GNU MPを用いた楕円曲線暗号の実装

Implementation of Elliptic Curve Cryptography
using GNU Multiple Precision Arithmetic Library

奥秋 清次、小笠原　裕[1]、中野　竜也[2]
OKUAKI Seiji,OGASAWARA Yutaka,NAKANO Tatsuya

1　はじめに

遠隔地にある端末から通信回線を用いて、機器を操作、監視する遠隔操作・監視システムがある。このシステムには、プラントの操作と監視、建物に進入する不審者の監視、分散した工場の状態監視などがある。今後、自宅の家電製品の操作、監視なども普及すると考えられる。従来、遠隔操作・監視システムは、通信回線に専用線などを用いていたが、安価なインターネットを用いることが検討されている [7]。

通信回線にインターネットを用いると、安価で遠隔操作・監視システムが実現できるだけでなく、遠隔地の端末を固定せず、インターネットに接続できる任意の場所から操作、監視が可能となる。しかしながら、インターネットは不特定多数のユーザが多様な用途で使用するため、悪意のある利用者が存在し、データを盗聴、データを改ざんされる場合がある。また、インターネットは専用線と異なり、通信速度はそのときの利用率により変動する。

悪意のある利用者を排除するためには個人確認法、データの改ざんを防ぐためには、電子署名、盗聴を防ぐためにはデータの暗号化が有効である。また、通信速度を保障するためにはQoS（Quality of Service）が有効である。

本大学校には卒業研究に相当する開発課題がある。この開発課題において、遠隔地の端末からインターネットを用いて、機器を操作、監視する実験システムを構築した。本稿ではその課題の中で作成した、楕円曲線暗号と楕円曲線を用いた電子署名の実装について述べる。楕円曲線暗号は、多倍長の整数演算を可能にするC言語のライブラリ GNU Multiple Precision Arithmetic Library（GNU MP）[6] を用いて実装する。実装した楕円曲線暗号は、Massey-Omura Encryption、楕円曲線を用いた電子署名は、ECDSA（Elliptic Curve Digital Signature Algorithm）である。

[1]株式会社オービックビジネスソリューション
[2]富士通サポートアンドサービス株式会社

2　準備

楕円曲線暗号は、従来のRSAなどに比べて短い鍵長で同等の安全性が得られる。たとえば、RSAの鍵長1024ビット、4096ビットに対して楕円曲線暗号はそれぞれ173ビット、313ビットで同等である [1]。したがって、楕円曲線暗号は計算能力の低い、限られた記憶容量の装置でも実装が可能である。有限体 $\mathbb{F}_p$ 上の楕円曲線 E とは、p を素数、$a, b \in \mathbb{F}_p$ とし、次式で表される。

$$E : y^2 = x^3 + ax + b \pmod{p}$$

ただし、暗号に用いる楕円曲線は、$p \geq 5$、$4a^3 + 27b^2 \neq 0$ を満たす必要がある。

2.1　楕円曲線上の演算

有限体 $\mathbb{F}_p$ 上の有理点の集合は、次に定義する演算により可換群をなす。この有理点の集合を $E(\mathbb{F}_p)$ で示す。ここで、無限遠点 $\mathcal{O} = (\infty, \infty)$ は単位元である。

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p^2 | y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

定義 1. P と Q を E 上の異なる有理点とする。P と Q を結ぶ直線を l とし、l と E の交

わる点を R' とする。点 R' について x 軸と対称な点を $R = P + Q$ とする。$P = Q$ の場合、直線 l は P における接線とする。

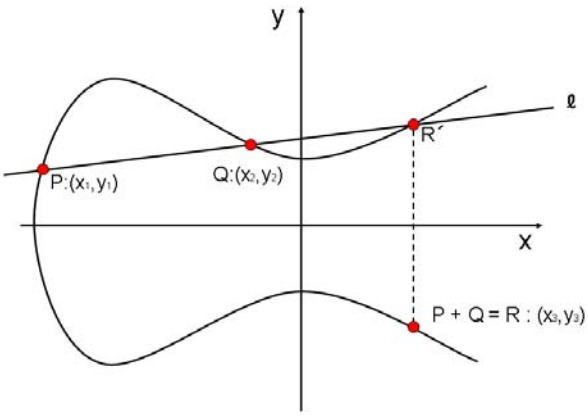

図 1: 楕円曲線上の演算

楕円曲線 E 上の点 $P = (x_1, y_1)$、$Q = (x_2, y_2)$ とした場合、定義 1 の演算により求められる点 $R = P + Q = (x_3, y_3)$ は、次式で与えられる。

