Z_2上の多項式を用いたElGamal暗号の作成

Ⅰ 数学的準備

以下で、Z_2は2を法とする整数Zの剰余類である。
そして、2^p-1をメルセンヌ素数として、Z_2を係数とする多項式の、

既約（因数分解できない）多項式　$P(x)=x^p+a_{p-1}x^{p-1}+a_{p-2}x^{p-2}+\cdots+a_0$　を法とする剰余類環Gを考える。ただし、a_iは0か1の値をとり、$a_0=1$である。
$P(x)$が既約のため、ユークリッドの互除法により、0以外の任意の$f(x)$に対して
$g(x)f(x)+h(x)P(x)=1$
となる$g(x),h(x)$が存在する。
このとき、$g(x)=\{f(x)\}^{-1}$であるから、Gは乗法について群になり、Gは体になる。
Gの位数がメルセンヌ素数2^p-1であるため、自明なもの以外に部分群が存在しない巡回群になる。

$$x^p=a_{p-1}x^{p-1}+a_{p-2}x^{p-2}+\cdots+a_0$$

を特性方程式とする数列は、M系列乱数として知られているものである。

$$f(x)=b_{p-1}x^{p-1}+b_{p-2}x^{p-2}+\cdots+b_0,\quad g(x)=c_{p-1}x^{p-1}+c_{p-2}x^{p-2}+\cdots+c_0$$

として、多項式の積$f(x)g(x)$を次のように計算する。
まず、特性方程式は使わずに

$$f(x)g(x)=\sum_{k=0}^{p-1}\left(c_kx^{p-j}(b_{p-1}x^{p-1}+b_{p-2}x^{p-2}+\cdots+b_0)\right)$$

を計算する。この積の計算は、排他的論理和の繰り返しで行われる。すると

$$f(x)g(x)=d_{2p-2}x^{2p-2}+d_{2p-3}x^{2p-3}+\cdots+d_0$$

の形になる。
次に、次数の高い方から低い方に順に

$$x^k=x^{k-p}x^p=x^{k-p}(a_{p-1}x^{p-1}+a_{p-2}x^{p-2}+\cdots+a_0)=(a_{p-1}x^{k-1}+a_{p-2}x^{k-2}+\cdots+a_0x^{k-p})\cdots①$$

を繰り返し用いて、次数をp-1次まで下げる。
また

$$\{f(x)\}^2=(b_{p-1}x^{p-1}+b_{p-2}x^{p-2}+\cdots+b_0)^2=\sum_{i=0}^{p-1}{b_i}^2x^{2i}+\sum_{i\neq j}^{p-1}2b_ib_jx^{i+j}$$

ここで、Z_2では${b_i}^2=b_i$，$2b_ib_j=0$であるから

$$\{f(x)\}^2=\sum_{i=0}^{p-1}b_ix^{2i}$$

であり、これに①を用いて、次数をp-1次まで下げる。$\{f(x)\}^2$の計算時間はp^2ではなくpに比例するので、極めて高速に行われる。

ここで n を任意の自然数とするとき、n を 2 進法で表す。

$$n=(\cdots(((s_0\cdot 2+s_1)\cdot 2+s_2)\cdot 2+s_3)\cdot 2+\cdots)+s_p$$

ただし、s_i は 0 か 1 の値をとり、$s_0\neq 0$ である。

$\{f(x)\}^n$ を求めるには、$n_0=s_0, n_i=n_{i-1}\cdot 2+s_i$ として

$f(x)g(x)$ と $\{f(x)\}^2$ の計算のアルゴリズムを用いて $\{f(x)\}^{n_i}=(\{f(x)\}^{n_{i-1}})^2\{f(x)\}^{s_i}$ を次々に求めていけば $\log_2 n$ のオーダーの計算回数で $\{f(x)\}^n$ が求められる。(バイナリー法)
特に、$x^{mn}=(x^m)^n$ と変形し $f(x)=x^m$ とすれば x^{mn} が求まる。
メルセンヌ素数 $2^{521}-1$ に対応する既約多項式として、例えば $x^{521}+x^{455}+x^{437}+x^{350}+1$　すなわち、特性方程式として、$x^{521}=x^{455}+x^{437}+x^{350}+1$ を用いると、周期は $2^{521}-1$ である。
メルセンヌ素数 $2^{1279}-1$ に対応する既約多項式を用いると、周期は $2^{1279}-1$ であり、メルセンヌ素数 $2^{2281}-1$ に対応する既約多項式を用いると、周期は $2^{2281}-1$ である。

