整数の性質と公開鍵暗号1

袖ヶ浦高等学校 小俣 泰隆

1 はじめに

近年では，ネットワークの発達により，暗号 (特に公開鍵暗号) 等が話題となってきています。また，新課程に登場した，数学 A における「整数の性質」も含め，整数論や代数の分野がやや注目されるように感じています。こういったことを踏まえて，世の中で役立っている整数論の分野の公開鍵暗号について，話していきたいと思います。

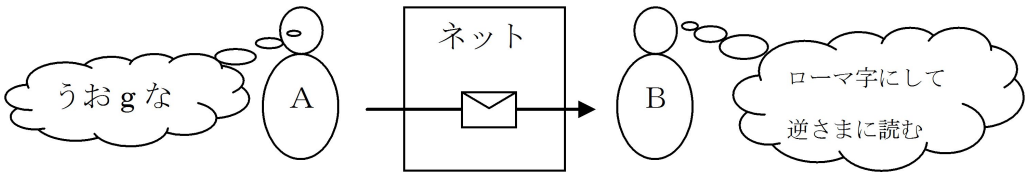
2 公開鍵暗号について

2.1 秘密鍵暗号と公開鍵暗号

公開鍵暗号を簡単に説明しますと，一方向性関数 (正確には一方向性擬似関数) を利用した暗号方法で，暗号化の方法を一般公開しても秘匿通信が可能となる暗号です。

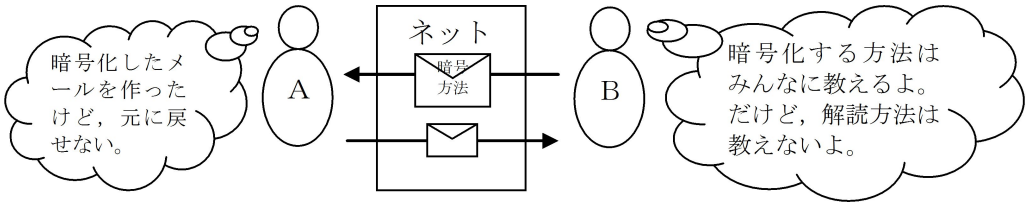
公開鍵暗号の具体的な説明をする前に，先に秘密鍵暗号 (共通鍵暗号ともいう) と公開鍵暗号の違いをみてみましょう。

下の図のように，A さんが B さんに誰にもばれないように暗号化してメールを通信する方法を考えてみましょう。



真っ先に思いつくのが，送り手と受け手がルール (暗号化の方法) を決めて，受け手が解読して読む方法だと思います。この暗号方法を秘密鍵暗号 (共通鍵暗号) といいます。仰々しく書いていますが，子供にも親しみがあり，有名な「たぬき」の暗号がこれに該当します。ところが，何度も利用しているといずれ第三者が暗号の解読方法に気付いてしまいます (「たぬき」の暗号がいい例)。そのため，定期的に暗号化する方法を変える必要があります。しかし，その暗号化する方法も誰にもばれないように B さんに伝えないといけないため，結局は本末転倒のような暗号というわけです。つまり，暗号化する方法を秘密裏に伝えられる方法があるなら，その秘密裏に伝えられる方法を用いて暗号化せずにメールを送ればよいことになります。

これを改善したのが公開鍵暗号というもので，次の図のような方法です。



まず，メールを受け取る B さんが暗号方法を指定（一般公開）します。A さんは B さんにメールを送る際にその方法でメールを暗号化して送ります。ところが，その暗号化したメールは暗号化した A さんでも解読できません。つまり，暗号化するのは簡単だが，解読する（復号化する）のが困難な暗号ということです。この一連の仕組みの暗号を公開鍵暗号といい，現在，インターネット上のネット銀行やパスワードで使われています。

ここでもう一度，公開鍵暗号の仕組みを見直してみると，公開鍵暗号にも 2 つの疑問が残ります。

(1) 暗号文を盗むことに成功したとして，暗号化する方法を公開しているので，暗号化する方法の逆をたどっていけば解読できるのではないか。
(2) 暗号化したメールを B さんは容易に解読できるのか。