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$$

無限遠点 $\mathcal{O}$ は単位元であるので、$P + \mathcal{O} = \mathcal{O} + P = P$ である。P の逆元 $-P$ は、$-P = (x_1, -y_1)$ である。集合 $E(\mathbb{F}_p)$ の個数、すなわち位数 $\#E(\mathbb{F}_p)$ は、Hasse の定理 [1, 2, 3, 4] より

$$p + 1 - 2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}$$

である。暗号では $\#E(\mathbb{F}_p) = fN$ とした場合、f が小さな整数、かつ N が大きな素数となる楕円曲線を用いる。

この群ついて、$P \in E(\mathbb{F}_p)$、$x \in \mathbf{Z}$ に対して、(xP, P) から x を求めることを楕円離散対数問題と呼び、解くことが非常に難しいと考えられている。現在、x のサイズが 160 ビット以上あれば現実的な時間内に解けないと信じられている。楕円離散対数問題を解く手法に、Baby step Giant step、ρ 法、λ 法がある。これらの手法は、x のサイズが小さいと x を求めることができる。

2.2 Massey-Omura 暗号

Massey-Omura 暗号 [2] は、交信回数が 3 回必要であるが、公開鍵も秘密鍵も必要としない暗号である。Alice が Bob にメッセージ m を送る場合、Massey-Omura 暗号は次の手順で暗号化と復号化を行う。

1. Alice と Bob は、$E(\mathbb{F}_p)$ と N を一致させる。
2. Alice のメッセージを $M \in E(\mathbb{F}_p)$ とする。
3. Alice は N と互いに素なランダムな整数 $k_A \in \mathbf{Z}_N$ を選ぶ。すなわち $gcd(k_A, N) = 1$ である。$M_1 = k_A M$ を計算し、Bob に M_1 を送る。
4. Bob は $gcd(k_B, N) = 1$ であるランダムな整数 $k_B \in \mathbf{Z}_N$ を選び、$M_2 = k_B M_1$ を計算し、Alice に M_2 を送る。
5. Alice は $k_A^{-1} \in \mathbf{Z}_N$ 求め、$M_3 = k_A^{-1} M_2$ を計算し、Bob に M_3 を送る。
6. Bob は $k_B^{-1} \in \mathbf{Z}_N$ 求め、$M_4 = k_B^{-1} M_3$ を計算する。このとき $M_4 = M$ である。

M_4 は次の計算により M に復号できる。

$$M_4 = k_B^{-1} k_A^{-1} k_B k_A M = M$$

2.3 Elliptic Curve Digital Signature Algorithm

米国の標準である電子署名 Digital Signature Algorithm（DSA）は、有限体上の乗法群を用いている。DSA を楕円曲線上の群に置き換えた方式が、Elliptic Curve Digital

Signature Algorithm（ECDSA）[2] である。ECDSA の手順は次のとおりである。

Alice は Bob にメッセージ m に署名付けて送信する。Alice は次の準備をする。

1. 有限体 $\mathbb{F}_p$、楕円曲線上の有理点の集合 $E(\mathbb{F}_p)$、$E(\mathbb{F}_p)$ の位数 $\#E(\mathbb{F}_p) = fN$ を選択する。
2. 位数 N である生成元 $G \in E(\mathbb{F}_p)$ を選択する。
3. 秘密鍵 $a \in \mathbf{Z}_N$ を選択し、$Q = aG$ を計算する。
4. $\mathbb{F}_p$、E、N、G、Q を公開する。

