解　説

暗号技術と高速算法†

森　田　　光††

1. まえがき

通信者間の情報伝達を安全に行い，第三者への情報もれを排除する手段として発展してきた暗号は，近年の公開鍵暗号の概念の創出を契機に，理論および実用面から精力的に研究が進められている．

公開鍵暗号の概念は，従来の暗号が必要とした秘密の暗号鍵の配送を不要とするだけではなく，相手確認機能およびデータ確認（署名）機能を新たに生み出した．この新しい機能はディジタル署名として実現され，電子契約・電子決済など多方面での利用が期待されている．

公開鍵暗号の安全性は，離散対数問題または素因数分解問題の困難性と密接な関係がある．安全性の向上のために，公開鍵暗号が扱うデータは500ビットから1,000ビット規模にする必要があり，一般的な計算機が扱う単精度演算の単純な組み合せだけでは高速処理を達成できない．

このため，秘密通信の用途では，通信メッセージを乱数化する秘匿機能に従来どおり秘密鍵暗号を使用し，暗号鍵を通信相手へ配送するのに公開鍵暗号を使うという併用方式が実用的であるとされてきた．しかし，鍵配送機能に限定したとしても，公開鍵暗号を安価な汎用マイクロプロセッサで十分な性能を達成するのは容易ではなく，ハードウェア化も含めた高速算法の研究が必要になってきている．

一方，データ確認機能を実現するディジタル署名の高速アルゴリズムが最近提案され注目されている．このディジタル署名アルゴリズムは，秘匿化とディジタル署名の両機能を実現する代表的な公開鍵暗号のRSA暗号に比べ，署名生成が高速にできる．ディジタル署名は，電子契約や決済などのシステムに組み込まれ，なつ印や署名のような個人対応の処理に利用されると想定できる．しかし，ICカードなどの小型システムでの実現が期待されるが，わずかな計算資源（低速なCPU，少容量のRAM，EEPROM，ROMなど）による高速化は容易ではない．

最近の通信と計算機技術の進歩にともない，高速かつ安全なシステムの重要性が高まっているので，暗号技術への実用的な要求も増している．ここで紹介する高速化，経済化，ならびにICカードのような小型システムへの適用化などの問題は，暗号を実際に使っていくために，早急に技術確立しなければならない課題である．

本稿では，理論面には深入せず，暗号技術全般について，実用面を紹介する．第一に，暗号技術で実現される機能を簡単に説明する．具体的には，秘密通信を実行するための秘密鍵暗号，鍵を安全に配送するための鍵配送法，そしてディジタル署名について述べる（**2.**）．第二に，公開鍵暗号やディジタル署名の実用化にとって必須の，高速算法の最近の話題を取り上げ，剰余乗算とべき乗剰余を扱う（**3.**）．これらの剰余演算を高速かつ小型に実行できる算法が進歩すれば，暗号技術を低コストに実現し，身近に利用できるようになると期待できるからである．

本稿では，簡単な解説を与えるだけなので，公開鍵暗号の概念など，さらに興味のある読者は他書を参照されたい（たとえば1)，2)の文献．公開鍵暗号の研究から生まれたゼロ知識証明は本学会誌小特集[3]にある）．算法は各参照文献を当たられたい．なお，本稿と関連する解説として，特集「数論アルゴリズムとその応用」が前号（93年2月号）に掲載されたので，あわせてご一読いただきたい．

† Cryptography Techniques and High-Speed Computation by Hikaru MORITA (NTT Network Information Systems Laboratories).

†† NTT 情報通信網研究所

2. 暗号技術で実現される機能

2.1 秘密通信

図-1 に秘密通信の概要を示す．送信者のメッセージ（平文）pを安全でない（盗聴されるかもしれない）通信路を経由して受信者に伝えるとき，メッセージを暗号化して秘密を守って通信する．現代の暗号では，暗号化と復号化の手順は公開にしても，メッセージと暗号文の組合せを決める鍵を秘密にすれば第三者からそのメッセージの秘密を守ることができる暗号が研究されている．

