暗号の中の数学

– 計算可能性について –

澤田秀樹

はじめに

インターネット (Internet) という言葉はもうすっかり私たちの日常生活に定着している。この世界中の大学や研究機関, 企業等の内部ネットワークを互いに結ぶ巨大なネットワークは, 世界 150 か国以上にわたって 2000 万台以上のコンピュータが関係しているのにもかかわらず, 特定の管理者によって運営されているわけではないという特徴をもっている。このような, 技術的な約束ごとと少しずつ整備されつつある法律のもとでなんとか運営されている, ネットワーク上を第三者によって改変や盗聴が可能なデジタル情報が行き交っているわけだから当然そこにはさまざまな問題が生じている。

今日ここに来られた皆様には暗号理論がデジタル情報の保護に有効であることを新聞や雑誌ですでに何回か目にされ, 公開鍵暗号とか素因数分解とかいった暗号理論や初等整数論の用語に親しまれた方も多いとおもう。この講演ではまず

- 暗号について概説し, つぎに
- その基礎にある「計算する」ことについて考え, さらに
- 数ある暗号の中から RSA 暗号や ElGamal 暗号といった公開鍵暗号をとりあげて

説明したい。

1 暗号とは何か

「暗号 (cryptography) とは情報の交換や記憶集積に際して, 第三者にそれが知られることや, あるいはさらに当事者をも含めて故意による情報の改変を防ぐために, その対象となる情報を秘匿する技術や方法のことである。このような暗号について研

究する数理情報科学を暗号理論 (cryptology) と呼ぶわけであるが, その中には暗号の組立 (encryption), 翻訳 (decryption), 解読 (cryptanalysis) や暗号鍵の管理 (key management) や暗号発信者等の認証 (authentification) などが含まれる。」と言ったりすると, なんだかとても難しいことのように聞こえるかも知れない。

要するに暗号と言うのは, 情報を伝えたり記録したりするときにその内容が第三者にはわからないようにする手段のことであり, もっと広く解釈して内容が不明でその解読には特別な方法を用いるものも指す。ではさっそく暗号の実例をあげよう。

[**実例**] 転置式暗号 (Transposition cipher)

原文に現れるビット列[1] や文字列の順序を入れかえて暗号化する方法を転置式暗号と呼ぶ。この方式ではたとえば次のレールフェンス暗号 (Rail fence cipher) が有名である。

YAMAGATA UNIVERSITY

Y G U E T

A A A A N V R I Y

M T I S

YGUETAAAANVRIYMTIS

鍵は柵の深さすなわち 3 である。

次のようにしても簡単に文字の順序を入れ換えることができる。

[**実例**] 転置式暗号

原文:「 数学の期末試験には関数の最大最小値問題が出そうだね」を
5 文字ずつ改行して書いて

数学の期末
試験には関
数の最大最
小値問題が
出そうだね

[1] ビット (bit, Binary Digit) : 0 あるいは 1 のいずれか一方のこと.
バイト (byte) : 一まとまりのビットの列, 通常は 8 個のビットを 1 単位とする。

としてからこれを縦に読んで
暗文：「数試数小出学験の値そのに最問う期は大題だ末関最がね」
とすればよい。これを解くには再び 5 文字ずつ改行して書いて

数試数小出
学験の値そ
のに最問う
期は大題だ
末関最がね

を縦に読めば良い。

この暗号は使用される言語の特性を調べることにより解読できる ([18] p.59)。
[実例] 換字式暗号 (Substitution cipher)
原文に現れるビットや文字を他のビットや文字に書き換えたり, あるいはいくつかのビットや文字をまとめてブロックごとに他のビット列や文字列に書き換えて暗号化する方法を換字式暗号と呼ぶ。

たとえば文中のアルファベットを K 文字づつ前へずらして (Z まで来たら一回りして A に戻る) 書き換えていく換字式暗号があるが, 鍵が $K = 3$ のときはジュリアスシーザー (Julius Caesar 100-44 B.C.) が使用したためシーザー暗号 (Caesar cipher) と呼ばれる。

シーザー暗号

$K = 3$

O L Y M P I A D
R O B P S L D G

上のシーザー暗号をもう少し発展させると次のような暗号が得られる。
[実例] 換字式暗号
暗号化したい文章たとえば
「秋になったら芋煮会をしましょうか」
AKINI NATTARA IMONIKAIWO SHIMASHOUKA に対して
なにかある本から一節たとえば「古池やかわず飛び込む水の音」を引用して

```
AKININATTARAIMONIKAIWOSHIMASHOKA (原文)
HURUIKEYAKAWAZUTOBIKOMUMIZUNOOTO (鍵)
IF.............................P (暗文)
```

