ElGamal 暗号の高速化法

ElGamal 暗号に中国人剰余定理を用いて復号処理を高速化する方法を考えました。さらに，暗号化の冪乗の指数を小さくしても安全であると考えられる方法を考えました。これにより ElGamal 暗号が「世界最速の公開鍵暗号」となります。ただし，公開鍵が通常の ElGamal 暗号より大きくなります。

第一のポイントは，p，q を素数として，合成数 $N = pq$ を法とする剰余類上に暗号を構築したことです。これにより中国人剰余定理を用いて，復号の冪乗の指数を小さくすることができます。第二のポイントは公開鍵の数を増やして，暗号化の冪乗の指数を小さくしても低指数攻撃を避けられるようにしたことです。また，通常の ElGamal 暗号は離散対数問題の困難性を安全性の根拠としますが，この ElGamal 暗号は合成数 $N = pq$ の素因数分解の困難性を安全性の根拠とします。

鍵作成

・素数 p，q を秘密鍵として $N=pq$ を計算する。

・乱数 a_0 を生成する。

・$a_p = a \bmod p$，$a_q = a \bmod q$ とする。

自然数 $m_p, m_q\ (m_p \neq m_q)$ を秘密鍵として，$b_p = a_p^{\,m_p} \bmod p$ と $b_q = a_q^{\,m_q} \bmod q$ を計算する。

中国人剰余定理により，$b_0 \bmod p = b_p$ かつ $b_0 \bmod q = b_q$ を満たす b_0 を求める。（註 1 を参照）

次に，N と同じ程度の bit 数の大きな乱数 $n_1, n_2, \cdots$ を選んで，$a_1 = a_0^{\,n_1}$，$a_2 = a_0^{\,n_2} \cdots$ と $b_1 = b_0^{\,n_1}$，

$b_2 = b_0^{\,n_2} \cdots$ を計算する。

・$N, a_0, a_1, a_2, \cdots, b_0, b_1, b_2, \cdots$ を公開鍵とする。（註 2 を参照）

暗号処理

・N が 2048bit 程度の数の場合，合計が 112bit 以上の自然数の乱数 $r_0, r_1, r_2, \cdots$ を生成する。

・$e = a_0^{\,r_0} a_1^{\,r_1} a_2^{\,r_2} \cdots \bmod N$ と $f = b_0^{\,r_0} b_1^{\,r_1} b_2^{\,r_2} \cdots \bmod N$ を計算する。

・f から共通鍵を作成し，共通鍵暗号でメッセージ M を暗号化して M' を得る。

・e，M' を受信者に送信する。

復号処理

・$e_p = e \bmod p$ と $e_q = e \bmod q$ を求める。

・$x_p = e_p^{\,m_p} \bmod p$ と $x_q = e_q^{\,m_q} \bmod q$ を計算する。

・中国人剰余定理により，$x \bmod p = x_p$ かつ $x \bmod q = x_q$ を満たす x を求める。このとき，$x = f$ が成り立つ。　（註 3 を参照）

・x から共通鍵を作成し，共通鍵暗号でメッセージ M' を復号して M を得る。

註 1　$a^m \bmod N = b$ を満たす自然数 m が存在するのは，$a^i \bmod p, a^i \bmod q$ の基本周期をそれぞれ，S，T とするとき，$m_q - m_p$ が S，T の最大公約数の倍数であるときであり，存在するときは N と同じ bit 数程度の大きな自然数になる。また，その際に $m_p = 2, m_q = 3$ 程度に小さく選んでも良いように思われる。このとき，$b = a^m$ を満たす m が存在しないようにすることができる。（補足を参照）

註 2　公開鍵の数を増やし暗号化を複雑にしたのは，低指数攻撃を避けるためである。公開鍵の数を増やしても安全性が損なわれることはないことは，公開鍵のペア a_0, b_0 から $a_1 = a_0^{n_1}$，$a_2 = a_0^{n_2}$ …と $b_1 = b_0^{n_1}$，$b_2 = b_0^{n_2}$ …を誰でも作れることから明らかである。

註 3　$x = f$ であることは，次のように示される。

$$b_0 = a_0^{m}, \quad b_k = b_0^{n_k} = (a_0^{m})^{n_k} = (a_0^{n_k})^{m} = a_k^{m} \quad (k = 1, 2, 3, \cdots)$$

$$e = a_0^{r_0} a_1^{r_1} a_2^{r_2} \cdots, \quad f = b_0^{r_0} b_1^{r_1} b_2^{r_2} \cdots$$

