鍵生成センタ）と呼ばれる，公開パラメータを配布する機能とユーザ ID をクライアントから受け取り，ユーザの復号鍵の生成・配布する機能を備える主体が必要となる．PKG の実装については 6 章で述べる．

5.1　BF 方式

BF 方式は, Boneh と Franklin により定式化された ID ベース暗号の実現例である．BF 方式でユーザ A がユーザ B に暗号通信を行う場合の手順を図 10 に示す．

① ユーザ A は PKG から公開パラメータを取得する
② ユーザ A は公開ハッシュ関数 H_1 により，ユーザ B の ID からユーザ B の公開鍵 P_B を計算する
③ ユーザ A は，暗号文 $C=(c_1, c_2)=(xP, m*h_2(e(P_B, xsP)))$を計算する



② 公開鍵 $P_B=H_1(ID_B)$
③ 暗号化：$C=(c_1,c_2)=(xP, m*h_2(e(P_B, xsP)))$

図 10　BF 方式

④ ユーザ A は暗号文 C をユーザ B に送信する
⑤ ユーザ B は自分の ID を PKG に渡し，秘密裏に復号鍵 sP_B を受け取る
⑥ ユーザ B は $m=c_2/h_2(e(sP_B, c_1))$を計算し復号する

5.2　クライアントの暗号化・復号のコード

ここでは公開パラメータと復号鍵を予めクライアントに渡しているものとする．

実装に用いた環境を表 7 に示す．Flash Builder 4 を用いて，②の公開鍵の計算と③の暗号文 $C=(c_1, c_2)=(xP, m*h_2(e(P_B, xsP)))$を計算するコード（図 11）と⑥の $m=c_2/h_2(e(sP_B, c_1))$を計算する復号のコード（図 12）を作成した．①の公開パラメータの取得と，⑤の復号鍵の受け取りの詳細は 6 章で述べる．

これらのプログラムを実行し，ライブラリの計算は正しく実行されていることを確認した．

6.　PKG の実装

6.1　システム構成

実装した PKG のシステム構成を図 13 に示す．

表 7　実装環境

OS	Windows 7 Professional 64bit
CPU	Intel® Core 2 Duo E4500
RAM	2.00GB
言語	ActionScript3.0
IDE	Flash Builder 4
Web ブラウザ	Firefox 4.0
仮想マシン	Flash Player 10.2.153.1

```
//乱数x（BigInteger）と公開パラメータPから暗号文c1=xPを計算
c1:EC_fund = EC_calc.exp(x, P);
//乱数xと公開パラメータsPからxsPの計算
xsP:EC_fund = EC.calc.exp(x, sP);
//ユーザBの識別子ID_b（String）から楕円曲線上の点P_Bを計算
Pb:EC_fund = IDBase.MapToGroup(IDb);
//楕円曲線上の2点，P_B，xsPからペアリングeを計算
e:F3_6_fund = EtatPairing.CalcEtatPairing(Pb, xsP);
//公開ハッシュ関数h2を用いてペアリングeのハッシュ値を求める
e_hash:BigInteger = IDBase.Hash_SHA1_F3_6(e);
//平文m（BigInteger）とハッシュ値を乗算し，暗号文c2を計算
c2:BigInteger = m.multiply(e_hash);
```

図 11　ユーザ A による公開鍵と暗号文 $C=(c_1, c_2)=(xP, m*h_2(e(P_B, xsP)))$の計算

実装した PKG では,インターフェイスとして RFC5408[14]で定められている通り，公開パラメータの HTTP の GET メソッドでの受付，復号鍵リクエストの HTTPS の POST メソッドでの受付，公開パラメータリクエスト，復号鍵の配布，認証の機能を持つ.