とするのである。ここでの鍵の役割は, アルファベットの表

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

に基づいて, H が 8 だから A を $1+8=9$ 文字目の I に U が 21 だから K を $11+21-26=6$ 文字目の F に.....O が 15 だから A を $1+15=16$ 文字目の P に書き換えることを表している。

[実例] 換字式暗号
戦国時代の武将上杉謙信 (1530 - 1578) は座標式暗号と呼ばれる次のような換字式暗号を使ったと言われる ([7] I, p34)。

れ	く	ふ	ゆ	の	き	あ	↘
ゑ	あ	や	ら	よ	ち	い	つ
ひ	さ	ま	む	た	り	ろ	れ
も	き	け	う	れ	ぬ	は	な
せ	ゆ	ふ	ゐ	そ	る	に	く
す	め	こ	の	つ	を	ほ	み
ん	み	え	お	ね	わ	へ	へ
	し	て	く	な	か	と	し

この表を使うと原文：「川中島に行かむ」は

暗文「きしきへのしきしくしふれあくあつきしゆれ」

となる。

これらの暗号も使用される言語の特性を調べることにより解読できる ([18] p.83)。
[実例] 換字式暗号
コード (code) はコードブックを鍵とする特別な換字式暗号である。例えばコンピュータ内部では A, B, C,... は次のような数値として扱われている。

アスキーコード
ASCII code

(American Standard Code for Information Interchange)

A	B	C	.....	X	Y	Z
65	66	67	.....	88	89	90

I	A	M	A	B	O	Y
73 32	65	77	32	65 32	66	79 89 46

[実例] 点暗号
これまでの例とはまったく異なるが, 文字のそばに針等で小さな印を付けその印をたどると通信文がでてくるという, 点暗号と呼ばれるものもある。たとえば
「お父さん お母さん, お げ んきですか？市内はも 一 毎日 む し暑くて困ります。部活が終わるお盆には そ ちらへ帰ります。ふ るさ と がなつかしいです。」とすれば
「ゲームソフト」を催促していることになる。

1.1 近代暗号

第二次世界大戦以前と以後の1970年代半ばまでの暗号は, 近代暗号と呼ばれていて, 以下のような特徴を持っている。

1. 暗号の作り方も鍵も秘密にする。
2. 暗号の安全性を言語の統計的特徴を基に議論する。
3. 暗号を使用するのが主に軍事と外交である。

前節で紹介した転置式暗号, 換字式暗号やコードを複雑に組み合わせた暗号を使用していたが, どの様な場合も作り方と鍵の秘密を守ることが最も重要な課題であった。([7], I) に基づいて2,3の例をあげてみよう。
[実例] 日露戦争日本海海戦で哨戒艦信濃は「タ」という1文字で「敵艦見ゆ」と伝えることになっていたし, 太平洋戦争開戦時の「新高山登れ」は「真珠湾を攻撃せよ」であった。

[実例] 1942年頃の陸軍の第一戦部隊では
000 0, 001 アイ, 002 極力, 003 河 (川), 004 上等兵, 005 アテ, 006 命令,.....,064 通信所,.....
といった1000語ほどの表に基づいて文章を数値に変換し, それをさらに暗号化していた。

[**実例**] 一方外交用コードとしては先ほどの上杉謙信が使用したのと同じ種類の座標式暗号が使われていたが, 次の 1931 年頃の外務省の暗号はアメリカに解読されていた。

↗	ew	fo	gc	gu	hy	if	in
ak	相成	内話	的確	.	.	.	
av	立場	.	.	.			
ba	.	.	.				
cc	.	.	.				
.							