であるから

$$f = (a_0^{m})^{r_0} (a_1^{m})^{r_1} (a_2^{m})^{r_2} \cdots = (a_0^{r_0} a_1^{r_1} a_2^{r_2} \cdots)^m = e^m$$

註 4　この暗号の欠点は，公開鍵の数が多いことである。これを減らすために，$a_0, a_1, a_2, \cdots$ を固定値にすることが考えられる。このとき，$b_0, b_1, b_2, \cdots$ は，a_0 から b_0 を求めたのと同様にして，中国人剰余定理により p, q と $a_0, a_1, a_2, \cdots$，m_p, m_q を用いて求めればよい。

暗号化の高速化

公開鍵のペアを $a_0, a_1, a_2, \cdots, b_0, b_1, b_2, \cdots$ と増やせば，同じ安全性で暗号化の計算量を減らすことができる。しかし，公開鍵の数を増やすと鍵の配送に時間がかかるため，公開鍵のペアの数を増やしたくない。そこで，公開鍵を a_0, b_0 と a_1, b_1 だけとし，r, s を乱数として $e = a_0^r a_1^s$ および $f = b_0^r b_1^s$ を計算するときに，少ない計算量で大きな r, s の値を得る技法を示す。

flag(i)を 0，1，2，3 の値をとる 2bit の乱数の列として，次のアルゴリズムにより $e = a_0^r a_1^s$ を求める。

```
LET e1=a0
LET e2=a1
FOR i=1 TO kaisuu
    LET x=e1*e2
    LET e1=e2
    IF flag(i)=0 OR flag(i)=1 THEN
```

```
        LET e2=x
    ELSEIF flag(i)=2 THEN
        LET e2=x*e2
    ELSE
        LET e2=x*x
    END IF
NEXT i
```

e についての漸化式は，flag(i)=0 の 0，1 に応じてそれぞれ $e_{n+2}=e_{n+1}e_n$，$e_{n+2}={e_{n+1}}^2e_n$ となるが，$e=a^rb^s$ における指数は，

$$r_1=1, r_2=0,\quad r_{n+2}=r_{n+1}+r_n \text{ または } r_{n+2}=2r_{n+1}+r_n \text{ または } r_{n+2}=2r_{n+1}+2r_n$$

$$s_1=0, s_2=1,\quad s_{n+2}=s_{n+1}+s_n \text{ または } s_{n+2}=2s_{n+1}+s_n \text{ または } s_{n+2}=2s_{n+1}+2s_n$$

を満たす。漸化式の特性方程式の解はそれぞれ

$$\frac{1\pm\sqrt{5}}{2}, 1\pm\sqrt{2}, 1\pm\sqrt{3} \quad \text{であり} \quad \frac{1+\sqrt{5}}{2}\approx 1.618,\quad 1+\sqrt{2}\approx 2.414,\quad 1+\sqrt{3}\approx 2.732$$

である。シミュレーションによると，flag(i)=0 に応じた $e=a^rb^s$ の指数 r，s のそれぞれには重なりがあるが，r と s の組み合わせでは重なりがほとんどないと予想される。
2048bit の暗号では kaisuu=112 とすれば十分である。このとき，計算量は 112bit の指数の計算量程度であり，安全性は 112bit の暗号強度であると考えられる。

処理速度

RSA 暗号の暗号化処理では，冪乗の指数の値は 3 か 65537 を使うため，計算量は極めて少ない。また，RSA 暗号では，復号に中国人剰余定理を用いるため，計算量は ElGamal 暗号の復号の $\frac{1}{4}$ となる。

一方，この暗号では，指数 m_p, m_q を極端に小さくとれば，計算量は極めて少ない。

暗号化では，「暗号化の高速化法」を用いれば，2048bit の暗号で冪乗の計算量を 112bit 程度に抑えられるから，通常の ElGamal 暗号に対する計算量の比は $\frac{112}{2048}\approx 0.055$ 倍となる。さらに，暗号化の冪乗の指数の合計を，3072bit の暗号では 128bit 程度に，7680bit の暗号では 192bit 程度に抑える。

2048bit の ElGamal 暗号の復号における冪乗の計算量を 1 として，冪乗の計算量の見積もりを示す。

暗号強度	法の bit 数	RSA 暗号		ElGamal 暗号		この暗号	
		暗号化	復号	暗号化	復号	暗号化	復号
112bit	2048bit	僅少	0.25	2	1	0.11	僅少
128bit	3072bit	僅少	0.84	6.8	3.4	0.28	僅少
192bit	7680bit	僅少	13.2	105	53	2.6	僅少

