15-A 楕円曲線暗号

15-A-1 OpenSSHの認証と鍵交換に使用される楕円曲線暗号

楕円曲線暗号は秘密鍵と楕円曲線上での公開鍵を使用した、暗号化、デジタル署名、鍵交換（鍵共有）等の方式の総称です。

本項では、SSHの通信路暗号化で重要な役割を果たす鍵交換のアルゴリズムである楕円曲線暗号について、使用されている曲線の方程式やパラメータ、そこから共通鍵が生成される仕組みを見ていきます。

以降、本項で出てくる数学用語については「数学用語一覧」（→ p.11）を参照してください。

□ 楕円曲線（elliptic curve）とは

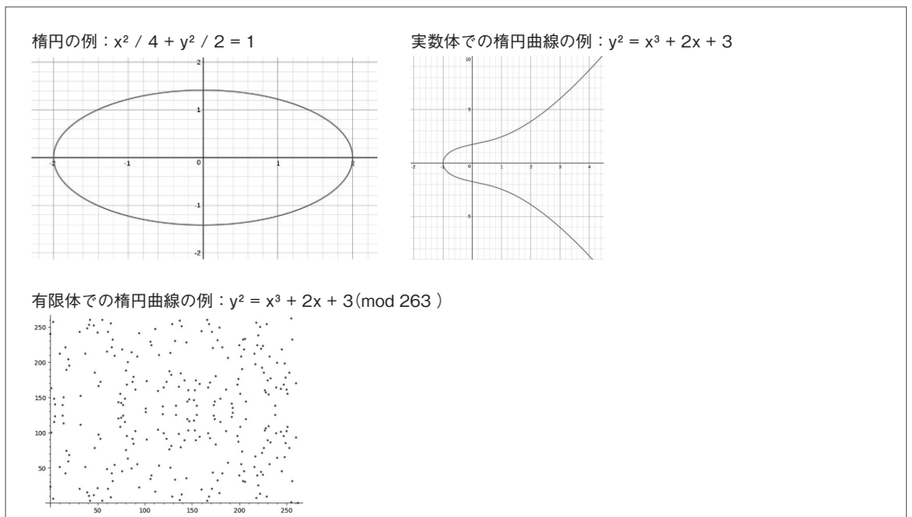
楕円曲線暗号で広く採用されているワイエルシュトラス（Weierstrass）型と呼ばれる楕円曲線は次の方程式で定義されます。

$$y^2 = x^3 + ax + b$$

楕円曲線には、実数体での楕円曲線、複素数体での楕円曲線、有理体での楕円曲線、有限体での楕円曲線があります。以下の式で表されるような楕円とは異なります。

$$x^2 / a^2 + y^2 / b^2 = 1$$

図15-A-1 楕円と楕円曲線



暗号で使用されるのは、有限個の整数を集合の元とする有限体での楕円曲線です。楕円曲線上での剰余演算による点（x, y）の集合なので、xもyも0か正の整数になります。x軸で上下が対称となる実数体での滑らかな楕円曲線とは異なります。

鍵交換のアルゴリズムでは、乱数で生成される秘密鍵（整数）から、適切に選んだ有限体での楕円曲線上の基点となる座標（x1, y1）を秘密鍵の値でスカラー倍算して楕円曲線上の座標（x2,y2）を算出し、これを公開鍵とします。

この公開鍵から離散対数による逆方向の演算で秘密鍵を求める困難さにより安全性が保たれます。

□ 暗号で使われる楕円曲線の式とパラメータ

暗号で使われる楕円曲線にはワイエルシュトラス（Weierstrass）型、モンゴメリー（Montgomery）型、エドワーズ（Edwards）型、ツイストしたエドワーズ型（twistedEdwards）といった種類があり、それぞれに係数やモジュロ（modulo：法）などのパラメータを設定します。どの型を選ぶか、どのようなパラメータを設定するかで計算効率やセキュリティ強度が異なります。