この 2 点が気になるところでが，これらを解消したのが一方向性関数です。具体的には次で詳しく説明します。

2.2 RSA 暗号と一方向性関数

まず，最初に RSA 暗号の根幹である積と素因数分解の関係をみてみましょう。たとえば，下の (3) と (4) の問題をみてください。

(3) 21649 × 513239 の計算をしなさい。
(4) 11111111111(1 が 11 桁) の素因数分解をしなさい。

電卓を用いてもよいので (3)，(4) の答えを出すことができますか？(3) は容易に求められますが，(4) がなかなか出せないものです。素因数分解の方法は知っている素数で割っていくという，下手な鉄砲も数打てば当たる方法が一般的です。そのため，桁が大きくなるにつれて，簡単には素因数分解ができなくなります。実際，この計算を試した方はお気づきでしょうが，$21649 \times 513239 \Longrightarrow 11111111111$ となるのはすぐ求められますが，$11111111111 \Longrightarrow 21649 \times 513239$ の答えを導くのは困難であることがわかります。こういった，一方への計算が容易で，逆の計算が困難な関数を一方向性関数 (一方向性擬似関数) [1] といいます。この一方向性関数の性質が公開鍵暗号には不可欠な要素です。

ここで，2.1 で述べた公開鍵暗号の仕組みを思い出してみましょう。まず，B さんが暗号化する方法を一般公開しました。この暗号化する方法の鍵が RSA 暗号では「11111111111」にあた

[1]「一方向性関数」の正確な定義は一方への計算が可能で，逆の計算が不可能な関数を指す。今回のような素因数分解は，時間をかければいずれ計算ができてしまうが計算が困難であることから，一方向性関数の性質に似ているので「一方向性擬似関数」という。

ります。一方で，B さんは解読する秘密の鍵をもっており，これが「21649 × 513239」にあたります。この鍵は誰にも知られてはいけません。まず，2.1 の (1) の疑問に関して，解読する方法が「11111111111」を素因数分解するところにあります。さきほど述べたとおり，これは困難な計算でした。また，(2) の疑問に関しては，B さんが暗号化の鍵を作る際に，桁の大きい 2 つの素数をとってきて鍵を作るので，暗号化する鍵を作る際に解読する鍵も容易に作られます。つまり，一方向性関数の性質が公開鍵暗号の仕組みを作っています。

2.3 RSA 暗号の構成

2.2 を踏まえて実際にはどのように暗号化，復号化（解読）していくのか見てみましょう。

定義 1.1　RSA 暗号
RSA 暗号とは以下のような操作で暗号化と復号化 (解読) を行うことである。
p, q を十分に大きな異なる素数とする。
n を $n = pq$ とする。
暗号化

暗号化鍵： (n, e)　　(公開)
　　$\exists e \in \mathbf{N}$ s.t.　$\mathrm{GCD}(e, (p-1)(q-1)) = 1$
暗号化関数： $C \equiv M^e \mod n$ (M; メッセージ，C; 暗号文)

復号化

暗号化鍵： (n, d)　　(非公開)
　　$\exists d \in \mathbf{N}$ s.t.　$ed \equiv 1 \mod \mathrm{LCM}(p-1, q-1)$
暗号化関数： $M \equiv C^d \mod n$ (M; メッセージ，C; 暗号文)

記号に関しては脚注[2]参照

すべての文字や記号は PC の中では 2 進数の自然数のため，M(メッセージ) や C(暗号文) もすべて自然数に置き換えられます。まず，M を e 乗して，n で割り，余りを出します。[3]この余りが暗号文 C です。ただし，自然数と互いに素でなければなりません。また，n 以下のそこそこ大きな数であると，解読が簡単にはできません。仮に第三者が解読を行ったとしましょう。すぐに思いつく方法は 2 つです。