これらの暗号にはすべて一般的な解読方法が見つかっている。情報理論の一般論から暗号は鍵が長いほど安全であり, その一方暗文が長いほど元の正しい原文を一つに限定できる可能性の高いことが知られている。しかしバーナム暗号 (Vernam cipher) と呼ばれる乱数式換字暗号は唯一無条件に安全な暗号である。

[**実例**] バーナム暗号 (Gilbert Vernam, 1917) この暗号は 1 回限りの使い捨て鍵 (One time pads) を使用するのが特徴である。たとえば原文が "hello" であるとき, 1 回限りの使い捨て鍵 K として完全にランダムで長さが原文と同じ文字列たとえば "iwpbu" をとりシーザー暗号と同じような方法を使って

$$\begin{array}{ccc} hello & \longmapsto & qbbnj \\ iwpbu & & \end{array}$$

ただし

$$\begin{array}{ccc} h & \longmapsto & q \\ i & & \text{(h から 9 番目の文字)} \end{array}$$

と暗文 "qbbnj" に変換する。 鍵 "iwpbu" がわかっていれば原文 "hello" は暗文 "qbbnj" より直ちに得ることができるが, この鍵についての情報がなければどんな 5 文字の文字列も鍵となる可能性があるので, 以下のようなことも起こってしまい, 決して正しく翻訳されることはない。

$$\begin{array}{ccc} peace & \longmapsto & qbbnj\ . \\ awake & & \end{array}$$

1.2 現代暗号

さて1970年代後半から現在までの暗号は, 現代暗号と呼ばれているが, 以下のような特徴を持っている。

1. 暗号の作り方は公開され鍵のみを秘密にする。
2. 暗号の安全性は計算量理論を基に議論する。
3. 暗号を使用するのが軍事と外交だけではなく, 一般社会において民間で使われる。
4. 公開鍵暗号 (public-key system) が使用される。

この現代暗号を代表するものにデータ暗号化規格 (DES)[2] や RSA 暗号がある。

DES 暗号 (Data Encryption Standard) は転置式と換字式を組み合わせた混合式暗号と呼ばれるもので, その基本的な考え方は１９４０年代の終わり頃シャノン (Chaude Shannon) によって提案され, その後１９７０年代に IBM によって開発された。標準暗号, データ暗号規格として１９７７年米国商務省標準局によって米国連邦政府機関の標準として公開されたものである。さらにこの DES 暗号は標準化の為に, そのアルゴリズムが完全に公開されると言う当時としては画期的なものであった。これは暗号の安全性をアルゴリズムの秘匿性よりも鍵の秘匿性におくという考えに基づいている。アルゴリズムの公開については今も問題点が多く指摘されているものの, この考え方は暗号の商用化を促進するうえで, 重要な役割を果たした。

一方 RSA 暗号というのはいくつかある公開鍵暗号を代表する暗号である。では公開鍵暗号とは何だろう。

定義 1.1 次の 5 つの集合 $\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}$ の組を暗号系 (cryptosystem) とよび $\mathcal{S}$ で表す。すなわち

$$\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}),$$

(i) 平文 (plaintext) の集合, $\mathcal{M}$.
(ii) 暗文 (ciphertext) の集合, $\mathcal{C}$.
(iii) 鍵 (key) の集合, $\mathcal{K}$.
(iv) 組立関数 (enciphering transformation) の集合, $\mathcal{E} = \{E_K \mid K \in \mathcal{K}\}$,

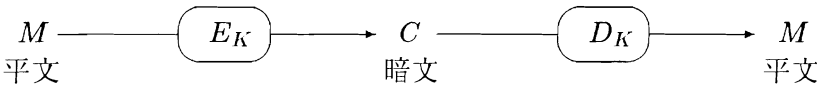
$$E_K : \mathcal{M} \longrightarrow \mathcal{C}, \text{ ただし } K \in \mathcal{K}$$

(v) 翻訳関数 (deciphering transformation) の集合 $\mathcal{D} = \{D_K \mid K \in \mathcal{K}\}$,

$$D_K : \mathcal{C} \longrightarrow \mathcal{M}, \text{ ただし } K \in \mathcal{K}$$

[2] この暗号についてはすでに鍵の大きさを現在の計算機の能力が越えているので次世代の規格が検討されている。

である。ただし $\mathcal{M}$ と $\mathcal{C}$ は長さ有限な記号列よりなる有限集合で互いに等しい数の元を含む。$\mathcal{M} = \mathcal{C}$ と仮定することが多い。E_K $(K \in \mathcal{K})$ は $\mathcal{M}$ から $\mathcal{C}$ への全単射, すなわち 1 対 1 上への写像で, E_K の逆写像は D_K である。



