数式処理 *J.JSSAC (2006)*
Vol. 12, No. 4, pp. 9 - 17

学生論文

数式処理を用いた暗号アルゴリズムの実験的評価

荒井 千里*

筑波大学 図書館情報メディア研究科

概　要

The processing time for public key cryptosystems is currently deemed too long. In this paper we describe the attempt to reduce this processing time by speeding up the prime number generation part of it. We show empirical estimation for typical public key cryptosystems: the RSA, ElGamal and Rabin systems. We implemented these algorithms on a computer algebra system, including the generation of code keys with practical bit length. To speed up the generation of prime number for code keys, we applied trial division and incremental search algorithms, and these approaches proved effective for the RSA and Rabin systems. However, the code keys for the ElGamal system remain difficult to compute in spite of these devices, because it needs a 'safe prime' to generate a code key, whose computation is essentially time-consuming.

1　はじめに

1970 年代半ばまで，全ての暗号のメッセージ交換には秘密鍵が用いられていた．これに対して 1976 年に Diffie と Hellman は公開鍵暗号方式の概念を [3] で発表した．1978 年に R. L. Rivest, A. Shamir，L. Adleman がこの概念を初めて実現し，RSA 暗号を考案した [9]．その後，別の方式として T. ElGamal が ElGamal 暗号 [4] を，RSA 暗号を修正することで M. O. Rabin が Rabin 暗号 [8] を考案した．

本研究では，公開鍵暗号方式の暗号理論を数式処理システム上で実装する．素数生成，べき乗計算等の数論アルゴリズムが公開鍵暗号方式の実装には必要となる．公開鍵暗号方式の効率化を実現するために，これらのアルゴリズムの効率的実装の研究を行った．

2　公開鍵暗号系の暗号方式

本研究では，公開鍵暗号系の RSA 暗号，Rabin 暗号，ElGamal 暗号を取り上げる [7].

2.1　RSA 暗号

1978 年に R. L. Rivest，A. Shamir，L. Adleman の 3 人により考案された世界初の公開鍵暗号である RSA 暗号は，素因数分解の困難性に基づいている．二つの大きな素数 p, q を掛け合わせ $N = pq$ を求めることは簡単だが，N を素因数分解して p と q を求めることは困難であり，「p

*m262@slis.tsukuba.ac.jp

と q が十分に大きければ効率的なアルゴリズムで素因数分解をすることは不可能」とする仮定を素因数分解仮定という．現在，RSA 暗号の安全性は素因数分解仮定に基づいて成立しているが，その同値性は証明されていない．つまり素因数分解が困難であれば，RSA 暗号が安全であるとはわかっていない．RSA 暗号を素因数分解より効率的に解読できるアルゴリズムが存在する可能性がある．

アルゴリズム 1 (RSA 暗号)
(Step 1) 暗号鍵生成

1. ふたつの大きな素数 p, q を生成し，$N = pq$ を計算する．
2. $\gcd((p-1)(q-1), e) = 1$ となる e をランダムに選ぶ．
3. 拡張 Euclid 互除法を $(p-1)(q-1)$ と e に適用し

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

となる d を求める．
4. $P_k = (N, e)$ を公開鍵として公開，d を秘密鍵として保持する．

(Step 2) 暗号化
公開鍵 (N, e) および平文 $m \in \mathbf{Z}_N$ を入力して，暗号文 C を出力する．平文に対し

$$C \equiv m^e \pmod{N}$$

を計算する．
(Step 3) 復号
秘密鍵 d および暗号文 C を入力して，平文 m を出力する．暗号文に対し

$$m \equiv C^d \pmod{N}$$

を計算する．

N を素因数分解できれば RSA 暗号が破れることは容易にわかる．現在，N が 1024 ビット以上であるとき，$N = pq$ の素因数分解が困難であり安全とされている [6].