Ⅱ 高速化の工夫

多項式の積の計算を、次のように分割して計算する。(Karatsuba 法)
p を奇数として p=2q-1 するとするとき

$$f(x)=\sum_{i=0}^{2q-1}b_i x^i=\left(\sum_{i=0}^{q-1}b_{q+i}x^i\right)x^q+\left(\sum_{i=0}^{q-1}b_i x^i\right)\quad,\quad g(x)=\sum_{i=0}^{2q-1}b_i x^i=\left(\sum_{i=0}^{q-1}b_{q+i}x^i\right)x^q+\left(\sum_{i=0}^{q-1}b_i x^i\right)$$

とする。

ここで、$\sum_{i=0}^{q-1}b_{q+i}x^i=A_1$, $\sum_{i=0}^{q-1}b_i x^i=A_0$, $\sum_{i=0}^{q-1}b_{q+i}x^i=B_1$, $\sum_{i=0}^{q-1}b_i x^i=B_0$ とおくと

$$f(x)g(x)=(A_1x^q+A_0)(B_1x^q+B_0)=A_1B_1x^{2q}+(A_1B_0+A_0B_1)x^q+A_0B_0$$

$$A_1B_0+A_0B_1=(A_1+A_0)(B_1+B_0)-A_1B_1-A_0B_0$$

であるから、
A_1B_1, A_0B_0, A_1B_0, A_0B_1 の 4 回の乗算を
A_1B_1, A_0B_0, $(A_1+A_0)(B_1+B_0)$ の 3 回に減らすことができる。
このような分割計算を再帰的に k 回行えば、乗算による消費時間は、この分割法を使わないときの、ほぼ $\left(\frac{3}{4}\right)^k$ 倍になる。計算のオーダーは $O\left(n^{\log_2 3}\right)=O\left(n^{1.585}\right)$ である。

521 ビット同士の掛け算を行う場合は、
521 ビット＝512+9 ビット，512 ビット＝2^4×32 ビット
であり、
521×521＝$(512+9)^2$＝512×512+512×9+9×512+9×9＝512×512+512×9+9×521
であるから
512×512 の部分は Karatsuba 法を使い、残りの 512×9 と 9×521 の部分は単純な方法を使うのが、一番効率が良いと思われる。

Karatsuba 法で 512×512 ビットの乗算を 16×16 ビットの乗算まで落とすと計算時間は理論上$\left(\frac{3}{4}\right)^5$となる。

メルセンヌ素数$2^{1279}-1$に対応した既約多項式を用いる場合は、1280 ビットに拡張して、拡張したビットを 0 で埋める。すると、1280＝(5×2^4)×16 となるから、5 分割 1 回と 2 分割 4 回を行ない、16×16 ビットの乗算にすると効率がよい。5 個に分割にするには、次のようにする。

$$f(x)g(x)=(A_4x^{4q}+A_3x^{3q}+A_2x^{2q}+A_1x^q+A_0)(B_4x^{4q}+B_3x^{3q}+B_2x^{2q}+B_1x^q+B_0)=\sum_{i=0}^{8}S_ix^{iq}$$

であり

$S_0=A_0B_0$

$S_1=A_1B_0+A_0B_1=(A_1+A_0)(B_1+B_0)-A_1B_1-A_0B_0$

$S_2=A_2B_0+A_1B_1+A_0B_2=(A_2+A_0)(B_2+B_0)-A_2B_2-A_0B_0+A_1B_1$

$S_3=A_3B_0+A_2B_1+A_1B_2+A_0B_3=(A_3+A_0)(B_3+B_0)-A_3B_3-A_0B_0+(A_2+A_1)(B_2+B_1)-A_2B_2-A_1B_1$

$S_5=A_4B_1+A_3B_2+A_2B_3+A_1B_4=(A_4+A_1)(B_4+B_1)-A_4B_4-A_1B_1+(A_3+A_2)(B_3+B_2)-A_3B_3-A_2B_2$

$S_6=A_4B_2+A_3B_3+A_2B_4=(A_4+A_2)(B_4+B_2)-A_4B_4-A_2B_2+A_3B_3$

$S_7=A_4B_3+A_3B_4=(A_4+A_3)(B_4+B_3)-A_4B_4-A_3B_3$

$S_8=A_4B_4$

$S_4=A_4B_0+A_3B_1+A_2B_2+A_1B_3+A_0B_4$

$\quad=(A_0+A_1+A_2+A_3+A_4)(B_0+B_1+B_2+B_3+B_4)-S_0-S_1-S_2-S_3-S_5-S_6-S_7-S_8$

より、25 回の乗算を 14 回に減らすことができる。計算のオーダーは$O\left(n^{\log_5 14}\right)=O\left(n^{1.640}\right)$である。

実際に、この方法で 1280 ビット同士の掛け算を行い、さらに、16×16 ビットの乗算を工夫することにより、筆算の方法で何も工夫しない時に比べて計算時間が約$\frac{1}{10}$になった。