この"現代の"暗号系には

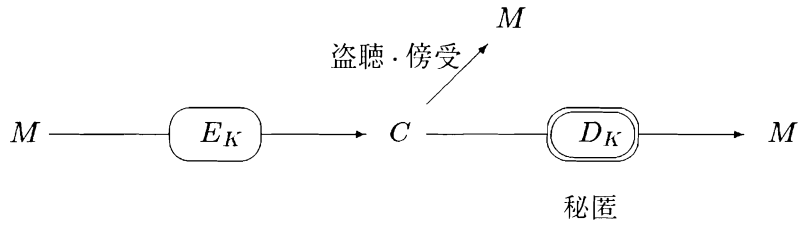
1. どの鍵に対しても組立関数, 翻訳関数ともに効率的に計算できる。
2. 使用方法が易しい。
3. 暗号系の安全性が鍵を秘匿することのみによっていて, 組立関数や翻訳関数の計算法を秘匿することにはよらない。

ことと言った 3 つの一般的要件が求められている。さらに以下のような守秘性 (secrecy) や正当性 (authenticity) に関する要請もある。

守秘性に関しては:

1. 暗号の解読を試みようとして, たとえ暗文やさらに暗文とそれに対応する平文の組を手にいれても, それから翻訳関数 D_K を体系的に決定することが暗号解読者 (cryptanalyst) には計算的に実行不可能であること。
2. 入手した暗文 C から対応する平文 M を体系的に決定することが暗号解読者には計算的に実行不可能であること。

守秘性の確保には翻訳関数を秘匿すれば良いことになる。

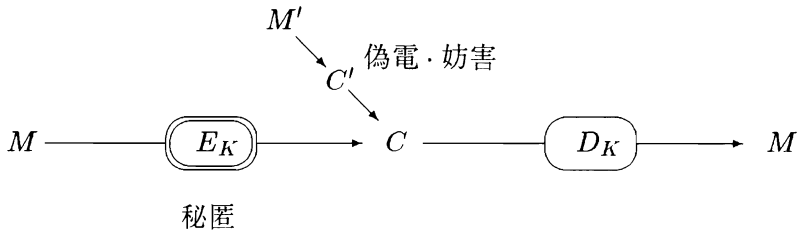


正当性に関しては:

1. 暗号解読者が, たとえ暗文やさらに暗文とそれに対応する平文の組を手にいれても, それから組立関数 E_K を体系的に決定することが計算的に実行不可能であること。

2. 翻訳したときに, その暗号系において, 正当な平文 M' となる暗文 C' を体系的に見つけることが暗号解読者には計算的に実行不可能であること。

であり, この正当性の確保には組立関数を秘匿すれば良いことになる。



暗号系 $(\mathcal{M},\ \mathcal{C},\ \mathcal{K},\ \mathcal{E},\ \mathcal{D})$ を使って相互に通信するとき, すでに述べたように古代から 20 世紀半ば頃までは, 組立関数 E_K, 翻訳関数 D_K ともに秘匿するのが常識であった。暗号通信が少数の当事者間で交わされる特殊なものであった場合はこれでも良いのだが, 簡単な計算からわかるようにこのような暗号系では n 人の間の交信に

$$ {}_nC_2 = \frac{n(n-1)}{2} $$

個以上の鍵が必要となってしまう。これではたった 1000 人の当事者間の通信にも鍵は 499500 個以上必要となる。

しかし W.Diffie と M.Hellman は公開鍵暗号系と呼ばれる新暗号系, New Directions in Cryptography. *IEEE Trans. on Info. Theory Vol. IT-22(6) (Nov. 1976) 644-654* を提唱し, n 人の間の暗号通信が

$$ 2n \quad \text{個} $$

の鍵でできることを示した。この公開鍵暗号系の実現にとってなくてはならないのがいわゆる一方向関数 (one-way function) と呼ばれる関数である。

定義 1.2 集合 A から B への 1 対 1 上への写像 f が擬一方向関数 (pseudo one-way function) であるとは A の元 $x \in A$ が与えられたときには $f(x) \in B$ は容易に計算できても, 逆に $f(x)$ から x を求める計算が容易ではないことを言う。すなわち $f(x)$ の計算法で多項式時間内のものがあっても $f^{-1}(x)$ の計算法で多項式時間内におさまるものは知られていないような関数のことである。

擬一方向関数は擬を省略して単に一方向関数と呼ばれることが多い。

注意 1.3 (i) 現在 "$x \in A$ が与えられたときに $f(x)$ の計算は多項式時間内にできても, $f^{-1}(x)$ の計算が多項式時間内にできない" ことが数学的に証明されているような関数は見つかっていない。