2.2 ElGamal 暗号

ElGamal 暗号は 1984 年に T. ElGamal によって提案された．x から $a \equiv g^x \pmod{p}$ を計算することは簡単だが，a から x を求めることは困難であり，「p が十分に大きければ効率的なアルゴリズムでこれを解くことは不可能」とする仮定を離散対数仮定という．ElGamal 暗号は離散対数仮定に基づく公開鍵暗号系である．

アルゴリズム 2 (ElGamal 暗号)
(Step 1) 暗号鍵生成

1. $p = 2q + 1$ である p が素数となるように，大きな素数 q 選択する．この p を safe prime という．（注意 3）
2. $\mathbf{Z}_p^*$ の原始元 g を選ぶ．($g^2 \not\equiv 1 \pmod{p}$ かつ $g^q \not\equiv 1 \pmod{p}$ を満たす $g \in \mathbf{Z}_p^*$)
3. $x \in \mathbf{Z}_{p-1}$ をランダムに選び
$$y \equiv g^x \pmod{p}$$
を計算する．
4. $P_k = (p, g, y)$ を公開鍵として公開し，x を秘密鍵として保持する．

(Step 2) 暗号化

公開鍵 (p, g, y) および平文 $m \in \mathbf{Z}_p$ を入力して，暗号文 $C = (c_1, c_2)$ を出力する．

$r \in \mathbf{Z}_{p-1}$ をランダムに選び

$$c_1 \equiv g^r \pmod{p} \qquad c_2 \equiv my^r \pmod{p}$$

を計算する．

(Step 3) 復号

秘密鍵 x および暗号文 $C = (c_1, c_2)$ を入力して，平文 m を出力する．暗号文に対し

$$m \equiv c_2 c_1^{p-1-x} \pmod{p}$$

を計算する．

注意 3

p が safe prime でない場合，原始元の判定に $g \not\equiv 1, \ldots, g^{p-2} \not\equiv 1 \pmod{p}$ のチェックが必要となり効率が悪くなる．

離散対数問題が解ければ，ElGamal 暗号は破られる．そのため，素数 p は離散対数問題が解けないくらい大きくなければならない．現在，素数が 1024 ビット以上のとき，離散対数を求めることが困難であり安全とされている [6].

2.3 Rabin 暗号

§2.1 で前述したとおり，現在，素因数分解が困難であれば RSA 暗号が安全であるとはわかっていない．したがって，RSA 暗号が素因数分解以外のもっと簡単な方法で解読される可能性は否定できない．一方，安全性が素因数分解の困難さと同等であることが証明されている暗号系も存在する．1979 年に M. O. Rabin によって提案された Rabin 暗号は，RSA 暗号に対し修正を行ったもので，解読が素因数分解の困難さと等価であると証明されている．しかし，この方式は一意に解読できないという欠点がある．

記法 4

$N = pq$ であるとき，$x = [a, b]$ によって

$$x \equiv a \pmod{p} \qquad x \equiv b \pmod{q}$$

となる $x \in \mathbf{Z}_N$ を表す．

アルゴリズム 5 (Rabin 暗号)

(Step 1) 暗号鍵生成

1. ２つの大きな素数 p, q を生成し，$N = pq$ を計算する．
2. N を公開鍵として公開し，p, q を秘密鍵として保持する．

(Step 2) 暗号化

公開鍵 N および平文 $m \in \mathbf{Z}_N$ を入力して，暗号文 C を出力する．平文に対し

$$C \equiv m^2 \pmod{N}$$

を計算する．

(Step 3) 復号

秘密鍵 p, q および暗号文 C を入力して，平文 m を出力する．

1. $x^2 \equiv C \pmod{p}$ の解 $x \equiv \pm a \pmod{p}$ を求める．
2. 同様に，$x^2 \equiv C \pmod{q}$ の解 $x \equiv \pm b \pmod{q}$ を求める．
3. $M_1 \equiv a \pmod{p}$ かつ $M_1 \equiv b \pmod{q}$ となる $M_1 = [a, b]$ を中国式剰余定理により求める．同様に

   $$M_2 = [a, -b] \quad M_3 = [-a, b] \quad M_4 = [-a, -b]$$

   を求める．以上４つの M_i が，$x^2 \equiv C \pmod{N}$ を満たし，平文の候補になる．
4. $M_1, \ldots, M_4$ のうち，何らかの冗長性を満たす M_i を平文 m とする．

3 素数生成

現在利用されているほとんどの公開鍵暗号は，素数の数学的性質にその安全性の基礎をおいている．素数生成の主たる部分は素数判定法となる．素数判定法には確定的素数判定法と確率的素数判定法がある．確定的素数判定法では「素数」を判定する．これに対し，確率的素数判定法は「合成数」を判定する．確率的素数判定法で合成数と判定されなかったものは，実際に素数である見込みが高く，このような数を probable prime と呼ぶ．以下で確率的素数判定法を取り上げ，その高速化手法について考える．

3.1 Miller-Rabin テスト

現在，最も一般的に使用されている素数判定法である Miller-Rabin テストについて取り上げる．Miller-Rabin テストは probable prime と判定された数が合成数であるという誤りを起こす確率が確率的素数判定法のなかで最も低く，かつ，計算効率のよい素数判定法となっている [5].