OpenSSHで使われているダニエル・バーンスタイン氏が設計したモンゴメリー型楕円曲線Curve25519の方程式は次のようになります。

$$By^2 = x^3 + Ax^2 + x \pmod p$$

係数A＝486662、係数B＝1、モジュロp＝2^255-19となります（pが素数2^255-19の故、Curve25519という名前になっています）。

その他、基点（base point。部分群の生成元（generator point）となる）、部分群の位数（orderof subgroup）、余因子（cofactor）等の値はRFC7748で規定されています。

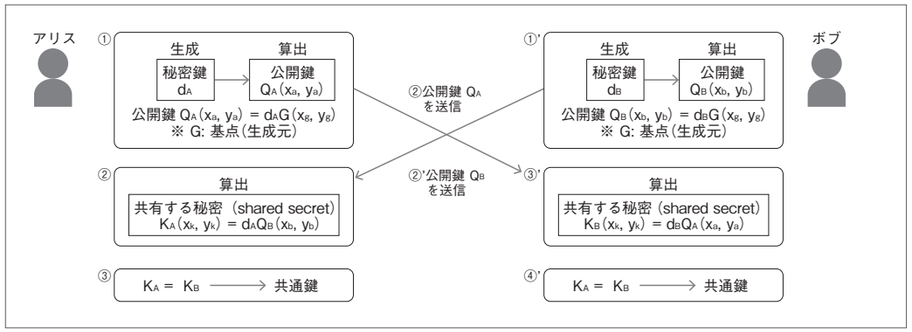
また、NIST（米国標準技術研究所）やSECG（The Standards for Efficient Cryptography Group）では楕円曲線についての推奨パラメータを公開しています。

□ 楕円曲線ディフィー・ヘルマン鍵交換の仕組み

ここでは、楕円曲線ディフィー・ヘルマン鍵交換（Elliptic curve Diffie-Hellman keyexchange：ECDH）による秘密鍵から公開鍵の生成、公開鍵の交換、共有鍵の生成の仕組みを見ていきます。

なお、OpenSSHで採用されているECDHの実装であるCurve25519では、秘密鍵から公開鍵を算出する方法が異なります（詳細は本項末尾の参考文献「Curve25519:new Diffie-Hellman speed records」、「RFC7748:Elliptic Curves for Security」を参照してください）。

図15-A-2　共有鍵を生成する仕組み



①アリスは乱数により秘密鍵dAを生成する

生成した秘密鍵d_A（整数）で基点G（x_g, y_g）をd_A倍して、楕円曲線上の点である公開鍵QAを算出します。

②'ボブは乱数により秘密鍵dBを生成する

生成した秘密鍵d_B（整数）で基点G（x_g, y_g）をd_B倍して、楕円曲線上の点である公開鍵QBを算出します。

②アリスは自身の公開鍵Q_Aをボブに送信する
③ボブは自身の公開鍵Q_Bをアリスに送信する
③アリスはボブから送られてきた公開鍵QBを自身の秘密鍵d_Aによりd_A倍した点K_A（x_k, y_k）のX座標x_k（整数）を「共有する秘密（shared secret）」とする
③'ボブはアリスから送られてきた公開鍵Q_Aを自身の秘密鍵dによりd_B倍した点K_A（x_k, y_k）のX座標x_k（整数）を「共有する秘密（shared secret）」とする
④ボブは「共有する秘密（shared secret）」を共通鍵とする
④'アリスは「共有する秘密（shared secret）」を共通鍵とする

③、③'、④、④'で算出される「共有する秘密（shared secret）」は、$d_AQ_B = d_Ad_BG = d_Bd_AG = d_BQ_A$となり、同じ値となります。

□ 秘密鍵から楕円曲線上で公開鍵を算出する

楕円曲線上で秘密鍵から公開鍵を生成するには、適切に選んだ有限体での楕円曲線上の基点となる座標（x^1, y^1）を秘密鍵の値でスカラー倍算して楕円曲線上の座標（x^2, y^2）を算出し、これを公開鍵とします。