暗号の手順を公けにすると，処理を実行する装置を共通化できるので経済的であり，共通の方式を使っていれば予定してなかった相手とも秘密通信できるなどの利点がある．反対に，使っている暗号の解読方法が見つかるとその危険の及ぶ範囲は大きくなる．一方，手順を公けにしても，解読方法が長い間見つからなければ，多くの人が解読を試みたと期待される分だけ信頼できると考えることができる．

暗号は以下の2種に分類される．暗号鍵と復号鍵を同一の値（$K_e=K_d$）とし，両方とも秘密とする秘密鍵暗号（または，慣用暗号，対称鍵暗号）と，暗号鍵と復号鍵を別の値（$K_e\neq K_d$）にし復号鍵を秘密とし，暗号鍵を公開する公開鍵暗号（または非対称鍵暗号）との2種である．

秘密鍵暗号は通信相手へ秘密の鍵（秘密鍵）を安全に配送しなければならないが，公開鍵暗号は秘密鍵の配送が不要である．

一般に，公開鍵暗号は，素因数分解問題や離散対数問題の困難性に基づいて構成されるので，500〜1,000ビット長の演算を必要とし，高速化が課題となっている．市販の専用LSIでは，公開鍵暗号は秘密鍵暗号のおよそ千分の1の処理速度であり，公開鍵暗号はディジタル通信の基本単位である64 kb/s程度でも専用LSIが必要である．

そこで，主に経済的理由から，メッセージを直接的に乱数化するのに秘密鍵暗号を用い，受信者に秘密鍵暗号の鍵を送る（鍵配送）ために公開鍵暗号を用いるというのが，実用的な形態だと一般に考えられている．

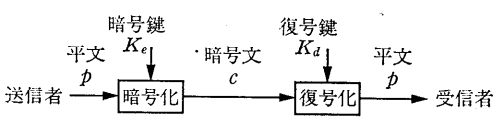


図-1 秘密通信の概要

代表的な秘密鍵暗号のアルゴリズムとしては，IBM開発の米国標準であるDES (Data Encryption Standards)[4]，NTTのFEAL (Fast Data Encipherment Algorithm)[5]，ならびに日立のMULTI 2 (Multimedia Encryption 2)[6]などがある．

処理速度は，汎用マイクロプロセッサ上のソフトによる実現例では，暗号化処理速度700 kb/sが達成できる（FEAL-8暗号で，i 80386を使用，16 MHzクロックの場合）．また，LSIによる高速な実現例としては，96 Mb/sの処理速度をもつ暗号専用のLSIがある（DES暗号，FEAL暗号とも市販品がある）．さらに高速な例としては，暗号LSIを複数個並列実行する暗号装置の試作がある[7]．このように，秘密通信の要求処理速度は，すでに秘密鍵暗号によりほとんどカバーされている（**図-2** 参照）．応用製品もすでに開発されていて，暗号機能が付加された，データモデム，FAX，電話などが市販されている．

2.2 秘密鍵暗号の鍵を配送する（鍵配送）

秘密鍵暗号では，あらかじめ通信相手の受信者に送信者の秘密の暗号鍵（秘密鍵）を配送する必要がある．鍵配送には，公開鍵暗号を鍵配送に用いる方法と，秘密鍵暗号を用いる方法がある．

公開鍵暗号を用いる方法では，配送すべき秘密鍵を受信者の公開鍵を使って暗号化して送るので，安全でない通信路を使っても鍵配送できる．これが，前節で紹介した秘密鍵暗号と公開鍵暗号を組み合わせる場合に使われる，公開鍵暗号の機

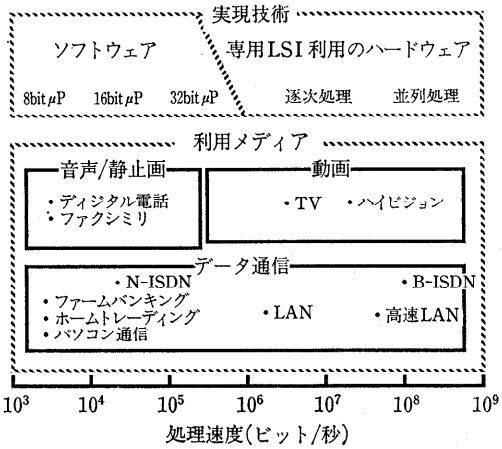


図-2 秘密通信の利用分野と実現技術

能である．

経済的な理由から全てを秘密鍵暗号で実行したい場合がある．以下では，紹介されることが比較的に少ない，鍵配送に秘密鍵暗号を用いる方法を説明する[8]．あらゆる2者間通信の鍵配送に使える公開鍵暗号に比べ，用途がセンタ端末間に限定される点に特徴がある．