アルゴリズム 6 (Miller-Rabin テスト)

入力：奇数 $n \geq 3$，および繰り返し回数 t

出力：n が素数かに対する答え　prime（素数）または composite（合成数）

1. $n - 1 = 2^s r$ ただし，r は奇数となる s を求める．$i = 0$ とする．

2. $i = t$ ならば，prime と出力して停止する．そうでなければ，$i := i + 1$ とする．
3. $1 \leq a \leq p - 1$ なる a をランダムに選ぶ．これを底と呼ぶ．
4. $y_0 = a^r \bmod n$ を計算する．
5. $y_0 = 1$ または $y_0 = n - 1$ ならば，1 へ戻る．
   そうでない場合は，$j = 1$ から $j = s - 1$ まで，以下を実行する．
   (a) $y_j = y_{j-1}^2 \bmod n$ を計算する．
   (b) $y_j = n - 1$ ならば，1 へ戻る．
6. composite と出力して停止する．

素数判定アルゴリズムは，一般に次の二通りの誤りをする可能性がある．

- 入力 n が素数なのに，composite と出力する．
- 入力 n が合成数なのに，prime と出力する．

これに対し，Miller-Rabin テストについては以下が成り立つ．

- n が素数の場合，composite と出力することはない．
- n が合成数の場合，prime と出力する確率は $(1/4)^t$ 以下である．

3.2 素数生成アルゴリズムの高速化手法

素数生成アルゴリズムの効率を高め計算時間の高速化をはかる手法として，数式処理システム上では以下の手法があげられる．

3.2.1 試行割算法

試行割算法は，素数候補 n を小さい素数で順に割っていくことで，その数が素数であるかどうかを判定する素数判定法である．もし n が $n^{\frac{1}{2}}$ 以下の全ての素数 p で割り切れなければ，n は素数であると判定される．ただし実際の暗号アルゴリズムでは，試行割算法は単独でなく他の素数判定法の前処理部分として素数生成の効率を高めるために使用されている．

試行割算を行う素数の個数の最適値を r として，Random Choice 素数生成法による素数生成の場合には次の評価式が知られている [2].

$$r = \frac{R}{D \log (R/D)} \tag{1}$$

ここで，R は Miller-Rabin テストにおける底 a を 2 に固定した場合のテスト 1 回に要する時間，D は試行割算時の小素数 1 個での除算に要する時間とする．

3.2.2 Random Choice 素数生成法

Random Choice 素数生成法は，ランダムに選んだ奇数を素数の候補として素数判定を適用して素数生成を行う方法である．

アルゴリズム 7 (Miller-Rabin テストを用いた Random Choice 素数生成法)
入力：k（ビット長）
出力：n（k ビットの probable prime）

1. n が"probable prime"と判定されるまで，以下を実行する
   (a) k ビットの奇数 n をランダムに選ぶ.
   (b) n に対して，t 回繰り返し Miller-Rabin テストによる素数判定を行う.
   (c) n が"合成数"の判定ならば (a) に戻る.
2. n を出力

3.2.3 Incremental Search 素数生成法

Incremental Search 素数生成法は Miller-Rabin テストを用いた素数生成法で，試行割算を両者へ適切に組み込んだ場合，Random Choice 素数生成法よりも約 25%速くなることを [2] では示している．このアルゴリズムは，最初にランダムに数を選択した後は，合成数と判定された数に 2 を足し次に判定する数として，判定繰り返し回数の上限に達するか，または probable prime と判定されるまで繰り返していく.