(5) 暗号文 C をみて，適当なメッセージを手当たりしだい e 乗して，C と一致しないか確かめる。

[2] “$\mathbf{N}$” は自然数，“$\mathrm{GCD}(x, y)$” は x, y の最大公約数，“$\mathrm{LCM}(x, y)$” は x, y の最小公倍数。“A s.t. B” は such that の略。A というのは説明すると B というものである。“mod” は余り。(例；「$8 \equiv 5 \mod 3$」 8 を 3 で割った余りは 5 を 3 で割った余りと合同である。)

[3] 余りの計算に関しては，一個一個を掛け算して余りを出すと，コンピュータの計算速度が指数的になってしまい，時間がかかりすぎてしまうので，多項式時間で終えられる mod の性質を利用した特殊な計算を行う。(詳しくは lp029omata@gmail.com に質問を送ってください。)

(6) 暗号文 C を 1 乗，2 乗と計算して元の文 M を導くという方法[4]もある。

ここで，M(メッセージ) は 1∼100 桁の任意な数として，1 回のコンピュータによる計算が 10^{-30} 秒だとしても，M を解読するには少なくとも，$10^{100}/(3600 \times 24 \times 365 \times 10^{30})$ 年，つまりは 10^{60} 年はかかってしまいます。たとえ解読できたとしても，その前に人の寿命がきてしまいますので，解読はできないといってよいでしょう。

次に復号化 (B さんの解読) の方法ですが，オイラーの定理を利用しますので，オイラーの定理の証明からみてみます。

定理 1.2　オイラーの定理
整数 n と互いに素である整数 a に対して，

$$a^{\phi(n)} \equiv 1 \mod n \quad (\phi(\cdot) \text{ はオイラー関数とする})$$

が成立する。
オイラー関数とは，約数の個数を表す関数で，例として，
$n = 12$ とすると，$\phi(12) = 6$ $(1, 2, 3, 4, 6, 12)$ となる。
$n = pq(p, q$ は素数) とすると，$\phi(n) = (p-1)(q-1)$ となる。

$x_i (1 \leqq i \leqq \phi(n))$ は n と互いに素で $x = \{x_1, x_2, \cdots, x_{\phi(x)}\}$ とします。
ax_i は n と互いに素であるので，

$$ax_i (\mod n) \in x \ , \ ax_i \neq a_j (\mod n) (i \neq j)$$

したがって，

$$\begin{aligned} ax_1 \cdot ax_2 \cdot \ \cdots \ \cdot ax_r &\equiv x_1 \cdot x_2 \cdot \ \cdots \ \cdot x_r \mod n \\ a^r &\equiv 1 \mod n \ (\text{両辺を } x_1 \cdot x_2 \cdot \ \cdots \ \cdot x_r \text{で割った}) \end{aligned}$$

このようにして a の $\phi(x)$ 乗を n で割った余りが 1 になることがわかります。では，復号化関数をみて見ましょう。
$C^d \equiv M \mod n$ を示す。

$$\begin{aligned} C^d &= (M^e)^d && (\text{暗号文の定義より}) \\ &= M^{ed} \\ &= M^{1+k(p-1)(q-1)} && (ed \text{ の定義より, } ed \text{ は } (p-1)(q-1) \text{ の最小公倍数}) \\ &= M^{1+k\phi(n)} \\ &= M \cdot M^{k\phi(n)} \\ &= M \cdot 1^k && (\text{オイラーの定理より}) \\ &= M \end{aligned}$$

というようにして暗号化を解読します。つまり，d という数がわかれば解読ができるということになります。ところが，d は e とかけた数が $p-1$ と $q-1$ の最小公倍数の余りが 1 になるような数です。e の値は公開されていますが，$p-1$ と $q-1$ の最小公倍数は容易にはわか

[4](6) の性質は群論における体や環であればおきる。尚，RSA 暗号の暗号文 C やメッセージ M は Z/nZ $(n = pq)$ という環の要素である。