多くの端末を相手するセンタが，メッセージ暗号用の秘密鍵（データ鍵）配送用に，あらかじめ全ての端末に対し，端末ごとに異なる秘密鍵（端末鍵またはセッション鍵）を物理的に安全に設定しているとする．センタは，全ての端末鍵のデータベース化をすべきであるが，オンライン処理が増大する．そこで，**図-3** に示すように，端末鍵 K_i を端末鍵生成関数 θ により生成するとし，端末に端末鍵 K_i を設定しておき，センタは必要に応じて相手の端末鍵 K_i を端末鍵生成関数 θ により生成する．そして，実際の秘密通信に使う秘密鍵をその端末鍵 K_i で暗号化して配送する．センタは，端末鍵をリアルタイムに生成する必要があるが，端末鍵のデータベース化は必要なくなる．

端末鍵生成関数 θ は，秘密であれば何でも良さそうだが，端末の盗難も想定して K_i と T_i が分かっても，θ と秘密の端末鍵生成変数 P との類推を困難とするために，秘密鍵暗号を利用して θ を定義する．この結果，個々の端末の端末鍵が漏れても，センタの θ と P とが漏れない限り，この鍵配送を使うシステムは安全である．しかし，仮に秘密の θ や P が漏洩すると，センタと端末からなるシステム全体が危険になる短所がある．したがって，センタの安全性の確保が容易で，比較的小規模なシステムで簡便に鍵配送したいときに向いている．

以上の2種の鍵配送法のほかにも，公開鍵配送法（文献2）第9章参照），個々の ID 対応に鍵を生成する ID ベース法（文献1）第4章参照）などがある．それぞれ，長所欠点があるので，用途にあった方法の選択が必要である．

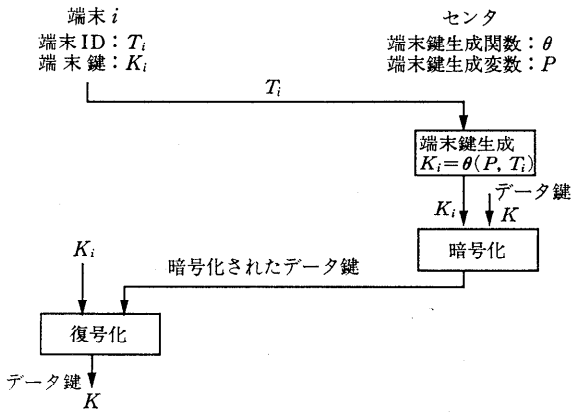


図-3　秘密鍵暗号を使った鍵共有法

2.3　電子的ななつ印（ディジタル署名）

メッセージの正当性を検証する技術としてディジタル署名がある．これは，文書に確認印を押す機能を電子的に実現する技術に相当する．

図-4 にディジタル署名の構成を示す．メッセージ p に対して，署名者は，秘密の署名鍵 K_s で署名処理をし，署名 s を生成する．これが印に相当する．検証者は，検証鍵 K_v で検証処理をし，署名 s とメッセージ p とが対応すれば正当と認め，逆に対応しなければ不正とする．なお，メッセージ長は一定とは限らないから，署名処理の入力でハッシュ関数を使ってメッセージ長を揃える．

ディジタル署名が，メッセージ p と署名 s の対応関係を保証できるのは，署名鍵を持っている者だけが正当な署名を生成でき，署名鍵を持っていない他人にはその署名を作り出せない性質があるためである．一方，公開の検証鍵があれば，署名の正当性はだれでも検証できる．

ディジタル署名によれば，メッセージ対応に印を生成する機能のほかに，相手確認機能を実現できる．単純な方法は，検証相手から乱数を送ってもらい，それに対する署名を送り返して実現できる．