例えば、Curve25519では以下のようなモンゴメリー型楕円曲線を使用しています。

$y^2 = x^3 + 486662\,x^2 + x \pmod{2^{255} - 19}$

わかりやすくするために、ここではパラメータを以下のような小さな値にしたワイエルシュトラス型楕円曲線上で秘密鍵から公開鍵を算出するプロセスをフリーな数学ソフトウェアSageMathを使って見ていきます（SageMathのインストール方法は後述します）。

・**楕円曲線**：$y^2 = x^3 + 2x + 3 \pmod{263}$ → 群（G）の位数（order of group）：270
・**パラメータ**
　基点：（126, 76）→ 生成される部分群（H）の位数：6
　余因子（cofactor）：（群Gの位数）/（部分群Hの位数）= 270 / 6 = 45
　単位元：（0, 1）
・**剰余演算（mod 263）の定義された有限体（F）の元**：0, 1, 2, … , 262
・**曲線の定義された群（C）の元**：（0, 1）（, 0, 23）, …（, 126, 76）, …（, 126, 187）, …,
（144, 35）, …（, 144, 228）, …（, 262, 0）
・**基点（126, 76）から生成される部分群（H）の元**：（0, 1）（, 126, 76）（, 144, 35）
（, 262, 0）,（144, 228）（, 126, 187）

SageMath(コマンド名はsage) を起動して。楕円曲線を定義し確認

```
[...]$ ./sage  ←現在のディレクトリ下にあるsageを起動
…
sage: F = FiniteField(263)
↑剰余演算（mod 263）の定義された有限体F（FiniteField）を生成
sage: C = EllipticCurve(F, [ 2, 3 ])
↑有限体F上で楕円曲線C（Curve）y2 = x3 + 2x + 3 を定義
sage: C  ←楕円曲線Cの定義を表示。「print(C) 」と同じ
Elliptic Curve defined by y^2 = x^3 + 2*x + 3 over Finite Field of size 263
sage: F  ←有限体Fの定義を表示。「print(F) 」と同じ
Finite Field of size 263
sage: F.order()  ←有限体Fの位数を表示
263
sage: F.cardinality()  ←有限体Fの濃度（位数と同じ）を表示
263
sage: C.order()  ←楕円曲線C（群C）の位数を表示
270
sage: C.cardinality()  ←楕円曲線C（群C）の濃度（位数と同じ）を表示
270
sage: C.points()  ←群Cの270個の元を全て表示。「print(C.points())」と同じ
                    （sageでは3次元（X：Y：Z）の座標として表示される。Zは1か0の値を取り、
                      1は楕円曲線上の点、0は無限遠点（単位元）を表す）
[(0 : 1 : 0), (0 : 23 : 1), (0 : 240 : 1), (1 : 100 : 1), (1 : 163 : 1),
(3 : 6 : 1), (3 : 257 : 1), (4 : 115 : 1), (4 : 148 : 1), (5 : 123 : 1),
…（途中省略）…
(119 : 126 : 1), (119 : 137 : 1), (120 : 99 : 1), (120 : 164 : 1),
(123 : 91 : 1), (123 : 172 : 1), (126 : 73 :1), (126 : 187 :1),
…（途中省略）…
(142 : 89 : 1), (142 : 174 : 1), (144 : 35 : 1), (144 : 228 : 1),
(145 : 119 : 1), (145 : 144 : 1), (146 : 103 : 1), (146 : 160 : 1),
…（途中省略）…
(253 : 185 : 1), (255 : 1 : 1), (255 : 262 : 1), (256 : 31 : 1),
(256 : 232 : 1), (260 : 93 : 1), (260 : 170 : 1), (262 : 0 :1)]
sage:
```