$\{f(x)\}^n$を求めるとき、n を 2 進展開するが、n を 16 進展開や 32 進展開などして、4 ビット、5 ビットなどと纏めて計算した方が高速に計算できる。その場合、事前に$\{f(x)\}^k$の k が小さい場合について計算しておく必要がある。実装に当たっては、さらに工夫を凝らし高速化した。すなわち、$\{f(x)\}^k$(k=1,3,5,7,…,63)を事前に計算し、これを掛けた。$\{f(x)\}^k$(k=1,3,5,7,…,63)は、先頭のビットと最終ビットが共に 1 である、1 ビット以上 6 ビット以下のビット列に対応する。この工夫により、1279 ビットの場合、乗算の回数は平均で約 215 回であり、1 ビットずつ計算したときの約$\frac{1}{3}$に減らすことができた。(「window method」として知られた方法です。)

window method で、例えば

n=10110100111010000110100011

のときa^nは次のようにします。

窓の大きさが 6bit の場合

a^k(k=1,3,5,7,…,63)を事前に計算しておき

n=101101｜00｜11101｜0000｜1101｜000｜11

と、先頭も最後も 1 である 6bit 以内の列と、0 だけの bit 列に分け

$a^n=(((a^{101101})^{10000000}a^{11101})^{100000000}a^{1101})^{100000}a^{11}$

として計算します。

既約多項式として$x^{1279}+x^{418}+1$を用いたとき、$\{f(x)\}^n$の計算時間は、athlon64 3700+ 2.2Ghz で約4.2m 秒である。また、x^nの計算時間は、約 0.8m 秒である。

メルセンヌ素数$2^{2281}-1$に対応した既約多項式を用いる場合は、2304 ビットに拡張して、拡張したビットを 0 で埋める。すると、2304＝($2^4\times3^2$)×16 となるから、3 分割 2 回と 2 分割 4 回を行ない、16×16 ビットの乗算にすると効率がよい。3 個に分割にするには、次のようにする。

$$f(x)g(x)=(A_2x^{2q}+A_1x^q+A_0)(B_2x^{2q}+B_1x^q+B_0)=\sum_{i=0}^{4}S_ix^{iq}$$

であり

$S_0=A_0B_0$

$S_1=A_1B_0+A_0B_1=(A_1+A_0)(B_1+B_0)-A_1B_1-A_0B_0$

$S_2=A_2B_0+A_1B_1+A_0B_2=(A_2+A_0)(B_2+B_0)-A_2B_2-A_0B_0+A_1B_1$

$S_3=A_2B_1+A_1B_2=(A_2+A_1)(B_2+B_1)-A_2B_2-A_1B_1$

$S_4=A_2B_2$

より、9 回の乗算を 6 回に減らすことができる。計算のオーダーは$O\left(n^{\log_3 6}\right)=O\left(n^{1.631}\right)$である。

Ⅲ さらなる高速化の工夫

「間引き乱数法」とでも名付けたい方法です。手抜き乱数と言われるかもしれませんが。

ビットが多くなるに従って演算が遅くなるが、これをくい止めるには、$\{f(x)\}^n$の計算において、n を安全性を損なわない範囲で特殊な乱数とすることが考えられる。Z_2上の多項式の乗算においては、異なる多項式のの乗算は遅いが、2 乗が極めて速いことに着目すると、n を 2 進展開したとき、0，1 のうち 1 の割合が少なければ少ないほど高速になる。例えば、n を単純な乱数でなく、乱数と乱数の and 演算をしたものとすれば、1 回の and 演算ごとに 1 の数は半減する。n のビット数が増えても、0 の数を増やして 1 の数を一定にしておけば、さほど低速にはならない。ただし、window method の効果は薄れる。

各ビットが 1 である確率を p とすると、各ビットごとのエントロピーは

$$-p\log_2 p-(1-p)\log_2(1-p)$$

である。したがって、1279 ビットのとき、エントロピーは次のようになる。ただし、乗算の回数はバイナリー法で計算したときであり、2 乗は含めない。

確率p	乗算の回数の平均	1ビット当りのエントロピー	総エントロピー	乗算1回当りのエントロピー
1/2	639.5	1	1279.0	2.0
1/4	319.8	0.81	1037.6	3.2
1/8	159.9	0.54	695.2	4.3
1/16	79.9	0.34	431.4	5.4
1/32	40.0	0.20	256.6	6.4
1/64	20.0	0.12	148.5	7.4
1/128	10.0	0.07	84.3	8.4

ここでは$p=\frac{1}{32}$の場合で考えてみよう。

総エントロピーが小さいので、window method の考え方を用いて、乗算の回数は余り増やさずに総エントロピーを増やす工夫をする。

n のビット列に 1 が現れたとき、1 を 10 か 11 に確率$\frac{1}{2}$で置き換えることを考える。1 回の置き換えで 1 ビット増えるので、n から最下位ビットを押し出す。1 回の置き換えで、エントロピーは 1bit 増えて、0.20bit 減るため、差し引き 0.80bit 増えることになる。ただし、11 は$\{f(x)\}^3$に対応するため乗算は 1 回増える。