GET メソッドと POST メソッドの受付の部分で，REST Style を取り入れ，クライアント側の実装環境に汎用性を与えている．また，復号鍵のリクエスト時の認証方法として，Email Answerback 認証を用いている．一般的に用いられる Basic 認証や Digest 認証などの方法では，サーバにユーザの認証情報が必要となり，事前処理としてユーザ登録が必要になる．Email Answerback 認証では，ユーザに認証用の URL が書かれた Email と Cookie を送信し，ユーザが URL にアクセスした際に Cookie の照合をして認証を行う．これにより，Email アドレス所有者に対する一時的な認証が可能となり，サーバ側でのユーザの認証情報の保管が必要なくなる.

```
//復号鍵sP_Bと暗号文c1を用いてペアリングeを計算
e:F3_6fund = EtatPairing.CalcEtatPairing(sPb, c1);
//公開ハッシュ関数h2を用いてペアリングeのハッシュ値を求める
e_hash:BigInteger = IDBase.Hash_SHA1_F3_6(e);
//c2とハッシュ値を除算し，平文mを計算
m:BigInteger = c2.divide(e_hash);
```

図 12　復号：ユーザ B による $m= c_2/h_2(e(sP_B, c_1))$の計算

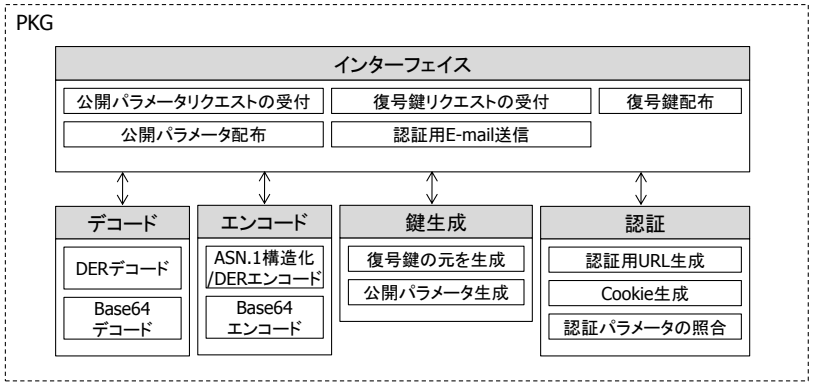


図 13　PKG のシステム構成

実装した PKG は，ASN.1 構造の DER エンコード/デコードや，Base64 エンコード/デコード, 鍵生成, Email Answerback 認証の機能を持つ. なお, RFC5091[15]により，リクエストデータ，公開パラメータ，復号鍵を通信する際の通信データの構造が ASN.1 構造で定められているが，標数 3 の η_Tペアリングでは，有限体の元を表すのに 3 進数を扱う関係上，これらの規定に一部準拠できないことから，元を表す部分のみを独自に設計している.

6.2　PKG へのアクセス手順と動作確認

実装した PKG へのクライアントからの公開パラメータのリクエストと復号鍵のリクエストの手順を以下に示す.

[公開パラメータのリクエスト]

1. 公開パラメータが記述されたファイルの URL に GET メソッドでアクセスする
2. 公開パラメータが記述されたファイルを受け取る

[復号鍵のリクエスト]

1. 復号鍵リクエスト用の HTTPS の URL に POST メソッドを使用し, XML で記述した ID（Email アドレス）を含むリクエスト（RFC5408[14]準拠）を送信する
2. Email アドレス宛に認証用 URL が書かれたメールが届く
3. 認証用 URL にアクセスし，復号鍵が記述されたデータを取得する

実装した PKG に対し，5 章で実装したクライアント側のプログラムでアクセスを行うのは次のようになる. IDBase クラスの request メソッドにより復号鍵リクエストデータを生成（図 14）し, PKG に対し HTTPS の POST メソッドで送信を行う. Email Answerback 認証により Email で送られてきた URL に作成したプログラムからアクセスし，Base64 でエンコードされた復号鍵の情報を受け取り，図 15 のコードによりデコードすることで，復号鍵を取得できることを確認した.

7. おわりに

セキュアな Web アプリケーションを Flash Platform で実現するために，標数 3 の拡大体の元を操作する機能と楕円曲線上の点の操作・演算をする機能を実装し，η_T ペアリングの演算ができるライブラリを開発した. 本ライブラリでは ID ベース暗号の実装に必要な楕円曲線上の点を算出する機能やハッシュ値を算出する機能，リクエストデータ生成機能，Base64 エンコード/デコード機能を持っている.

標数 3 の拡大体の実装では，高速なビット演算アルゴリズムを採用することにより，Web ブラウザ上での高速なペアリング演算を実現した．本ライブラリでは，拡大体の元を表す F3_fund クラスの変数定義部分を変更することで拡大次数と既約多項式を決定するパラメータを設定できるようにした．本ライブラリは，As3Cryprto という外部ライブラリのインポートと，As3Pairing パッケージのインポートを行うことで利用できる.