アルゴリズム 8 (Miller-Rabin テストを用いた Incremental Search 素数生成法)
入力：k（ビット長）c（繰り返し回数の上限値）
出力：n（k ビットの probable prime）

1. k ビットの奇数 n をランダムに選ぶ.
2. $i \leftarrow 0$
3. $i \leq c$ の間，以下を繰り返す.
   (a) n に対して t 回繰り返して Miller-Rabin テストによる素数判定を行う.
   (b) n が"probable prime"と判定されれば n を出力．そうでなければ n に 2 を足す.
   (c) $i \leftarrow i + 1$
4. "fail"を出力

4 速度評価

公開鍵暗号の暗号アルゴリズムを数式処理システム上で実装する．そして，暗号鍵生成，暗号化，復号速度を実験的に評価する．実験に用いた計算時間計測環境は次の通りである.

機種	H9000L2000
CPU	PA-8500(440MHz)
OS	HP-UX11.0
使用メモリ	64MBytes
言語	reduce3.7+rlisp88

4.1 暗号鍵生成

4.1.1 素朴な計算法の場合

予備実験として，Random Choice 素数生成法，試行割算なしによる暗号鍵生成の計算時間を計測する．この実験では各暗号方式の暗号鍵生成で素数生成を行っている部分とそれ以外の部分の計算時間の計測を行い，素数生成が暗号鍵生成の中で占めている時間を評価した．暗号鍵は 100 個生成し，p, q は各 156 桁で実験を行った.

RSA 暗号では，素数生成とはアルゴリズム 1–Step1 の 1.1 を，素数生成以外とは 1.2〜1.5 を示している．Rabin 暗号では，素数生成とはアルゴリズム 5–Step1 の 1.1 を，素数生成以外とは 1.2〜1.3 を示している．素数生成には，RSA 暗号，Rabin 暗号の両暗号方式で同一のアルゴリズムを使用しているが，組み込み関数による乱数発生のため結果は一定とはならない．Rabin 暗号の暗号鍵生成では，素数 p, q を生成して $N = pq$ の計算を行っているだけとなり，素数生成以外の計算時間はほぼ 0 に等しくなる．ElGamal 暗号は生成する素数が 310 桁であり，その素数は原始元の効率的な生成のために safe prime となる必要がある．このため RSA 暗号，Rabin 暗号と同一の条件下では計測ができなかった．詳細は後に示す．

表 1: 予備実験　Random Choice 素数生成法による暗号鍵生成時間 (sec)

	暗号鍵生成	素数生成	素数生成以外
RSA 暗号	226.28	226.24	0.03
Rabin 暗号	211.10	211.10	0.00

4.1.2 工夫した計算法の場合

§3.2 素数生成アルゴリズムの高速化手法で挙げた各アルゴリズムを用いて，素数生成部分の高速化をはかる．試行割算を行う小素数の上限の最適値，素数 1 個を生成するために必要となる計算時間，および，素数を生成するまで判定した素数候補の個数を表 2 に示す．これは各桁数で素数をそれぞれ 300 個生成した平均 1 である．試行割り算を行う小素数の上限は，評価式 (1) から求めた値をもとに実験的に最適値を設定した．実験では素数候補の生成法は Incremental Search 素数生成法，素数判定には Miller-rabin テストを用いた．

表 2: 試行割算による素数 1 個の生成時間

桁数	小素数の上限	素数生成時間 (sec)	素数候補判定個数
10 桁	80	0.0079	13.3
50 桁	3000	0.39	52.3
100 桁	5000	2.61	116.7
156 桁	7000	10.04	180.3
200 桁	10000	23.80	238.1
250 桁	15000	46.87	280.2
310 桁	20000	87.48	361.1

これをもとにして暗号鍵生成の計算時間の計測を行った結果が表 3 となる．暗号鍵は各 300 回生成して平均をとった．

Random Choice 素数生成法を用いて試行割算なしで計測を行った予備実験（表 1）と比べ，素数生成の計算時間が約 1/10 となり，暗号鍵生成では素数生成が大部分を占めるため暗号鍵生成

表 3: 暗号鍵生成時間 (sec)

	暗号鍵生成	素数生成	素数生成以外
RSA 暗号	21.60	21.57	0.03
Rabin 暗号	21.46	21.46	0.00

の計算時間も約 1/10 となった.