Alice は次の手順で、メッセージ m の署名 s を作成する。

1. 整数 k（$1 \leq k < N$）を選択し、$R = kG = (x, y)$ を計算する。
2. $s = k^{-1}(m + ax) \pmod N$ を計算する。

Alice は署名文 (m, R, s) を Bob に送る。Bob は次の手順で署名を検証する。

1. $u_1 = s^{-1}m \pmod N$ と $u_2 = s^{-1}x \pmod N$ を計算する。
2. $V = u_1G + u_2Q$ を計算する。
3. $V = R$ ならば署名は正しい。そうでなければ署名は正しくない。

署名が正しければ、次の計算により $V = R$ となる。

$$\begin{aligned} V &= u_1G + u_2Q = s^{-1}mG + s^{-1}xQ \\ &= s^{-1}(mG + xaG) = kG = R \end{aligned}$$

2.4 実装に必要なアルゴリズム

2.4.1 2 進展開法

楕円曲線暗号の実行時間では、点 $P \in E(\mathbb{F}_p)$ の α 倍の計算が主となる。αP の計算を高速化するためのアルゴリズムが 2 進展開法 [1, 3] である。2 進展開法は、整数 α を

$$\alpha = \sum_{i=0}^{l-1} \alpha_i 2^i$$

として展開する。ここで、l はビット長 $l = \lceil \log_2 \alpha \rceil$、$\alpha_i \in \{0, 1\}$ である。

アルゴリズム 1　$Binary(\alpha, P)$

$Q \leftarrow \mathcal{O}$
for $i \leftarrow l - 1$ **downto** 0
do $Q \leftarrow 2Q$
　if $\alpha_i = 1$
　　then $Q \leftarrow Q + P$
return Q

2.4.2 メッセージと楕円曲線上の点との対応

メッセージ m は、整数で表される。この整数 m を楕円曲線上の点 M に対応付ける必要がある。Koblitz が提案した方法 [2] は次のとおりである。

楕円曲線 $y^2 = x^3 + ax + b \pmod p$ が与えられると、m を $0 \leq m < \frac{p}{100}$ となるようにとる。

アルゴリズム 2　$Msg\_to\_Pnt(M, m)$

1. $0 \leq i < 100$ に対して $x_j = 100m + i$ を計算する。
2. $0 \leq i < 100$ に対して $s_i = x_i^3 + ax_i + b$ を計算する。
3. もし、$s_i^{\frac{p-1}{2}} \equiv 1 \pmod p$ ならば、$y_i \equiv \sqrt{s_i} \pmod p$ とし、(x_i, y_i) を楕円曲線上の点 M とする。

楕円曲線上の点 $M = (x_i, y_i)$ からメッセージ m を得る場合、次式で計算する。

$$m = \left\lfloor \frac{x_i}{100} \right\rfloor$$

2.4.3 p を法とした平方根

合同式

$$x^2 \equiv a \pmod{p}$$

が解を持つ場合、a を p での平方剰余、そうでない場合を平方非剰余とよぶ。平方剰余であるか、平方非剰余であるかは Legendre 記号

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \pmod{p}$$

により、

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{平方剰余} \\ -1 & \text{平方非剰余} \\ 0 & a \text{ が } p \text{ の倍数} \end{cases}$$

である。$Sqrt\_Mod(a, p)$ は p を法とした a の平方根を求めるアルゴリズム [5] である。

アルゴリズム 3 $Sqrt\_Mod(a, p)$

1. p を法とした平方非剰余 g を求める。$g = 2, 3, 4, \ldots$ と順に平方剰余記号を計算して探す。
2. $p - 1 = 2^r q$、q は奇数となる r、q を計算する。
3. $h \leftarrow g^q \pmod{p}$ と $b \leftarrow a^q \pmod{p}$ を計算する。
4. 6.〜12. により $b^{-1} \equiv s^2 \pmod{p}$ となる s を計算する。
5. $a^{\frac{q+1}{2}} s \pmod{p}$ を出力して終了。
6. $s \leftarrow 1$、$h \leftarrow h^{-1} \pmod{p}$
7. $i = r - 2$ から 0 まで 8.〜11. を実行する。
8. $b^{2^i} \pmod{p} = -1$ ならば 9.、10. を実行。
9. $s \leftarrow sh \pmod{p}$
10. $b \leftarrow bh^2 \pmod{p}$
11. $h \leftarrow h^2 \pmod{p}$
12. s を出力する。