MOV security を同一にした上で Java により実装された既存のライブラリ jPBC[6] による Tate ペアリングの演算と本ライブラリを用いた η_T ペアリングの演算の速度比較を行い，同程度の演算性能であることを確認した.

本ライブラリを用いた Web アプリケーションの例として，BF 方式の ID ベース暗号を実装し，動作を確認した． PKG の実装では REST スタイルを適用し，認証部分には Email Answerback 認証を採用した．実装した PKG にアクセスして BF 方式の Web クライアントが動作することを確認した.

```
//復号鍵のリクエスト文字列の生成
//アルゴリズム名とID情報を指定してリクエスト文字列を生成
req:String = Request(“BF”, ID);
```

図 14　復号鍵リクエスト文字列の生成コード

```
//PKGから受信したBase64エンコード文字列をデコード
//受信したBase64エンコード文字列recvをデコード
str:String = Base64dec(recv);
```

図 15　PKG から受信した Base64 エンコード文字列をデコードするコード

参考文献

[1] Jean-Luc Beuchat, Nicolas Brisebarre, Jeremie Detrey, Eiji Okamoto,Masaaki Shirase, and Tsuyoshi Takagi, “Algorithms and Arithmetic Operators for Computing the η_T Pairing in Characteristic Three,” IEEE transactions on computers, Vol.57, No.11, November 2008.

[2] Iwan Duursma, Hyang-Sook Lee, “Tate pairing implementation for hyperelliptic curves y2 = xp-x+d,” Advances in Cryptology-ASIACRYPT 2003, LNCS 2894, pp.111-123, Springer-Verlag, 2003.

[3] Paulo S. L. M. Barreto, Steven Galbraith, Colm O hEigeartaigh, Michael Scott, “Efficient Pairing Computation on Supersingular Abelian Varieties,” Cryptology ePrint Archive, Report 2004/375.

[4] Ben Lynn, “PBC Library - Pairing-Based Cryptography (online),” http://crypto.stanford.edu/pbc/ (accessed 2011-01-14).

[5] 光成滋生，“η_T ペアリングの高速な実装(mitsunari@cybozu labs) (online),” http://labs.cybozu.co.jp/blog/mitsunari/2008/03/t_1.html (accessed 2011-05-20).

[6] Angelo De Caro, “Java Pairing-Based Cryptography Library (online),” http://gas.dia.unisa.it/projects/jpbc/ (accessed 2011-01-14).

[7] World Trans System, "システム開発／Flex／AIR (online)," http://www.wts-c.co.jp/flex.html (accessed2011-05-12).

[8] 川原祐人，青木和麻呂，高木剛，“最小の論理命令数での GF(3) 上の加算による η_T ペアリングの高速実装,” 情報処理学会論文誌，Vol.50，No.11，pp.2717-2726(Nov. 2009).

[9] 川原祐人，小林鉄太郎，高橋元，高木剛，“標数 3 の超特異楕円曲線における高速な MapToPoint 写像,” 2009 年暗号と情報セキュリティシンポジウム予稿集（CD-ROM）, 3C3-3, 2009.

[10] Henri Torgemane, “as3crypto -Project Hosting on Google Code- (online),” http://code.google.com/p/as3crypto/ (accessed 2011-01-14).

[11] Robert Granger,Daniel Page,Martijn, Stam,“Hardware and Software Normal Basis

Arithmetic for Pairing-based Cryptography in Characteristic Three," IEEE transactions on computers, Vol.54, No.7, July 2005.
[12] Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone,"Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field," IEEE transactions on computers, Vol.39, No.5, pp.1639-1646, September 1993.
[13] 高木剛, 伊豆哲也, 岡本健, 小林鉄太郎, 境隆一, 高島克幸, 田中秀磨, 花岡悟一郎, "IDベース暗号に関する調査報告書", CRYPTREC ID ベース暗号調査WG, 2009.
[14] G. Appenzeller, L. Martin, M. Schertler, "Identity-Based Encryption Architecture and Supporting Data Structures", RFC 5408, January 2009.
[15] X. Boyen, L. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, December 2007.