濃度については「数学用語一覧」（→ p.10）を、無限遠点（曲線上にない点）については「楕円曲線上の加法」（→ p.7）を参照してください。

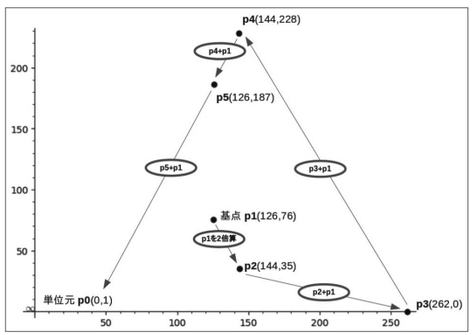
上記の「C.points()」で表示される群Cの270個の元のグラフが前述の「楕円と楕円曲線」のなかの「有限体での楕円曲線の例」となります。そのうち、網掛けの6個の元が以下で定義する部分群Hの元となります。

以下はわかりやすさのため、小さい位数6の元（126, 76）を基点に使用して、楕円曲線上の座標を計算します。

基点を決めて、スカラー倍算により楕円曲線上の座標を計算

```
sage: H = C.point((126, 76))  ←（126, 76）を基点にした部分群Hを定義。以降、これを使う
sage: H.order()
6
sage: [H*0, H*1, H*2, H*3, H*4, H*5]
[(0 : 1 : 0),  ←単位元、P0とする
(126 : 76 : 1),  ←基点、P1とする。この点が部分群Hの生成元（generator）となる
(144 : 35 : 1),  ←基点*2、P2とする。基点P1のスカラー２倍算により算出
(262 : 0 : 1),  ←基点*3、P3とする。「P2 ＋ 基点P1」による加算により算出
(144 : 228 : 1),  ←基点*4、P4とする。「P3 ＋ 基点P1」による加算により算出
(126 : 187 : 1)]  ←基点*5、P5とする。「P4 ＋ 基点P1」による加算により算出
                    「P5 ＋ 基点P1」はP0（単位元）に戻る
                    部分群Hは基点P1を生成元とする巡回群となる
sage: pts = [H*0, H*1, H*2, H*3, H*4, H*5]
sage: print(pts)
[(0 : 1 : 0), (126 : 76 : 1), (144 : 35 : 1), (262 : 0 : 1),
(144 : 228 : 1), (126 : 187 : 1)]
sage: sum([ plot(p) for p in pts ])
↑格納されている配列ptsに格納されている6個全ての元を画像ビューワに表示
Launched png viewer for Graphics object consisting of 6 graphics
primitives
```

図15-A-3　**部分群Hの全ての元（6個）を画像ビューワーにプロット**



座標上の点の2倍算と加法の計算式については、後述の「楕円曲線上の加法」（→ p.7）を参照してください。

秘密鍵d（整数値）と楕円曲線上の点P（基点）が与えられた時、公開鍵は点Pをd倍した「P * d」となります。

モンゴメリー型楕円曲線Curve25519では秘密鍵は乱数により32バイト（256ビット）の整数として生成し、方程式と基点は以下のようになります。算出する公開鍵も32バイト（256ビット）の整数です。

・**方程式**：$y^2 = x^3 + 486662\,x^2 + x \pmod{2^{255} - 19}$
・**基点**：X（P）9
　　Y（P）14781619447589544791020593568409986887264606134616475288964881837755586237401