3 実装

楕円曲線暗号は、多倍長の整数演算を可能にする C 言語のライブラリ GNU MP を用いて実装する。GNU MP には多倍長の整数での四則演算、剰余演算、べき乗演算、p を法とした逆数演算、素数判定がある。しかし、p を法とした平方根を求める関数はないので自作する。

3.1 実装の手順

Massey-Omura 暗号と ECDSA を次の手順で実装する。

1. 楕円曲線の選択
2. $P + Q$ を計算する関数の作成
3. 2 進展開法の作成
4. p を法とした平方根を求める関数の作成
5. メッセージと楕円曲線上の点との対応付けをする関数の作成
6. Massey-Omura 暗号の実装
7. ECDSA の実装

3.1.1 楕円曲線の選択

楕円曲線暗号は、鍵長 160 ビット以上で安全であるとされている。鍵長を 160 ビット以上とするためには、$N > 2^{160}$ となる楕円曲線を選択する必要がある。今回、文献 [1] の楕円曲線を利用して実装する。

Elliptic Curve parameters:

$$y^2 = x^3 + ax + b \pmod{p}$$

$$a = 10$$

$$b = 1348462411414361312611054113\backslash$$
$$16931087580694918677422294274$$

$$
\begin{aligned}
p &= 2^{190} + 129 \\
&= 1569275433846670190958947355\diagdown \\
&\quad 8019166040255888611160086283 53 \\
\#E(\mathbb{F}_p) &= 1569275433846670190958947355\diagdown \\
&\quad 7 \\
&\quad 8028704030525554089694699788 3 \\
&\quad \text{(素数)}
\end{aligned}
$$

無限大 ∞ は、p より大きい値を割り当てる。

$$
\begin{aligned}
\mathcal{O} &= (\infty, \infty) \\
\infty &= 9999999999999999999999999999\diagdown \\
&\quad 9999999999999999999999999999 99
\end{aligned}
$$

実装において楕円曲線 E のパラメータ a、b、p、$\#E(\mathbb{F}_p)$ と無限遠点 $\mathcal{O}$、無限大 ∞ は、大域変数として扱う。この楕円曲線は位数 $\#E(\mathbb{F}_p)$ が素数なので、$f = 1$ であり、$N = \#E(\mathbb{F}_p)$ である。

3.1.2 楕円曲線上の演算

楕円曲線上の有理点を $P = (x_1, y_1)$、$Q = (x_2, y_2)$、$R = (x_3, y_3)$ とする。$Point\_Add(R, P, Q)$ は、P と Q が与えられると、$R = P + Q$ を計算する。

アルゴリズム 4 $Point\_Add(R, P, Q)$

if $x_1 = \infty$ **and** $y_1 = \infty$
  then $R \leftarrow (x_2, y_2)$
    return
if $x_2 = \infty$ **and** $y_2 = \infty$
  then $R \leftarrow (x_1, y_1)$
    return
if $x_1 = x_2$
  then if $y_1 \neq y_2$
    then $R \leftarrow (\infty, \infty)$
      return
    else $\lambda \leftarrow \frac{3x_1^2 + a}{2y_1}$
  else $\lambda \leftarrow \frac{y_2 - y_1}{x_2 - x_1}$
$x_3 \leftarrow \lambda^2 - x_1 - x_2$
$y_3 \leftarrow \lambda(x_1 - x_3) - y_1$
$R \leftarrow (x_3, y_3)$
return

3.1.3 2 進展開法

α を多倍長とした場合の 2 進展開法を実装する。$Bit\_Length(\alpha)$ は、α のビット長 $\lceil \log_2 \alpha \rceil$ を計算する。