ただし，RSA 暗号や署名では，法とする数より小さい数を掛けるため，計算量は上記より減少する。
また，積の計算量は法の bit 数の 2 乗で見積もっているが，Karatsuba 法を用いれば計算量は減少する。
その場合でも，同じ暗号強度で比べれば，各暗号の計算量の比は変わらない。

安全性についての考察

RSA 暗号やラビン暗号は素因数分解の困難性を安全性の根拠とし，一般の ElGamal 暗号は離散対数問題の困難性を安全性の根拠とする。しかし，ElGamal 暗号を素数の積 $N = pq$ を法とする剰余類上に構築した場合は，ElGamal 暗号も素因数分解の困難性を安全性の根拠とすることを示す。

離散対数問題 $a^r \equiv b \mod N$ を解くのに，離散対数問題 $a^r \equiv b^2$ を解いてもよい。

$a^r \equiv b^2$ が解けて，2 つの偶数の解 r_1, r_2 が求まったとする。このとき，$\alpha = a^{r_1/2}, \beta = a^{r_2/2}$ は方程式

$x^2 \equiv b^2$ の解であるから，$\alpha^2 \equiv b^2$，$\beta^2 \equiv b^2$ を満たす。よって

$$\alpha^2 - \beta^2 \equiv 0 \mod N \quad \text{すなわち} \quad (\alpha+\beta)(\alpha-\beta) \equiv 0 \mod N$$

が成り立つ。したがって，$\alpha+\beta$ と N および $\alpha-\beta$ と N との最大公約数を求めれば，条件により N の素因数が求まる。これにより

離散対数問題の解が求まる ⇒ $N = pq$ の素因数分解ができる

が示されたことになる。この逆である

$N = pq$ の素因数分解ができる ⇒ 離散対数問題が解ける

は，素因数分解ができれば離散対数問題を $\mod p$ および $\mod q$ で解くこととなるので，成立するとしてよいであろう。すなわち，素数の積 $N = pq$ を法とする剰余類上の ElGamal 暗号の離散対数問題は，素因数分解の困難性と同じ安全性であると考えられる。

補足

定理　中国人剰余定理の拡張

自然数 m, n に対して，m, n の最大公約数を g，最小公倍数を l とおくとき，連立合同式

$$\begin{cases} x \equiv a \pmod{m} \\ x \equiv b \pmod{n} \end{cases} \quad \cdots①$$

が整数解を持つための必要十分条件は，$a \equiv b \pmod{g}$ である。また，この条件が成り立つとき，整数解 x は l を法として一意的に定まる。