公開鍵暗号の暗号化と復号化の順序を変えることができれば，復号化処理を署名処理に，暗号化処理を検証処理に対応させ，ディジタル署名を実現できる．このような公開鍵暗号に RSA 暗号があるので，RSA 暗号はディジタル署名の機能を包含していると言える．

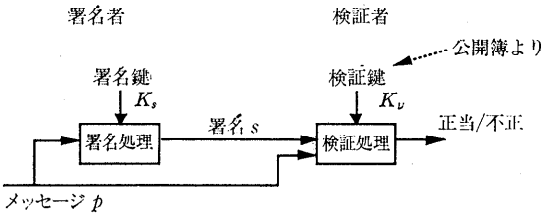


図-4　ディジタル署名の概要

最近，署名生成では，RSA 暗号より演算量が少なくて高速なディジタル署名アルゴリズムが提案されている．米国 NIST (National Institute of Standards) が米国標準に提案している DSA (Digital Signature Algorithm)[9] は，IC カード上で比較的高速にディジタル署名ができることを特徴にしている．同様に，日本では ESIGN (Efficient Digital Signature Scheme)[10] がある．

ディジタル署名により，メッセージ，署名，そして検証鍵に対応関係があることを説明できても，確かにX氏，あるいは，Y氏が関係して署名が生成されたとするには，人と検証鍵との対応関係を確認する枠組が必要である．また，仮にこの問題が解決されても，署名を生成するのは，実際のところ IC カードなどの物であって本人ではないから，本人自身が署名したかは確認できない．この問題の解決には個人識別技術など他の技術が必要になりそうである．

3. 高速算法における最近の話題

公開鍵暗号やディジタル署名のアルゴリズムでは，扱うデータサイズが 500～1,000 ビット長の演算が必要なので，高速化は重要な課題である．特に，IC カード上の実現が期待されるディジタル署名では，小型化も重要な課題である．

アルゴリズムのほとんどは剰余演算を用い，とりわけ剰余乗算とべき乗剰余（または剰余累乗）が重要なので，これらの最近の話題を取り上げる（なお，ここでは，整数演算のみを扱い，有限体上の演算は扱わない）．

3.1 剰余乗算

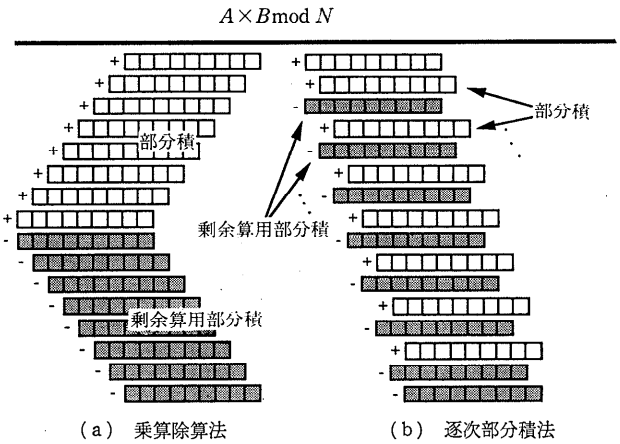
剰余乗算は $A\times B \bmod N$ のように表現され，法 N 上の乗算とも言う．剰余乗算を実行するアルゴリズムには，A と B の積を求めた後，法 N の剰余を求める方法（乗算除算法）と，乗算をしながら割算する方法（逐次部分積法）の2通りがある（**図-5** 参照）．

前者は，単純に，積 $A\times B$ を法 N で割って余りを取り計算できるが，長いデータ長同士の除算は時間がかかるので，なんとかなくしたいという動機で研究されてきた（ここでは言及しないが，長いデータ同士の除算でも，最上位桁同士の除算を実行して効率的に処理する方法はある[12]）．

後者は，積 $A\times B$ の部分積 $A\times B_i$ (B_i は B の一部の桁の値）を大きいものから生成し，法 N の倍数を引いて上位桁を消していくという計算を繰り返す方法で，演算処理幅を少なくしたいという動機で研究されてきた．