こういった "真に数学的な一方向関数は存在するであろう" というのが $\mathbf{P} \neq \mathbf{NP}$ 予想である。この問題について深く知りたい読者は [2] 等を参照してほしい。

(ii) 擬一方向関数は経験上よく知られている。解くのは難しいが, 与えられた答が正しいかどうかは比較的簡単に確かめられる, といったことはしばしば経験することである。たとえば与えられた自然数を素因数分解することは, 得られた素因数分解が正しいかどうか検算することよりも難しいことは誰もが経験したことであろう。

(iii) 日常生活でも「覆水盆に帰らず」「落花枝に帰らず破鏡再び照らさず」といった故事に擬一方向関数の考え方を見つけることができる。

定義 1.4 (公開鍵暗号系) $\mathcal{M} = \mathcal{C}$ なる暗号系 $(\mathcal{M},\ \mathcal{C},\ \mathcal{K},\ \mathcal{E},\ \mathcal{D})$ において, 鍵の空間 $\mathcal{K}$ がユーザ達の集合 $A,\ B,\ C, ...$, すなわち

$$\mathcal{K} = \{A,\ B,\ C, ...\}$$

であって, 各ユーザ A が一方向関数から作られた組立関数 E_A 及び翻訳関数 D_A の組を所有し E_A は公開登録し D_A は他に公開せず秘匿して, しかも公開された組立関数からその翻訳関数を体系的に決定することが計算的に実行不可能で, A からおくられた暗文は A の翻訳関数でしか容易には翻訳できないとき, このような暗号系を公開鍵暗号系という。

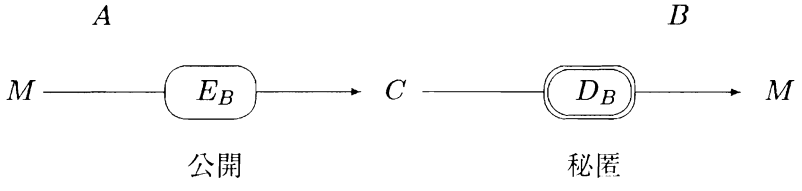
公開鍵暗号系では E_A から D_A を得ることは容易ではないと仮定しているので, この 2 つを区別して, E_A を公開鍵 (public key), D_A を秘密鍵 (private key) と呼ぶ。

公開鍵暗号系に対して組立関数 (公開鍵) E_A と翻訳関数 (秘密鍵) D_A が本質的に同じで, すなわち一方から他方が容易に得られて, その安全性を鍵の秘匿性に大きく依存している, 従来から使われてきた暗号系を慣用暗号系 (conventional cryptosystem) あるいは対称暗号系 (symmetric cryptosystem) という。公開鍵暗号系では守秘性と正当性を以下のように実現する。

守秘性を保つには:

ユーザ A はユーザ B に平文 $M \in \mathcal{M}$ を送るとき B の公開鍵 E_B を使って暗文 $C = E_B(M)$ を B に送れば良い。B はこの暗文を自分自身の秘密鍵 D_B を使って解くことになるが, それができるのは B のみなので守秘性は保たれることになる。

$$D_B(C) = D_B(E_B(M)) = M$$

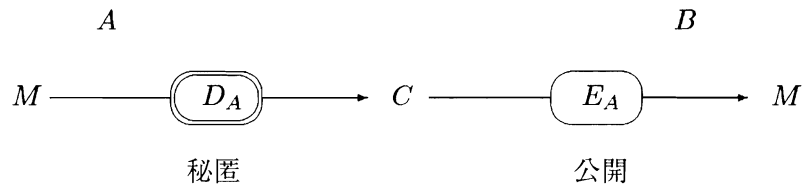


どのユーザも B の公開鍵を知っていて暗文 C のかわりに別の平文 M' から作られた暗文 $C' = E_B(M')$ を B に送ることができるので上の方法では暗文 C の発信者は確かに A であるという正当性は主張できない。

正当性を主張するには:

