RSA 暗号について

1. はじめに

RSA 暗号は，1978 年に Rivest, Shamir, Adleman の 3 人によって発明された公開鍵暗号である．

RSA 暗号の原理を理解するために，フェルマーの小定理が重要な役割を果たす．フェルマーの小定理とは，p を素数とするとき，任意の正整数 $a(< p)$ について，

$$a^{p-1} = 1 \bmod p \tag{1}$$

が成立するという定理である．そこで，以下のような簡単な暗号を考えてみる．まず，$p-1$ と互いに素な整数 e を選び，

$$e \cdot d = 1 \bmod (p-1) \tag{2}$$

となる整数 d をユークリッドの互除法により求める．そして，メッセージ M を暗号化して暗号文 C を得るときは，

$$C = M^e \bmod p \tag{3}$$

とすることにする．このとき，復号は，

$$M = C^d = M^{ed} \bmod p \tag{4}$$

により可能であることがわかる．上記の方法を暗号として利用するためには，p と e の双方を秘密にしなければならないから，公開鍵暗号にはならない．

2. RSA 暗号の原理

RSA 暗号は，式 (3) の法 p を，二つの素数 p, q の積 $n = pq$ に置き換えたものである．このとき，式 (1) に相当する式は，

$$a^{\varphi(n)} = 1 \bmod n \tag{5}$$

となる (ただし，$(a, n) = 1$)．$\varphi(n)$ はオイラー関数であって，$\varphi(n) = (p-1)(q-1)$ である．

RSA 暗号を利用する場合，以下のような手順を踏むことになる．まず，2 つの大きな素数 p と q を決め，積 $n = pq$ を求める．次に，オイラー関数 $\varphi(n)$ と互いに素な整数 e を選び，

$$e \cdot d = 1 \bmod \varphi(n) \tag{6}$$

となる d を求める．以上で準備は終了であり，n と e を公開鍵として公開する．

メッセージ (平文と呼ぶ)M を暗号文 C に暗号化するには，

$$C = M^e \bmod n \tag{7}$$

とすればよい．暗号化は，公開されている n と e のみによって計算可能であることに注意する．復号は，

$$M = C^d = M^{ed} \bmod n \tag{8}$$

により可能である．このとき，d は公開されていないから，復号が可能であるのは p と q を知っている正規の受信者のみであることがわかる．

正規の受信者でない者が n と e を用いて，暗号文を復号することは可能だろうか．e から d を得るためには，式 (6) を解く必要があるが，オイラー関数 $\varphi(n) = (p-1)(q-1) = n-p-q+1$ であるから，p と q の値がわからなければ，$\varphi(n)$ を求めることはできない．もちろん，n を素因数分解すれば p と q を求められるが，p と q の大きさを 10 進で 170 桁程度以上に選んでおけば，現在最速の素因数分解アルゴリズムを用いても現実的な時間で素因数分解することは不可能であることが知られており，素因数分解により p と q を求めることはできないと考えてよい．

なお，式 (5) において，a の指数 (べき乗して初めて 1 になる数) は，$\varphi(n)$ ではない．なぜなら，

$$a^{p-1} = 1 \bmod p$$
$$a^{q-1} = 1 \bmod q$$

であるから，

$$a^{\mathrm{lcm}(p-1,q-1)} = 1 \bmod n \tag{9}$$

となるからである．$\mathrm{lcm}(p-1, q-1)$ は $\varphi(n)$ より小さいので，一般にはこちらが用いられる．すなわち，式 (6) において，e を $\mathrm{lcm}(p-1, q-1)$ と互いに素になるように選び，

$$e \cdot d = 1 \bmod \mathrm{lcm}(p-1, q-1) \tag{10}$$

となる d を求めるようにする．

3. RSA暗号によるディジタル署名

RSA 暗号は，暗号としての利用だけでなく，ディジタル署名としての利用も可能である．

ディジタル署名として利用するときは，署名者は，署名したいメッセージ M を，n と秘密鍵 d を用いて

$$S = M^d \bmod n \tag{11}$$

により S に変換する．S が署名となる．署名の検証者は，公開されている n と e を用いて

$$M = S^e \bmod n \tag{12}$$

を計算し，もとのメッセージ M が復元されれば，署名 S が正しいということを検証できる．なぜなら，S を作成するためには秘密鍵 d が必要であり，それを知っているのは正規の署名者のみのはずだからである．

なお，実際には，計算量を節約するために，メッセージ M を直接用いるのではなく，一方向性ハッシュ関数[1] $h(\cdot)$ を用いて，ハッシュ値 $h = h(M)$ を求め，h に対する署名を用いることが多い．

4. 具体例

実用的な RSA 暗号を設計するためには，素数 p，q として 10 進 170 桁以上の大きな素数を選ぶ必要がある．このような大きな素数は，計算機を用いることにより比較的簡単に得ることができる．

ここでは，RSA 暗号の原理を理解するために，ごく小さな素数 $p = 3$ と $q = 11$ を用いた例を示す．このとき，$n = pq = 33$ となる．$\mathrm{LCM}(p-1, q-1) = 10$ であるので，$e = 3$ と選んでみよう．$3 \times 7 = 1 \bmod 10$ であるので $d = 7$ である．以上より，公開鍵は $n = 33$ と $e = 3$ であり，秘密鍵は $p = 3, q = 11$ と $d = 7$ である．