ここでは上記のSageの実行例で定義したワイエルシュトラス型楕円曲線と基点、および基点から生成された部分群Hを使うことにします。

・**方程式**：$y^2 = x^3 + 2x + 3 \pmod{263}$
・**基点**：X（P）126
　　Y（P）76

例えば、アリスの秘密鍵d_Aの値を4、基点Pを（126, 76）とすると、アリスの公開鍵Q_Aは、

$$Q_A = P * d_A = P * 4 = (144, 228) = P4$$

となります。

例えば、ボブの秘密鍵d_Bの値を5、基点Pを（126, 76）とすると、ボブの公開鍵Q_Bは、

$$Q_B = P * d_B = P * 5 = (126, 187) = P5$$

となります。

アリスはボブの公開鍵と自身の秘密鍵により、共通鍵「$d_A * Q_B$」を計算します。これは、

$$d_A * Q_B = 4 * (P * 5) = (144, 35) = P2$$

となります。

ボブはアリスの公開鍵と自身の秘密鍵により、共通鍵「$d_B * Q_A$」を計算します。これは、

$$d_B * Q_A = 5 * (P * 4) = (144, 35) = P2$$

となります。

アリスとボブの共通鍵を算出

```
sage: F = FiniteField(263)
↑剰余演算（mod 263）の定義された有限体F（FiniteField）を生成
sage: C = EllipticCurve(F, [ 2, 3 ])
↑有限体F上で楕円曲線C（Curve）y2 = x3 + 2x + 3 を定義
sage: H = C.point((126, 76))  ←基点：P1
sage: P0 = H*0  ← (0 : 1 : 0) 単位元
sage: P1 = H*1  ← (126 : 76 : 1) 基点（生成元）
sage: P2 = H*2  ← (144 : 35 : 1)
sage: P3 = H*3  ← (262 : 0 : 1)
sage: P4 = H*4  ← (144 : 228 : 1)
sage: P5 = H*5  ← (126 : 187 : 1)
sage: pts=[P0, P1, P2, P3, P4, P5]
sage: print(pts)
```

```
[(0 : 1 : 0), (126 : 76 : 1), (144 : 35 : 1), (262 : 0 : 1),
(144 : 228 : 1), (126 : 187 : 1)]

sage: dA=4  ←アリスの秘密鍵：4
sage: QA=P1*dA  ←アリスの公開鍵：P4
(144 : 228 : 1)

sage: dB=5  ←ボブの秘密鍵：5
sage: QB=P1*dB  ←ボブの公開鍵：P5
(126 : 187 : 1)

sage: KA=dA*QB  ←アリスの共通鍵を計算
sage: KA  ←P2（ボブの共通鍵と同じ）
(144 : 35 : 1)

sage: KB=dB*QA  ←ボブの共通鍵を計算
sage: KB  ←P2（アリスの共通鍵と同じ）
(144 : 35 : 1)
```

これまでの実行例ではSageMathを使用したため、内部的な演算処理は表示されませんでした。
以下に楕円曲線上の2点（2個の元）の足し算の式を示し、次に数学ソフトウェアGeniusを使って、実際に部分群Hの点を計算してみます。Geniusのインストール方法は後述します。

□ 楕円曲線上の加法

楕円曲線（$y^2 = x^3 + ax + b$）上の2点P（x_p, y_p）とQ（x_q, y_q）の和P + Qは、PとQを通る直線が楕円曲線と交わる他の点R'（x_r', y_r'）を計算し、$x_r = x_r'$、$y_r = -y_r'$として、R'のX軸に関して対称な点R（x_r, y_r）となります。

Pを2倍（P + P）する場合は、Pの接線が楕円曲線と交わる点R'となります。

以下のようにしてR（x_r, y_r）を求めます。

①2点PとQを通る直線の傾きλを計算する。同じ点Pを2倍する時（P + P）は点Pの接線の傾きλを計算する

- **点PとQを通る直線の傾きλの計算**：（式1）$\lambda = (y_p - y_q) / (x_p - x_q)$
- **点Pの接線の傾きλの計算**：（式2）$\lambda = (3x_p^2 + a) / (2y_p)$

②上の①で算出したλを使って、P を2倍（P + P）した点R、あるいはP と Qを足した点RのX座標x_rとY座標y_rを計算する

- **X座標の計算**：（式3）$x_r = \lambda^2 - x_p - x_q$
  （式3'）P + Pの場合は$x_r = \lambda^2 - x_p - x_p$
- **Y座標の計算**：（式4）$y_r = \lambda (x_p - x_r) - y_p$

点P（x, y）のX軸に関して対称な点-P（x, -y）をPの「**逆元**」と定義します。

点Pと点-Pの加算はこの2点を通る直線が楕円曲線上で交わる点ですが、このY軸に並行な垂直線は楕円曲線と他で交わることがありません。これを無限遠で交わる点として無限遠点O（オー）と定義します。この無限遠点O（0, 1）を「**単位元**」と定義します。