A は自分自身の秘密鍵 D_A を使って暗文 $C = D_A(M)$ を B に送れば良い。B は A の公開鍵 E_A を使って M を求めるわけであるが, E_A で復号できる暗文を作れるのは A しかいないので, A は自分が正当な発信者であると主張できる。

$$E_A(C) = E_A(D_A(M)) = M$$



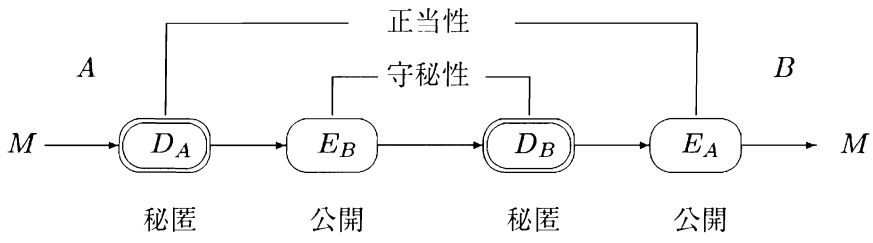
どのユーザも A の公開鍵 E_A を知っていて平文 $M = E_A(C)$ を知ることができるので上の方法で秘密を守ることはできない。

公開鍵暗号系において守秘性と正当性を同時に実現するには発信者 A 受信者 B のそれぞれが二重の変換をしなければならない。A が平文 M を B に送りたいときは, まず M を A 自身の秘密鍵 D_A で変換し次にその結果を B の公開鍵 E_B で暗号化し

$$C = E_B(D_A(M))$$

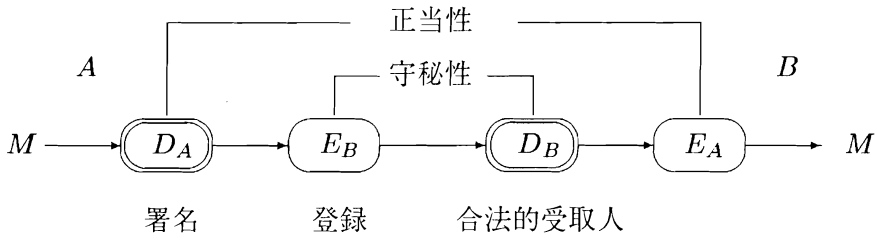
を B に送る。B は初めに自分の秘密鍵 D_B, 次に A の公開鍵 E_A を使って M を求める。A の公開鍵を使うことにより正当性が確かめられる。

$$E_A(D_B(C)) = E_A(D_B(E_B(D_A(M)))) = E_A(D_B(M)) = M$$

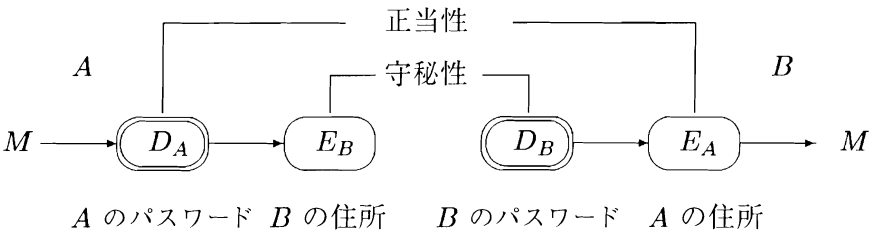


公開鍵暗号系と良く似た考えかたを日常の書留郵便や電子メールのなかに見いだすことができる ([10])。

[**実例**] A から B に書留郵便を送るとする。A は手紙に署名 (捺印) し郵便局で登録した後, それを B に送ればよい。この場合 A の署名が A の秘密鍵 D_A, B の住所が B の公開鍵 E_B に対応する。



[**実例**] A から B に電子メールを送るとする。A は手元の計算機システムに自分のパスワードでログインし, B にメールを送る。この場合 A のパスワードが A の秘密鍵 D_A, メールリストに公表された B の住所が B の公開鍵 E_B に対応する。ただし, これは公開鍵暗号の考えかたを説明するための例であり, 実際の電子メールシステムは安全とはいえない。



以上が公開鍵暗号系の概念であるが, W.Diffie と M.Hellman がこの暗号系を発表したとき, それを非常に高く評価したのが G.J.Simmons (Sandia National Laboratories) だと言われている (W.Diffie, The National Security Establishment and the Development of Public-Key Cryptography. *Designs, Codes and Cryptography, 7 (1996) 9-12* 参照のこと)。彼が公開鍵暗号系を評価したのはそれが核兵器の管理と深く関わってくるからであった。すなわち現地司令官が大統領から重大な命令を受けた場合に, 確かにその命令が大統領からのものであると確認できる. 通信手段がどうしても必要になるからである。

このように一方向関数の存在が証明できないにも関わらず, 応用上公開鍵暗号系はきわめて重要である。正当性を証明する問題は通信だけではなくコンピュータシステムを不正な進入者から守る為にも研究されているが, 現在よく話題になる電子マネーも公開鍵暗号の応用である。

ではこの公開鍵暗号系という"都合のよい話"を実現するにはどうしたらよいのだろうか?