剰余テーブル法と Montgomery 法は前者の代表例であり，後者には Brickell 法と MY 法などがある．

剰余テーブル法は，**図-6** に示すように，あらかじめ大きな2のべき乗の剰余を求めて置き，法 N より大きな積 $A\times B$ の上位の各桁を，法 N より小さい下位の剰余の加算に置き換えて計算する．上位の各桁に対応する剰余は互いに独立に参照できるから，参照された剰余同士を並列に加算でき，高速化を達成できる．剰余テーブル法は，鳥居らの発表[13]後，国内で活発に研究された．剰余テーブルは必ずしも，N より大きい上位の全



注) 図中の四角1個でバイトなどの演算単位を表現している．
A, B, N は四角8個分の値を例とする．

図-5 剰余乗算

1. 準備

$t_i \leftarrow 2^i \bmod N$

$2n>i\geq n$ なる各種 i に対応する値 t_i を，あらかじめ剰余テーブルに蓄積する．

2. 手順

$x\leftarrow A\times B$.
For $i=2n-1$ downto $i=n$
  If $x_i=1$, then $x_i\leftarrow 0$ and $x\leftarrow x+t_i$.
$x\leftarrow x \bmod N$.
return(x).

注) $x\triangleq(x_{2n-1}, x_{2n-2}, \cdots, x_1, x_0)_2$
n は N のサイズを表す．$n\triangleq\lfloor\log_2(N)\rfloor+1$

図-6 剰余テーブル法

桁に対応して必要ではなく，並列度を犠牲にすればテーブルを小規模にできるなどの各種提案が報告されている．

Montgomery 法は，**図-7** に示すように，演算対象の数 A, B を X, Y のように R 倍した剰余に変換し，法 N の割算の代りに $ZR^{-1} \bmod N$ を実行し，その演算が2のべきの数 R ($=2^n$, n は N のサイズ以上の整数) による剰余または割算(図-7 式(2)の右辺参照) で実行できることを示した[14]．剰余演算 $\bmod R$ は2進数表現されている値の下位 n ビット分をそのまま剰余の結果とし，除数 R の割算は R 以上の桁を結果とすれば良い．法 N による割算を不要とする R の演算への置換は，中国剰余定理（後述）から導かれる．当初，法 N による割算が不要で計算機向きの R の演算が利用できるので顕著な高速化が期待されたが，それほど利用されなかった．その後，バイトなどの小さな2のべき単位演算の組合わせに分解する改良がなされ，もとの長いデータ同士の乗算2回分相当まで演算量が削減されてから[15]，注目されるようになった．Montgomery 法は，準備段階で，拡張ユークリッド互除法に基づく N' の生成と，R 倍するなどの前後の変換処理は依然必要だが，このユークリッド互除法の除算は不要である．この方法は，前後の変換処理が相対的に多い $A \times B \bmod N$ の回数がわずかなときは効率的ではない．しかし，指数部が法 N 程度のべき乗剰余 ($a^b \bmod N$) は，変換後の乗算部の繰返しだけで実行できるので，前処理や後処理が無視できるほど少なくなり，効率的になる．

Brickell 法は，図-5 (b)の演算を部分積と除算について1ビットごとに実行する[16]．比較的単純な構成なため，LSI などのハードウェアに利用された，その後，各種の変形がなされた[17]．剰余乗算は，法 N のサイズ n に比例する回路量のハードを逐次利用して実現される．これは，サイズ n 同士の乗算を実行するのに，$O(n^2)$ の回路量の並列乗算機か，$O(n)$ の直列乗算機かのハード化が考えられるが，剰余乗算機は，演算サイズが大きいので直列的な構成が選択されたことによる．この Brickell 法を，バイトなどを処理単位とする計算機のソフトウェア向きに変形したのが MY 法である．

MY 法は部分積と除算を計算機の演算単位（1バイト，2バイトなど）に実行するソフトウェア指向の手法である[18]．**図-8** に処理の詳細を示す．C が処理途中の剰余であり，最終段で計算結果となる．大きいデータ同士の乗算は，演算単位の小さい乗算と加算を組み合わせて容易に実現できる．しかし，大きい値同士の除算は簡単でないため，ここでは，2バイト割る1バイト（演算単位が1バイトのとき）の除算を利用する．部分的な除算を実行するので近似による誤差が生じるが，その誤差が演算量を増大させるケースを除く目的で，Step 7 が実行される．平均的に，元の長