以下はGeniusを使って上記の計算式により基点p1（126, 76）を2倍（p1 + p1）した点p2、3倍（p2 + p1）した点p3を求めます。

Geniusを起動し、計算式から基点のスカラー倍算を行う

```
[...]$ genius
…（途中省略）…
genius> p = 263
= 263
genius> a=2
= 2
genius> x1=126
= 126
genius> y1=76
= 76

（ここからp2（x2, y2）の計算。P2 = P1 * 2を計算する）
genius> lamda=(3*x1^2+a)*(2*y1)^-1 mod p   ←❶
= 213
genius> x2=lamda^2-x1-x1 mod p  ←（式3'）により計算
= 144
genius> y2=lamda*(x1-x2)-y1 mod p  ←（式4）により計算。P2 = (144, 35)
= 35

（ここからp3（x3, y3）の計算。P3 = P2 + P1を計算する）
genius> lamda=(y2-y1)*(x2-x1)^-1 mod p  ←❷
= 100
genius> x3=lamda^2-x1-x2 mod p  ←（式3）により計算
= 262
genius> y3=lamda*(x1-x3)-y1 mod p  ←（式4）により計算。p3 = （262, 0）
= 0
```

❶「楕円曲線上の加法」の(式2)により計算
（3*x1^2+a）/（2*y1）は（2*y1）の逆元（2*y1）^-1の乗算により計算
❷（式1）により計算
（y2-y1）/（x2-x1）は（x2-x1）の逆元（x2-x1）^-1の乗算により計算

□ SageMathのインストール方法

SageMathはPythonで書かれた数学ソフトウェアです。

> **SageMath(公式サイト)**
> https://www.sagemath.org/

> **SageMath(ドキュメント)**
> https://www.johannes-bauer.com/compsci/ecc/

公式サイトにソースコードとUbuntuのバイナリはありますが、rpmパッケージは提供されていません。しかし、DockerHubからUbuntuベースのコンテナが提供されているので、これを使うことができます。

SageMathのインストール

```
[...]# podman pull sagemath/sagemath
[...]# podman images
REPOSITORY                      TAG      IMAGE ID       CREATED        SIZE
docker.io/sagemath/sagemath     latest   b08484514181   6 months ago   2.3 GB
[...]# podman run -it --name sagemath -h sagemath sagemath bash
sage@sagemath
:~$ sage
 ┌────────────────────────────────────────────────────┐
 | SageMath version 9.1, Release Date: 2020-05-20     |
 | Using Python 3.7.3. Type "help()" for help.        |
 └────────────────────────────────────────────────────┘
sage:
sage: exit   ←exitコマンドで終了
```

□ Geniusのインストール方法

Fedoraプロジェクトのサイトから Fedora 28用のrpmパッケージが提供されているので、これを使用します。

> **Genius(公式サイト)**
> **https://www.jirka.org/genius.html**

Geniusのインストール

```
[...]# wget https://kojipkgs.fedoraproject.org//vol/fedora_koji_archive02/packages/
genius/1.0.24/1.fc28/x86_64/genius-1.0.24-1.fc28.x86_64.rpm
[...]# rpm -ivh genius-1.0.24-1.fc28.x86_64.rpm
[...]# genius
genius>
genius> exit   ←exitコマンドで終了
```


□ 数学用語一覧

楕円曲線暗号のドキュメントには抽象代数学、集合論、整数論などの分野の数学用語(日本語、英語)が出てきます。これらについて、参考のために以下の表で簡単な説明を加えました。