ElGamal 暗号では計算時間が大きくなるために計測ができなかった．原始元の効率的な生成のために safe prime を求める必要があるが，safe prime の個数は素数よりも少ないため，その生成には素数生成より多くの時間がかかる．なお，調べた限りでは，その分布に関する理論的評価は知られていなかった.

実際に必要となる計算時間の予測を行った実験の結果は表 4 となる．この実験では，Pentium M 1600MHz の環境で各桁数の safe prime を各 300 個生成し，1 個あたりの計算時間の平均を求めた．ただし，310 桁の場合のみ，150 個の結果となっている．100 桁までの safe prime についてはもとの環境で計測可能であり，表 4 では実測値を示している.

表 4: safe prime1 個の生成に要する計算時間

桁数	T_q(sec)	$\#q_i$	T_p(sec)
10 桁	0.0079	18.9	0.15
50 桁	0.39	84.5	33.0
100 桁	2.61	152.5	398.0
156 桁	10.04	262.5	~ 2635.5
200 桁	23.80	363.0	~ 8639.4
250 桁	46.87	405.4	~ 19001.1
310 桁	87.48	464.0	~ 40590.0

T_q 素数 1 個あたりの生成時間．表 2 と共通.
$\#q_i$ safe prime $p(= 2q + 1)$ 生成に要した q の個数.
T_p safe prime 生成時間．予測値は $T_p \sim T_q \times \#q_i$

現在，ElGamal 暗号が安全であるために必要となる素数の桁数は 310 桁とされている．表 4 で示した通り，310 桁の safe prime を 1 個を生成するのに平均約 40590sec(=11.3 時間) かかると予測され，もとの環境では計算時間を計測できなかった.

4.2 暗号化・復号

上記の暗号鍵生成で生成した暗号鍵を用い 2 桁の数字 1 個を 300 回ずつ暗号化・復号した．その平均を表 5 に示す.

表 5: 暗号化・復号計算時間 (sec)

	RSA 暗号	ElGamal 暗号	Rabin 暗号
暗号化	0.90	5.37	0.00
復号	2.74	2.67	0.10

Rabin 暗号の暗号化では，平文 m に対し $C \equiv m^2 \pmod{N}$ の計算を行っている．この操作は実験を行った環境では計測可能時間に満たないため 0.00 の結果となった．

5 まとめ

本論文は著者の卒業論文 [1] を整理したものである．現代暗号の代表的な暗号方式のひとつである公開鍵暗号方式の中から RSA 暗号，ElGamal 暗号，Rabin 暗号を取り上げ，数式処理システム上で各アルゴリズムの実装を行った．素数生成において，試行割算，Incremental Search 素数生成法を取り入れることは有効な高速化手法であった．しかし，safe prime の生成は計算時間が非常に大きいために，これらの方法だけでは効率のよいアルゴリズムにはならなかった．今後，計算時間を短くするための方法として，安全性を損なわずに短い鍵長を実装する方向などが考えられる．

参 考 文 献

[1] 荒井千里: 数式処理を用いた暗号アルゴリズムの実験的評価, 卒業論文, 筑波大学図書館情報専門学群, 2005.

[2] Brandt, J., Damgard, I., and Landrock, P.: Speeding up Prime Number Generation, *ASIACRYPT'91*, *Lecture Notes in Conp. Sci.*, **739**, Springer-Verlag, Berlin and Tokyo, 1993, 440–449.

[3] Diffie, W. and Hellman, M.: New Direction in Cryptography, *IEEE Trans. Information Theory*, **IT-22**(6), 1976, 644–654.

[4] ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Trans. Information Theory*, **IT-31**(4), 1985, 469–472.

[5] 情報処理振興事業協会: 素数生成アルゴリズムの調査・開発, Technical report, CRYPTREC, 2003. http://www.ipa.go.jp/security/enc/CRYPTREC/index.html.

[6] 情報処理推進事業協会, 通信・放送機構: 暗号技術評価報告書 (2001 年度版), Technical report, CRYPTREC, 2002.

[7] 黒澤馨, 尾形わかは: 現代暗号の基礎数理, コロナ社, 東京, 2004.

[8] Rabin, M. O.: Digital Signatures and Public-Key Encryptions as Intractable as Factorization, Technical report, MIT, 1979.

[9] Rivest, R. L., Shamir, A., and Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptography, *Comm. ACM*, **21**(2), 1978, 120–126.