同様に、n のビット列に 1 が現れたとき、1 を 1000,1100,1010,1110,1001,1011,1101,1111 に確率$\frac{1}{8}$で置き換えることを考える。1 回の置き換えで 3 ビット増えるので、n から下位の 3 ビットを押し出す。このとき、1 回の置き換えで、エントロピーは 3bit 増えて、0.20×3＝0.60bit 減るため、差し引き 2.4bit 増えることになる。したがって、もしも 36 回置き換えれば、総エントロピーは約 86bit 増えて、約 343bit になる。ただし、1100,1010,1110,1001,1011,1101,1111 は、それぞれ$\{f(x)\}^3,\{f(x)\}^5,\{f(x)\}^7,\{f(x)\}^9,\{f(x)\}^{11},\{f(x)\}^{13},\{f(x)\}^{15}$に対応するため、乗算は 7 回増えるが、乗算 1 回当りのエントロピーは約 8bit と大きくなる。

このとき、普通の乱数の場合に較べて、n の総エントロピーは約$\frac{1}{4}$に減るが、乗算の回数も約$\frac{1}{4}$に減り、$\{f(x)\}^n$の計算時間は約 40%に減る。このようにエントロピーを減らしても、総当たり攻撃（ブルートフォース攻撃）に対しては十分すぎる総エントロピー量である。要は、公開鍵暗号としては、既知や未知である効率的な攻撃法に耐性があれば良いのである。ただし、どこまで耐性があるかは不明であるが、どうしても高速化したい場合は検討する価値がある。

Ⅳ 受信者の初期設定

1. Z_2上の既約多項式$P(x)=x^p+a_{p-1}x^{p-1}+a_{p-2}x^{p-2}+\cdots+a_0$を設定する。例えば$P(x)=x^{1279}+x^{418}+1$
2. $m\in Z$（$m<2^p-1$）を決定してx^m計算し、$P(x)$と$\alpha=x^m$を公開する。m が秘密鍵であり、α が公開鍵である。

Ⅳ 送信者の作業

1. 乱数$k\in Z$ $(k<2^p-1)$を生成し、x^kを計算する。さらに、$\alpha=x^m$から$\alpha^k=(x^m)^k=x^{mk}$を計算する。
2. 平文を M とし、x^{mk}をブロック暗号の秘密鍵として M をブロック暗号化したものと、x^kを送信する。

Ⅴ 復号処理

1. x^kと秘密鍵 n から$(x^k)^m=x^{mk}$を計算する。
2. x^{mk}をブロック暗号の秘密鍵として復号する。

Ⅵ デジタル署名

次のような電子署名（Schnorr 署名の一種）が考えられる。

送信者の秘密鍵を n とし、$\beta = x^n$ を公開鍵とする。
送信者は、文書 M に対して乱数$k \in Z \quad (k < 2^p - 1)$を生成し，次の$\gamma$および、文書 M と$\gamma$を合わせたもののハッシュ e=h(M, γ)を計算し、さらにδを計算する。

$$\gamma = x^k$$

e=h(M, γ)

$$\delta \equiv k - n \cdot e \mod 2^p - 1$$

ただし、δの計算における乗法、減法、mod の計算は、普通の整数の四則演算で行う。

(e, δ) が文書 M の署名である。このとき

$$x^\delta \beta^e = x^{k-ne} \cdot (x^n)^e = x^k = \gamma$$

が成り立つ。

送信者は、受信者に文書 M とその署名(e, δ) を送る。

受信者は、v=$x^\delta \alpha^e$ を計算し、これよりハッシュ h(M,v)を求める。e=h(M, γ)と h(M,v)が一致すれば文書 M の署名が確認され、受信者は文書 M が送信者のものと確信できる。
v=$x^\delta \alpha^e$ を計算において、x^δ の計算は高速であり、α^e の計算も、e の桁数が $2^p - 1$ の桁数より小さいため高速に計算できる。

[注]
mod $2^p - 1$ の計算において、次の事が成り立つ。

not を c 言語のビット否定演算とするとき、a+not(a)は、すべてのビットに 1 が立つから

$$a+not(a)= 2^p - 1$$

よって

$$-a \equiv not(a) \mod 2^p - 1$$

また、k≧p のとき

$2^k = 2^{k-p} \cdot 2^p = 2^{k-p} \cdot (2^p - 1) + 2^{k-p}$ より

$$2^k \equiv 2^{k-p} \mod 2^p - 1$$

よって、a の p+1 位以上のビットについて、ビット shift 演算を用いて

$$a \equiv (a>>p) \mod 2^p - 1$$