証明

①が整数解を持つとすると，

$$x = a + sm = b + tn$$

となる整数 s, t が存在する。これより

$$sm - tn = b - a \quad \cdots②$$

よって，$b-a$ は g の倍数であるから，$a \equiv b \pmod{g}$

逆に，$a \equiv b \pmod{g}$ のとき②は整数解 s, t をもつから，$x = a + sm = b + tn$ とおけば①の解になる。

また，x, x' が共に①の整数解であるとすると，

$$x \equiv x' \pmod{m} \text{ かつ } x \equiv x' \pmod{n}$$

であるから　$m \mid x - x'$ かつ $n \mid x - x'$

したがって，$x - x'$ は m, n の公倍数である。よって $l \mid x - x'$ であるから $x \equiv x' \pmod{l}$

すなわち，x は l を法として一意的に定まる。

証明終り

次に，実際に①の解を求める。まず，拡張ユークリッドの互除法により

$$sm + tn = g \cdots③$$

となる整数 s, t を求める。このとき

$$(b-a)sm/g+(b-a)tn/g=b-a$$

よって，

$$x=a+\{(b-a)/g\}sm=b+\{(a-b)/g\}tn$$

は①の解である。③を用いて変形すると

$$x=a(1-sm/g)+bsm/g=atn/g+bsm/g$$

これを l で割った余りを求めればよい。

十進 BASIC のコード

以下に，$N=pq$ の素因数分解が求まる例を示す。

```
!ElGamal 暗号
LET p=8423 !安全素数
LET q=7823
LET pq=p*q
LET a=314159 !公開鍵
LET n=1234567
LET b=ruijyou(a,n,pq)
LET b=MOD(b^2,p*q)
CALL dlog(a,b,pq,s,t)

LET x=ruijyou(a,s/2,pq)
LET y=ruijyou(a,t/2,pq)

PRINT gcd(MOD(x+y,pq),pq)
PRINT gcd(MOD(x-y,pq),pq)
END

EXTERNAL FUNCTION GCD(a,b) !互除法
DO
    LET r=MOD(a,b)
    IF r=0 THEN EXIT DO
    LET a=b
    LET b=r
LOOP
LET GCD=b
END FUNCTION

EXTERNAL SUB dlog(a,b,N,s,t) !離散対数
LET t=a
LET flag=0
FOR i=2 TO 2*N
    LET t=MOD(a*t,N)
    IF t=b THEN
        PRINT a;b;i;"あった"
        IF MOD(i,2)=0 THEN
            LET flag=flag+1
```

```
            IF flag=1 THEN LET s=i
            IF flag=2 THEN
                LET t=i
                EXIT FOR
            END IF
        END if
    END if
NEXT I
IF flag=0 THEN PRINT a;b;"なかった"
END SUB

EXTERNAL FUNCTION ruijyou(a,r,N) !累乗の計算
LET s=1
FOR i=1 TO r
    LET s=MOD(s*a,N)
NEXT I
LET ruijyou=s
END function
```

```
 314159   16489241   2469134  あった
 314159   16489241   35407576  あった
 8423
 7823
```

以下に，暗号化と復号のコードと出力結果を示す。

```
!ElGamal 暗号
LET p=8423 !安全素数
LET q=7823
LET pq=p*q
LET a=3141593 !公開鍵
LET n=1234567
LET b=ruijyou(a,n,pq)
LET a1=MOD(a,p)
LET a2=MOD(a,q)
LET m1=2 !秘密の乱数
LET m2=3
LET c1=ruijyou(a1,m1,p)
LET c2=ruijyou(a2,m2,q)
LET c=Chinese(c1,c2,p,q) !公開鍵を作成
LET d=ruijyou(c,n,pq)
LET m=Chinese(m1,m2,p-1,q-1) !周期から指数を求める。これで求まらない場合もある。
PRINT m
PRINT ruijyou(a,m,pq);c !一致を確認
PRINT
```

```
CALL dlog(a,c,pq)
CALL dlog(b,c,pq)
CALL dlog(a,d,pq)
CALL dlog(b,d,pq)
print

LET e=Fibonacci(a,b,20,2236067977499,pq)
LET f=Fibonacci(c,d,20,2236067977499,pq)

LET x1=ruijyou(MOD(e,p),m1,p)
LET x2=ruijyou(MOD(e,q),m2,q)
LET x=Chinese(x1,x2,p,q)
PRINT f;x !一致を確認
END

EXTERNAL SUB dlog(a,b,N) !離散対数
LET t=a
LET flag=0
FOR i=2 TO N
    LET t=MOD(a*t,N)
    IF t=b THEN
        LET flag=1
        PRINT a;b;i;"あった"
    END if
NEXT I
IF flag=0 THEN PRINT a;b;"なかった"
END SUB

EXTERNAL FUNCTION ruijyou(a,r,N) !累乗の計算
LET s=1
FOR i=1 TO r
    LET s=MOD(s*a,N)
NEXT I
LET ruijyou=s
END function

EXTERNAL FUNCTION Chinese(a1,a2,p,q) !中国人剰余定理
CALL ExGCD(p,q,s,t,c)
LET x=a1*t*q+a2*s*p
LET Chinese=MOD(x/c,p*q/c)
End function

EXTERNAL SUB ExGCD(a,b,s,t,c) !拡張ユークリッドの互除法（再帰版）
IF b=0 THEN
    LET s=1
```

```
    LET t=0
    LET c=a
ELSE
    LET q=INT(a/b)
    LET r=MOD(a,b)
    CALL ExGCD(b,r,s1,t1,c)
    LET s=t1
    LET t=s1-t1*q
END IF
END SUB

EXTERNAL FUNCTION Fibonacci(a,b,bit,r,n) !漸化式
LET e1=a
LET e2=b
FOR i=1 TO bit
    LET x=MOD(e1*e2,n)
    LET e1=e2
    LET flag=MOD(r,4)
    LET r=INT(r/4)
    IF flag=0 OR flag=1 THEN
        LET e2=x
    ELSEIF flag=2 THEN
        LET e2=mod(x*e2,n)
    else
        LET e2=mod(x*x,n)
    end if
next i
LET Fibonacci=x
END FUNCTION
```

5983833
20408025　8962976

3141593　8962976 なかった
52353760　8962976 なかった
3141593　48352149 なかった
52353760　48352149 なかった

42728613　42728613