アルゴリズム 5 $Bit\_Length(\alpha)$

$len \leftarrow 0$
$hw \leftarrow mpz\_popcount(\alpha)$
if $hw = 0$
  then return 0
$len \leftarrow mpz\_scan1(\alpha, 0)$
for $i \leftarrow 1$ **to** $hw - 1$
  do $len \leftarrow mpz\_scan1(\alpha, len + 1)$
return $len + 1$

$Bit\_Length(\alpha)$ で用いた $mpz\_popcount(\alpha)$ と $mpz\_scan1(\alpha, st)$ は、GNU MP の関数である。$mpz\_popcount(\alpha)$ は、α のビット'1' の個数、すなわちハミング重みを計算する。$mpz\_scan1(\alpha, st)$ は、α についてビット位置 st から MSB(Most Significant Bit) に向けてビット'1' を検索し、検索したビット'1' の位置を返す。$Binary(Q, \alpha, P)$ は、$Q = \alpha P$ を計算する関数である。

アルゴリズム 6 $Binary(Q, \alpha, P)$

$Q \leftarrow (\infty, \infty)$
$l \leftarrow Bit\_Length(\alpha)$
for $i \leftarrow l - 1$ **downto** 0
  do $Point\_Add(R, Q, Q)$
    $Q \leftarrow R$
    if $mpz\_tstbit(\alpha, i) = 1$
      then $Point\_Add(R, Q, P)$
        $Q \leftarrow R$
return

$mpz\_tstbit(\alpha, i)$ は、GNU MP の関数であり、α のビット位置 i の 1 ビットを返す。

3.2 p を法とした平方根

$Sqrt\_Mod(result, a, p)$ は、p を法とした a の平方根を求める関数である。$Get\_QNP(qnr, p)$ は、p を法とした平方非剰余 qnr を求める関数である。$mpz\_legendre(q, p)$ と $mpz\_invert(h, h, p)$ は、GNU MP の関数である。$mpz\_legendre(q, p)$ は、$(\frac{q}{p})$ を計算する。$mpz\_invert(h, h, p)$ は、p を法とした h（2 番目の引数）の逆数を h （1 番目の引数）に返す。

アルゴリズム 7 $Sqrt\_Mod(result, a, p)$

$Get\_QNR(g, p)$
$q \leftarrow p - 1$
$rem \leftarrow q \pmod 2$
$r \leftarrow 0$
while $rem = 0$
  do $q \leftarrow \frac{q}{2}$
    $rem \leftarrow q \pmod 2$
    $r \leftarrow r + 1$
$h \leftarrow g^q \pmod p$
$b \leftarrow a^q \pmod p$
$Sqrt\_Mod\_Sub(s, p, h, b, r)$
$result \leftarrow a^{\frac{q+1}{2}} s \pmod p$
return

アルゴリズム 8 $Get\_QNP(qnr, p)$

for $q \leftarrow 2$ **to** $p - 1$
  do if $mpz\_legendre(q, p) = -1$
    then $qnr \leftarrow q$
      return
return *error*

アルゴリズム 9 $Sqrt\_Mod\_Sub(s, p, h, b, r)$

$s \leftarrow 1$
$mpz\_invert(h, h, p)$
for $i \leftarrow r - 2$ **downto** 0
  do if $b^{2^i} \pmod p = -1$
    then $s \leftarrow sh \pmod p$
      $b \leftarrow bh^2 \pmod p$
    $h \leftarrow h^2 \pmod p$
return

3.3 メッセージと楕円曲線上の点の対応

$Msg\_to\_Pnt(M, m)$ は、メッセージ m を楕円曲線上の点 $M = (x, y)$ に対応付ける。$Pnt\_to\_Msg(m, M)$ は、楕円曲線上の点 M をメッセージ m に変換する。

アルゴリズム 10 $Msg\_to\_Pnt(M, m)$

for $i \leftarrow 0$ **to** 99
  do $x_i \leftarrow 100m + i$
    $s_i \leftarrow x_i^3 + ax_i + b$
    if $mpz\_legendre(s_i, p) = 1$
      then $Sqrt\_Root(y_i, s_i, p)$
        $M \leftarrow (x_i, y_i)$
        return
return *error*