メッセージ $M = 5$ として暗号文を求めてみよう．

$$C = M^e = 5^3 = 26 \bmod 33$$

となる．復号は，

$$M = C^d = 26^7 = 5 \bmod 33$$

となり，もとのメッセージを復元できることがわかる．

5. 暗号化・復号処理の高速化

RSA 暗号は，その暗号化，復号化処理時に大きな法 n によるべき乗計算を行う必要があり，計算時間の短縮化が大きな課題となっている．

式 (7) では，$e-1$ 回の乗算を行う必要があるように見えるが，実際はそうではない．例えば，$e = 10$ とすれば，

$$M^{10} = ((M^2)^2)^2 \times M^2$$

と変形することにより，4 回の乗算で計算できることがわかる．一般に，式 (7) の計算は，高々 $2\lfloor \log_2 e \rfloor$ 回の乗算により計算することができる．

次に，復号化 (式 (8)) の計算は，$n = pq$ の関係を用いることにより更に高速化することができる．

まず，秘密鍵 d に対して，$d_1 = d \bmod (p-1)$ と $d_2 = d \bmod (q-1)$ を計算しておく．次に，

[1] ハッシュ関数とは，長いメッセージを「圧縮」して，短い (数百ビット程度) 数値に変換する関数のこと．一方向性ハッシュ関数は，$h = h(M)$ であるとき，$h = h(M')$ となる他のメッセージ M' を見つけることが困難な性質を持つハッシュ関数である．

暗号文 C に対して，$C_1 = C \bmod p$，および，$C_2 = C \bmod q$ を求め，

$$\begin{aligned} M_1 &= C_1^{d_1} \bmod p \\ M_2 &= C_2^{d_2} \bmod q \end{aligned}$$

を計算する．すると，

$$\begin{aligned} M &= M_1 \bmod p \\ M &= M_2 \bmod q \end{aligned}$$

であるから，中国剰余定理 (Chinese Remainder Theorem) により M を求めることができる．この方法では，n に比較して小さな数 p と q を法とする計算を行えばよいこと，M_1 と M_2 を求める計算は並列処理が可能であること等の理由により，4〜8 倍の高速化をはかることができる．

付録

式 (6) あるいは，式 (10) では，$\varphi(n)$ や $\mathrm{lcm}(p-1, q-1)$ を法として e の逆数を求める必要がある．式 (6) や式 (10) の法を m とおけば，適当な整数 r が存在して

$$e \cdot d + r \cdot m = 1$$

と変形できるから，e と m の最大公約数を求めるユークリッドの互除法を利用して d を求めることができることがわかる．このアルゴリズムを以下に示す．

```
int inv( int e, int n )
{
    int q, s=e, t=n, u=1, v=0, w;
    while( s>0 ){
        q = t/s;
        w = t-q*s; t=s; s=w;
        w = v-q*u; v=u; u=w;
    }
    if( v<0 ) return( v+n );
    else      return( v );
}
```

上記のプログラムでは，型名が int になっているが，もちろん，通常計算機で扱われる整数型 (32bit) では暗号用途に用いることはできない．適切な多倍長整数計算ライブラリを用いる必要がある．

多倍長整数計算ライブラリは自分で作成することもできるが，さまざまなソフトウェアツールが存在するので利用するとよい．ここでは，PARI/GP(http://pari.math.u-bordeaux.fr/) というツールを紹介しておく (課程計算機システムでもインストールされている)．PARI/GP は，GPL に従って利用することができる．

PARI/GP は C 言語のライブラリとして利用する方法と，コマンドラインから対話的に利用する方法とがある (対話的に利用するプログラムが GP である)．GP を起動するとプロンプト (gp>) が出るので，ほぼ数式の通りにコマンドを入力すればよい．以下に，RSA 暗号の鍵を作成し，暗号化・復号化を行ってみた例を示す．

なお，#で始まる行は説明のためのコメントであり，実際には入力しない．また，% で始まる行はシステムの応答メッセージ (計算結果) である．

```
gp> p=nextprime(random(10^15))
# 10 進 15 桁のランダムな素数を作る
%1 = 726379153703171
gp> q=nextprime(random(10^15))
%2 = 983831585518963
gp> n=p*q
%3 = 714634754475713249199293731673
gp> l=lcm(p-1,q-1)
# p-1 と q-1 の最小公倍数 l を求める
%4 = 357317377237855769494277254770
gp> e=random(l)
# l までの乱数 e を作る
%5 = 141988331376849742978770366511
gp> gcd(e,l)
# e と l が互いに素か確認．素でなければ e を作り直す
%6 = 1
gp> d=lift(Mod(1/e,l))
```

```
# 法 l で e の逆数を求める. lift は通常の整
数に変換する関数
%7 = 248369886954536325338177745991
gp> m=Mod(random(n),n)
# 平文 m を法 n の下でランダムに選ぶ
%8 = Mod(64777820050671248855127634539, 714634754475713249199293731673)
gp> c=m^e
# 暗号化 (m は Zn の要素なので Mod は不要)
%9 = Mod(44043154285754551516841494607 5, 714634754475713249199293731673)
gp> c^d
# 復号して元に戻ることを確認
%10 = Mod(64777820050671248855127634539, 714634754475713249199293731673)
```