1. 準備
   $RR^{-1}-NN'=1$ なる N を求める．
2. 変換
   $X \leftarrow AR \bmod N$
   $Y \leftarrow BR \bmod N$
3. 乗算
   $Z \leftarrow XY$　　(1)
   $Z \leftarrow ZR^{-1} \bmod N = \dfrac{Z \bmod NR + (ZN' \bmod R)N}{R} \bmod N$　　(2)
4. 逆変換
   $A \times B \bmod N \leftarrow ZR^{-1} \bmod N$ (式(2)を再び利用)

図-7 Montgomery 法

Step 1. $C \leftarrow 0$.
　　$i \leftarrow n-1$.
Step 2. $C \leftarrow r \times C + A \times B_i$.
Step 3. $q \leftarrow \left\lfloor \dfrac{C_{n+1} \| C_n}{N_{n-1}+1} \right\rfloor$.
Step 4. $C \leftarrow C - r \times q \times N$.
Step 5. $q \leftarrow \left\lfloor \dfrac{C_{n+1} \| C_n}{N_{n-1}+1} \right\rfloor$.
Step 6. If $q=1$ then $C \leftarrow C - r \times N$.
　　If $q=2$ then $C \leftarrow C - r \times (2N)$.
Step 7. If $C_n \| C_{n-1} \geq (r-1)(N_{n-1}+1)$ then
　　$C \leftarrow C - (r-1) \times N$.
Step 8. $i \leftarrow i-1$.
　　If $i \geq 0$ then goto Step 2.
Step 9. If $C < N$ then return.
Step 10. $q \leftarrow \left\lfloor \dfrac{C_n \| C_{n-1}}{N_{n-1}+1} \right\rfloor$.
Step 11. $C \leftarrow C - q \times N$.
Step 12. If $C < N$ then return.
Step 13. $C \leftarrow C - N$.
　　Goto Step 12.

注)
- r は基数で，1バイト単位なら $r=2^8$
- B は $B \triangleq (B_{n-1}, B_{n-2}, \cdots, B_1, B_0)_r$, N は $N \triangleq (N_{n-1}, N_{n-2}, \cdots, N_1, N_0)_r$, C は $C \triangleq (C_{n+1}, C_n, C_{n-1}, \cdots, C_1, C_0)_r$ で表現される．
- $X \| Y \triangleq r \times X + Y$
- $\lfloor X \rfloor$ は X を越えない最大の整数．
- n は N のサイズを表す．つまり，$n \triangleq \lfloor \log_2 N \rfloor + 1$

図-8 MY 法

いデータ同士の乗算2回分の演算量で実行できるので，前述の Montgomery 法と同程度の演算量である．プログラム作成者の技量に左右されるので，単純な優劣比較はできないが，演算開始前と終了後に変換処理がない点と，演算の作業メモリ量が約半分で済む点から，MY 法は，少なくとも IC カードなどの小型システムなどに有効であると考えられる．

3.2 べき乗剰余（剰余累乗）

べき乗剰余（剰余累乗）は，$a^b \bmod N$ で表される演算である．べき乗剰余は自乗の剰余乗算 $x^2 \bmod N$ および一般の剰余乗算 $x \times a \bmod N$ なる剰余乗算を組み合わせて実行される（**図-9**）．

仮に，512 ビットの演算をする場合，指数 b の2進数表現 $(b_{511}, b_{510}, \cdots, b_1, b_0)_2$ における1の個数を $\gamma(b)$ とするとき，自乗の剰余乗算に 511 回，一般の剰余乗算に $\gamma(b)-1$ 回の演算が必要で，平均 766 回の剰余乗算が必要となる．

最近，ディジタル署名 DSA の提案と合わせて注目されているのが，累乗テーブル法[19),20)]である．ここでは，要点のみを簡単な例を使って紹介

```
x←1
For i=n−1 downto 0
    x←x×x mod N
    If b_i=1 then x←x×a mod N
return(x)
注) b≜(b_{n−1}, b_{n−2}, ⋯, b_1, b_0)_2
    nはbおよびNのサイズを表す．
    n≜⌊log_2(N)⌋+1
```