アルゴリズム 11 $Pnt\_to\_Msg(m, M)$

$m \leftarrow \lfloor x/100 \rfloor$
return

3.4 Massey-Omura 暗号

$Massey\_Omura\_Alice(m)$ は、Alice がメッセージ m を暗号化し、Bob に暗号文を送る。$Massey\_Omura\_Bob()$ は、Bob は Alice からの暗号文を受信し、暗号文を復号する。*true* は真を表す。$mpz\_invert(k_{A\_inv}, k_A, N)$ は、N を法とした k_A の逆数を k_{A_inv} に返す。$mpz\_invert(k_{B\_inv}, k_B, N)$ も同様である。$random(seed)$ は乱数を返す関数である。send、receive は、Alice と Bob の間での送信、受信である。

アルゴリズム 12 $Massey\_Omura\_Alice(m)$

$Msg\_to\_Pnt(M, m)$
while *true*
  do $k_A \leftarrow random(seed)$
    if $gcd(k_A, N) = 1$
      then break
$Binary(M_1, k_A, M)$
send M_1 to Bob
receive M_2 from Bob

$mpz\_invert(k_{A\_inv}, k_A, N)$
$Binary(M_3, k_{A\_inv}, M_2)$
send M_3 to Bob

アルゴリズム 13　$Massey\_Omura\_Bob()$
receive M_1 from Alice
while *true*
　do $k_B \leftarrow random(seed)$
　　if $gcd(k_B, N) = 1$
　　then break
$Binary(M_2, k_B, M_1)$
send M_2 to Alice
receive M_3 from Alice
$mpz\_invert(k_{B\_inv}, k_B, N)$
$Binary(M_4, k_{B\_inv}, M_3)$
$Pnt\_to\_Msg(m, M_4)$

3.5　ECDSA

$Sign\_Alice(m)$ は、Alice がメッセージ m に署名し、Bob に署名文 (m, R, s) を送信する。$Verify\_Bob()$ は、Bob が署名を検証する関数である。

アルゴリズム 14　$Sign\_Alice(m)$
$k \leftarrow random(seed)$
$Binary(R, k, G)$
$mpz\_invert(k_{inv}, k, N)$
$s \leftarrow k_{inv}(m + ax) \pmod N$
send (m, R, s) to Bob

アルゴリズム 15　$Verify\_Bob()$
receive (m, R, s) from Alice
$mpz\_invert(s_{inv}, s, N)$
$u_1 \leftarrow s_{inv}m \pmod N$
$u_2 \leftarrow s_{inv}x \pmod N$
$Binary(V_1, u_1, G)$
$Binary(V_2, u_2, Q)$
$Point\_Add(V, V_1, V_2)$
if $V = R$
　then *accept*
　else *reject*

4　実行例

実行例 1 は楕円曲線暗号のパラメータ a、b、p、N、inf の設定を示している。実行例 2 は Massey-Omura 暗号、実行例 3、4、5 は ECDSA についての結果である。実行例 2 において、メッセージ msg=8765435486431 を楕円曲線上の点 M に対応付ける。M1、M2、M3、M4 は端末（Alice）と機器（Bob）との通信における暗号文 M_1、M_2、M_3、M_4 にそれぞれ対応する。msg=[8765435486431] は、暗号文を復号した結果である。

実行例 1

```
Elliptic Curve parameter:
     y^2 = x^3 + a*x + b (mod p)
a = 10
b = 134846241141436131261105411311693108758069491
    8677422294274
p = 156927543384667019095894735580191660402558886
    1116008628353
N = 156927543384667019095894735578028704030525554
    0896946997883
Infinity:
inf = 999999999999999999999999999999999999999999999
      99999999999999999
```