りません。第三者が知り得る情報を整理すると C(暗号文) と n(暗号化鍵 1) と e(暗号化鍵 2) だけです。ここで，暗号を作った B さんは素数 p, q を決めて n の値を決めた人なので，p, q の値は知っており復号化できますが，第三者は素因数分解のされていない（p, q のわからない）n を知っているだけなので，復号化することができません。ここがまさに素因数分解の困難を利用したところとなります。

3 整数の性質と公開鍵暗号

高校の数学 A における「整数の性質」の中の最大公約数を簡単に導き出すことのできる「ユークリッドの互除法」は RSA 暗号の計算処理で多く使われています。

たとえば，

(7) 互いに素 $\Longleftrightarrow$ 最大公約数が 1 であるということ
(8) (a, b) の最小公倍数 $\Longleftrightarrow$ $a \times b \div$ 最大公約数

はユークリッドの互除法を用いて導いています。こういった点をふまえると，中学校で習った素因数分解や，高校で学ぶ「ユークリッドの互除法」は最新技術の一端を担っているといっても過言ではないと思います。

公開鍵暗号は RSA 暗号がよく知られていますが，他にエルガマル (ElGamal) 暗号や楕円曲線暗号などがあります。どちらも，異なる一方向性関数を使っており，特に楕円曲線暗号は図形と暗号を合体させたもので，これまでとは異なる暗号ともいえます。次回，$\alpha - \omega$ に記載させていただける機会がありましたら，ぜひ「整数の性質と公開鍵暗号 2」という題でやらせていただきたいと思います。

最後に，Perl 言語ではありますが，今回 RSA 暗号で使用した，ユークリッドの互除法と，拡張ユークリッドの互除法のプログラムを記載させていただきます。

＜ユークリッドの互除法＞

```
01:print"最大公約数を求めます。値を二つ代入してください\n";
02:$a=<>;
03:$b=<>;
04:while(($a%$b)!=0){
05:   ($a,$b)=($b,$a%$b);
06:}
07:print("最大公約数は$b\n");
```

S1：a, b の入力
S2：$b \neq 0$ ならば S2〜S4 を繰り返す
S3：$b \neq 0$ ならば $a = qb + r$, $0 \leqq r < b$ で r を求める
S4：(b, r) を新しい (a, b) とする
S5：a を出力

＜拡張ユークリッドの互除法＞ ($ed \equiv 1 \mod \mathrm{LCM}(p-1, q-1)$ で利用)

```
print"拡張ユークリッドの互除法\n";
print"値を二つ代入してください\n";
$a=<>;
$b=<>;

$r[1]=$a; $x[1]=1; $y[1]=0;
$r[2]=$b; $x[2]=0; $y[2]=1;

while($r[2]!=0){
$r[3]=$r[1]%$r[2];
$q=($r[1]-$r[3])/$r[2];

$x[3]=$x[1]-$q*$x[2];
$y[3]=$y[1]-$q*$y[2];

($r[1],$r[2],$x[1],$x[2],$y[1],$y[2])=($r[2],$r[3],$x[2],$x[3],$y[2],$y[3]);
}
print"$r[1],$x[1],$y[1]\n";
print"$r[1]=$a*$x[1] + $b*$y[1] ";
```

S1：a, b の入力
S2：$r_1 = a$, $x_1 = 1$, $y_1 = 0$, $r_2 = b$, $x_2 = 0$, $y_2 = 1$ とする。
S3：$r_2 \neq 0$ ならば S3〜S6 を繰り返す。
S4：$r_1 = qr_2 + r_3$, $0 < r_3 < r_2$ で r_3 を求める。
S5：$x_3 = x_1 - qx_2$, $y_3 = y_1 - qy_2$ とする。
S6：$(r_2, r_3, x_2, x_3, y_2, y_3) = (r_1, r_2, x_1, x_2, y_1, y_2)$
S7：r_1, x_1, y_1 を出力 $(r_1 = ax_1 + by_1)$