（上位桁の指数部から $a^b \bmod N$）

図-9 べき乗剰余

```
1. 準備（以下の値を累乗テーブルに蓄積する）
    t_1←a mod N
    t_2←a^{2^m} mod N
    t_3←a^{2^{2m}} mod N
      ⋮
    t_{l−1}←a^{2^{(l−2)m}} mod N
    t_l←a^{2^{(l−1)m}} mod N
2. 手順
    x←1
    For i=m−1 downto i=0
        x←x×x mod N
        x←x×∏_{j=0}^{l−1} t_{j+1}^{b_{jm+i}} mod N
    return(x)
注) b≜(b_{n−1}, b_{n−2}, ⋯, b_1, b_0)_2
    nはNのサイズを表す．n≜⌊log_2(N)⌋+1
    mは単位区間幅，lは区間の全個数．n≜lm
```

図-10 累乗テーブル法

$a^b \bmod pq \Rightarrow (a^b q' \bmod p)q + (a^b p' \bmod q)p \bmod pq$

ただし，拡張ユークリッドの互除法を用いて，あらかじめ $p'p+q'q=1$ なる p', q' を求めておく．

（$a^b \bmod pq$ の場合）

図-11 中国剰余定理

する（**図-10**）．指数部を単位区間幅 m に分け，とびとびの区間幅を飛び越す単位に a のべき乗剰余のテーブルを用意する（図-10 の準備）．そして，とびとびの櫛状に指数データをとり，その指数の値に合わせて，剰余テーブルの値を組み合わせるのである．手順では，$t_{j+1}^{b_{jm+i}}$ なる記述をしているが，2進数表現では b_{jm+i} は0または1なので，$b_{jm+i}=1$ で t_{j+1} を，$b_{jm+i}=0$ で1を意味する．

たとえば，$b=(101110)_2=(46)_{10}$ を，$l=3$, $m=2$ で実行するとき，累乗テーブルとして，法 N 上で，$t_3=a^{2^4}$, $t_2=a^{2^2}$, $t_1=a$ を用意し，$i=1$ のとき，$x=1\times t_1\times t_2\times t_3=a^{1+4+16}$ とし，x を自乗して，$x=(a^{21})^2$ としてから，$i=0$ のとき，$x=x\times 1\times t_2\times 1=a^{42+4}=a^{46}$ を得る．

剰余テーブルを作る手間を考えないとき，512ビットの処理を8ビット区間で処理するとき，7回の自乗の剰余算と，$\gamma(b)$ の剰余乗算の手間で処理できる．仮に，b の半分のビットが1の場合，トータルで 263 回の剰余乗算回数で済む．Brickell らは，2進数以外の基数による拡張を行っており，さらに高速処理ができるとしている[20)]．本手法は，a と N が固定ならば，高速化が達成できる．

最後に中国剰余定理を紹介する．**図-11** では，二つの素数 p, q の合成数 pq を法とする場合の算法を示している．左辺を右辺に置き換え，並列化が可能なハードで最大8倍程度の高速化が，逐次処理のソフトで最大4倍程度の高速化が見込める．図-11 では p と q の2個に分解した例を示したが，この古くから知られる定理はもっと一般に使え，巨大な数を計算するのに，互いに素な多くの素数の剰余演算に分解して並列処理し，後でまとめあげることに利用できる．この定理を積極的に応用する技術として剰余数系（Residue Number Systems）があるが，あまり利用されてこなかった．

4. あとがき

本稿では，実用的な側面から，秘密通信を実行するための秘密鍵暗号，秘密鍵を安全に配送するための鍵配送法，そしてディジタル署名を説明した．また，今後，重要になる，高速算法における最近の話題を取り上げ，剰余乗算とべき乗剰余の各種算法について述べた．

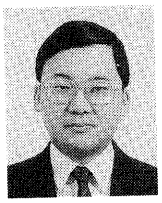
従来あまり扱われることがなかった公開鍵系の各種方式に多用される2種の剰余演算に対して，高速化小型化を目指した研究を進め，暗号技術をより身近で実用的にしたいものである．本稿がその一助になれば幸である．