表15-A-1　数学用語一覧

用語	用語(英語)	説明
体	field	次の条件を満たす集合を体と呼ぶ ①加法について、交換法則 a･b=b･a を満す群になっている ②乗法について、交換法則 a･b=b･a を満す群になっている ③加法と乗法について分配法則が成り立つ (a・b) c=ac・bc, a (b・c) =ab・ac ※群には演算が一種類だけ与えられている(演算は加法でも乗法でもよい)。体には、加法と乗法という二種類の演算が入っている ※加法の逆演算は減法，乗法の逆演算は除法なので、体は四則演算が可能な集合のこと。実数全体の集合は体(実数体)、有理数全体の集合は体(有理数体)となる。整数は乗法に関する逆元が存在しない(1 / 2 は整数ではない)ので、体にならない
群	group	次の条件を満たす集合を群と呼ぶ。 ①集合 (G) の要素の間に演算が成り立ち、その演算に関して集合は閉じている ②演算に対して結合則が成り立つ a・(b・c)=(a・b)・c ③演算には 単位元が存在する ④演算には 逆元が存在する
部分群	subgroup	集合Gが二項演算・に関して群であり、Gの部分集合Hが演算・に関して群である時、HはGの部分群であるという
巡回群	cyclic group	1つの元から生成される群を巡回群という。巡回群Gがａによって生成される時、ａをＧ の生成元 (generator) という 群がただ1つの元 aで生成される時、その群のどの元もaの整数冪(べき)として、あるいはaの整数倍として表される
閉性(閉じている)	closed	集合Gの元a、bに対して演算・を施した結果が再び元の集合Gに属すること。自然数全体の成す集合は、加法について閉じているが減法については負数になる場合があるので閉じていない。整数全体の成す集合は、乗法について閉じているが除法については小数になる場合があるので閉じていない
二項演算	binary operation	数の四則演算 (加減乗除) 等の「2つの数から新たな数を決定する規則」を一般化した概念
単位元	identity element	集合Gとその上の二項演算・について、元eがGの全ての元aに対してa・e = e・a = a を満たす時、eをGの単位元という
逆元	inverse element	集合Gとその上の二項演算・について、eを単位元とした時、x・y = y・x = e を満すy はxの逆元であるといい、またxはyの逆元になる
生成元	generator	１つの元から生成される群を巡回群という。巡回群Ｇがａによって生成される時、ａを Ｇの生成元 (generator) という
有限体	finite field	有限個の元からなる体、すなわち四則演算が定義され閉じている有限集合のこと
位数	order	集合、群、環、体の位数：群、環、体の元の数を位数 (order) と呼ぶ。濃度 (cardinality) あるいはサイズ (size) ともいう 元の位数：有限群の場合はどの元 aも必ず、あるkが存在して、(a) ^k=eとなり、この kのうち最小のものを元ａの位数という
集合の濃度	cardinality	濃度は無限集合の大きさを測るために導入された概念。実数全体の集合の大きさ (濃度) は有理数全体の大きさ (濃度) より大きい。整数全体の濃度と自然数全体の濃度は等しい
素数	prime number	約数が1と自分自身のみの、1より大きい自然数。例) 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, …
剰余演算	modulo operation	2つの正の整数の、ある数値を別の数値 (法と呼ばれる) で除算し、余りを取得する演算

法	modulo	剰余演算における除数
余因子	cofactor	楕円曲線暗号の場合、楕円曲線の群の位数がn、部分群の位数がrの時、「cofactor」hは、h=n/rとなる。Cofactorが1の時は部分群は群と等しい
スカラー倍算	scalar multiplication	楕円曲線上の点Pのk倍点kPを計算すること

□ 参考文献

以下に、参考文献を示します。

「楕円曲線暗号入門（2013年度）」伊豆哲也
https://researchmap.jp/izu_tetsuya/資料公開

「楕円曲線ディフィー・ヘルマン鍵共有」ウィキペディア
https://ja.wikipedia.org/wiki/楕円曲線ﾃﾞｨﾌｨｰ･ﾍﾙﾏﾝ鍵共有

「Curve25519」ウィキペディア
https://en.wikipedia.org/wiki/Curve25519

「Curve25519：new Diffie-Hellman speed records」ダニエル・バーンスタイン
https://cr.yp.to/ecdh/curve25519-20060209.pdf

「RFC7748：Elliptic Curves for Security」
https://www.ietf.org/rfc/rfc7748.txt

「RFC5656：Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer」
https://tools.ietf.org/html/rfc5656