実行例 2

```
Massey-Omura Encryption:
msg = 8765435486431

M:(876543548643106,
   133207249678545540565987949256783865644620566 5
   808884104207)

M1:(730790666863198097498660253744381483002758758
    225032492091、
    485172728275404008016209170791768614406138010
    56498893389)

M2:(423526056820031495280345885174755655760474630
    129282623958、
    634217155110305216571519195339503367929047609
    546060934281)

M3:(129111957963126722451737517145527218449888327
    1579358420968、
    114416176456520792455045569323787432614607672
    3042717233527)

M4:(876543548643106、
    133207249678545540565987949256783865644620566
    5808884104207)

msg = [ 8765435486431 ]
```

生成元 G と秘密鍵 sa を選択し、公開鍵 Q=saG を計算する。

実行例 3

```
ECDSA Setting:
BasePoint G(G.x、G.y)
G.x = 1173123732641356773152361639530506865380315
      398604879179638
G.y = 9150388699998307896993499378392929894792121
      51162182558851
secret int sa : 157237245993378884061583032837171
      629950074461405774542247
Q = saG = (Q.x、Q.y)
Q.x = 1102475631922331488566438140096536289238255
      977330222225937
Q.y = 6867962940074833240748385241488195064371219
      91953623958351
```

メッセージを msg=91621338272768 として、乱数 k を選択し、R=kG、署名 sign を計算する。

実行例 4

```
ECDSA Sign:
msg = 91621338272768
k = 1173123732641356773152361639530506865380315398
    8604879179638
R = kG = (R.x, R.y)
R.x = 7157421154431209053000996064144107921635173
      8844525571065
R.y = 1408688967666688789901944887539265792324118
      894367949905297
sign = 6641861694763059144940144671512340208625036
      38788986127777
```

メッセージ msg、R、署名 sign より、V を計算し、署名が正しいことを検証する。

実行例 5

```
ECDSA Verify:
u1 = 1433311228360730506112180800860875313261039774
     880433923043
u2 = 8435497303097047848075145590827707745483125545
     16182283763
V1 = u1G = (V1.x, V1.y)
V1.x = 4337056519079104547354254830705161900609338
       31406302023642
V1.y = 1229083560204752313127377134498564518017708
       740377548240942
V2 = u2Q = (V2.x, V2.y)
V2.x = 5324204667883976990520799468767295471327392
       91118280246961
V2.y = 6354809916588312278408993554934978436816992
       18004141069458
V = V1 + V2 = (V.x, V.y)
V.x = 7157421154431209053000996064144107921635173
      8844525571065
V.y = 1408688967666688789901944887539265792324118
      894367949905297

V.x = 7157421154431209053000996064144107921635173
      8844525571065
R.x = 7157421154431209053000996064144107921635173
      8844525571065
V.x = R.x
V.y = 1408688967666688789901944887539265792324118
      894367949905297
R.y = 1408688967666688789901944887539265792324118
      894367949905297
V.y = R.y
V = R !!
```

5 まとめと今後の課題

本稿では、開発課題で構築した遠隔操作システムの操作者の認証とデータを秘匿するためのセキュリティ部分について述べた。楕円曲線暗号と電子署名は GNU MP を用いて、実用上安全な鍵長 190 ビットで実装した。

今後は計算の高速化を行い、楕円曲線暗号の安全性を Baby step Giant step、ρ 法、λ 法で検証したい。また、暗号に適した楕円曲線を選び、位数を計算することから始めたい。

参考文献

[1] I. Blake,G. Seroussi, and N. Smart. Elliptic Curves in Cryptography, Cambridge University Press,1999.

[2] L. Washington. Elliptic Curves: Number Theory and Cryptography, Chapman and Hall/CRC,2003.

[3] D. Stinson. Cryptography: Theory and Practice Third Edition, Chapman and Hall/CRC,2006.

[4] W. Mao. Modern Cryptography: Theory and Practice, Prentice Hall,2004.

[5] 木田祐司, 初等整数論, 朝倉書店,2001.

[6] The GNU Multiple Precision Arithmetic Library, http://gmplib.org/.

[7] 高信頼性/高セキュリティ制御システムの研究, 情報処理推進機構,2000.