謝辞　本稿に対してご助言いただいた，NTT情報通信網研究所の宮口庄司,岡本龍明,太田和夫,藤崎英一郎，阿部正幸の各氏に感謝いたします．

参考文献

1) 辻井重男・笠原正雄編著：暗号と情報セキュリティ，昭晃堂（平成2年）．
2) 池野信一・小山謙二：現代暗号理論，電子情報通信学会（コロナ社）（昭和61年）．
3) 太田和夫編：小特集「ゼロ知識証明とその応用」，情報処理，Vol. 32, No. 6, pp. 642-681（平成3年6月）．
4) Federal Information Processing Standards: Data Encryption Standard, FIPS-PUB-46 (Jan. 1977).
5) 宮口，栗原，太田，森田：FEAL 暗号の拡張，*NTTR&D*, Vol. 39, No. 10, pp. 1439-1450（平成2年10月）．
6) 宝木，佐々木，中川：マルチメデイア向け高速暗号方式，情報処理学会，マルチメディア通信と分散処理研究会報告，No. 40-5（昭和61年）．
7) Morita, H. and Yamane, M.: Hardware Approach to Fast Encipherment Processing and Its Implementation, *IEICE Trans.*, Vol. E 74, No. 8, pp. 2143-2152 (Aug. 1991).
8) Miyaguchi, S.: ID Based Key Sharing System Managed by Network, *Proceedings of ICIN*, pp. 89-94 (Mar. 1989).
9) The American National Standards Institute: Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry, Part 1: The Digital Signature Algorithm (DSA), Working Draft, X 9.30-199 x. (Feb. 1992).
10) 岡本，藤岡，岩田：高速ディジタル署名方式 ESIGN, *NTT R&D*, Vol. 40, No. 5, pp. 687-696（平成3年5月）．
11) 楊：IC カード用剰余演算アルゴリズム，信学技報，ISEC 92-16, pp. 11-15（平成4年8月）．
12) Knuth, D. E.: Section 4.3.1, *The Art of Computer Programming 2nd ed.* (*Seminumerical Algorithms*), Addison-Wesley (1981).
13) 鳥居，東，秋山：RSA 分割計算処理の一検討，1986年暗号と情報セキュリティワークショップ，WCIS 86-4, pp. 15-17（昭和61年8月）．
14) Montgomery, P. L.: Modular Multiplication without Trial Division, *Math. Comput.*, Vol. 44, No. 170, pp. 519-521 (1985).
15) Dussè, S. R. and Kaliski, B. S. Jr.: A Cryptographic Library for the Motorola DSP 56000, *Advances in Cryptology-EUROCRYPT '90, Lecture Notes in Computer Science 473*, pp. 230-244, Springer-Verlag (1991).
16) Brickell, E. F.: A Fast Modular Multiplication Algorithm with Application to Two Key Cryptography, *Advances in Cryptology-CRYPTO '82*, pp. 51-60, Plenum (1983).
17) Brickell, E. F.: A Survey of Hardware Implementations of RSA, *Advances in Cryptology—CRYPTO '89, Lecture Notes in Computer Science* 435, pp. 368-370, Springer-Verlag (1990).
18) Morita, H. and Yang, C.-H.: A Modular-Multiplication Algorithm Using Lookahead Determination, *IEICE Trans. Fundamentals*, Vol. E 76-A, No. 1, pp. 70-77 (Jan. 1993).
19) 藤原：暗号アルゴリズムと計算量，bit, Vol. 17, No. 8, pp. 954-959（昭和60年8月）．
20) Brickell, E., Gordon, D. M., McCurley, K. S. and Wilson, D.: Fast Exponentiation with Precomputation, *EUROCRYPT '92 Extended Abstracts*, pp. 193-201 (May 1992).

（平成4年9月17日受付）



森田　光（正会員）

昭和57年北海道大学大学院修士課程（電子工学専攻）修了．現在，NTT 情報通信網研究所に所属．情報セキュリティの研究開発に従事．電子情報通信学会，IEEE，国際暗号学会（IACR）各会員．