2 NP 問題について

この節では [2] に基づいて計算機科学の基礎理論から, 一方向関数の存在証明に密接に関連する NP 問題, すなわち非決定性計算モデルにより多項式ステップで解くことのできる問題について概説する。NP 問題を知ることは現代暗号を理解する上で大切である。計算モデルと呼ばれる仮想コンピュータを定義し計算できるとは何かについて考えよう。

2.1 計算モデル

定義 2.1 記憶領域を無限に使えるコンピュータを考え, 計算モデルと呼ぶ。小数は浮動小数点表示によって自然数に関する演算でできるので, この計算モデルでの演算は自然数のみに限り, 以下の条件を仮定する。

(i) 一定の長さのビット数ごとの記憶領域とのアクセスが 1 ステップでできる。
(ii) 四則演算, シフト, 判断などの基本命令も 1 ステップでできる。
(iii) (i) で定められた長さを越えるビット数の演算は 1 ステップではできない。

この計算モデルで, 定まったビット数を越えるデータを扱う場合は, 複数の定まったビット数を使用する多倍長形式と呼ばれるものを用いる。この場合四則演算等は 1 ステップではできず, データの長さに応じたステップ数 (計算時間) がかかるとする。計算量をはかる場合は計算時間の定数倍は無視する。

計算モデル上のプログラミング言語 $\mathcal{L}$ を以下のように定める。

定義 2.2 $\mathcal{L}$ は { 代入文, `while` 文, `if` 文, 入力文, `yes` 文, `no` 文 } で構成される。変数や定数は計算モデルによって定められたビット数で表現され, 配列変数も使用できる。`x,y,z` を変数, `a` を定数, `s,s1,s2` を 1 個以上の実行文の列, T(`c`) を条件式 `c` の判定に要するステップ数, T(`s`) を `s` の実行に要するステップ数とする。それぞれの実行文は以下の形式, 意味, ステップ数を持つ。

代入文

形式	意味	ステップ数
`x=a`	定数 `a` の値を `x` に代入する	$O(1)$ ステップ

`x=y+z`
`x=y-z`
`x=y*z`
`x=y div z`
`x=y mod z`
	`y` と `z` の四則演算等の値を `x` に代入する	$O(1)$ ステップ

`while` 文, `if` 文

形式	意味	ステップ数
`while c do s end`	条件式 `c` が真の間 `s` を繰り返し実行する	$r(T(c)+T(s))$ ステップ, ただし r は繰り返しの回数
`if c then s1 else s2 endif` `if c then s1 endif`	条件式 `c` が真なら `s1` を, 偽なら `s2` を実行する	`c` が真なら $T(c)+T(s1)$, 偽なら $T(c)+T(s2)$ ステップ

入力文

形式	意味	ステップ数
`input x`	1 個のデータを変数 `x` に読み込む	1 ステップ

`yes` 文, `no` 文

形式	意味	ステップ数
`yes`	入力に対し `yes` と答えて停止する	1 ステップ
`no`	入力に対し `no` と答えて停止する	1 ステップ

注意 2.3 上の表中の記号 $O(1)$ は計算に要するステップ数が 1 の定数倍であることを意味する。 入力データに多倍長形式も含めるのでこのように書く。

代入文中の演算の順序は括弧により指定できる。括弧の内部等特に指定されていない場合は `*`, `div`, `mod` を `+`,`-` に優先させる。 文には空文も許し, 実行文の列 `s` を使った手続き文:

`procedure` 手続き名 `;` `s` `endprocedure`

も書けることとする。

定義 2.4 x を変数, y を変数または定数とする。 このとき`x==y`, `x!=y`, `x>y`, `x>=y`, `x<y`, `x<=y` を基本条件式といい, これらの基本条件式を `and`, `or`, `not` の演算子で結合したものを条件式と呼ぶ。 基本条件式 c の判定に要するステップ数 $T(\mathtt{c})$ は 1 とし, 条件式 `c1,c2` に対しては

$$\begin{aligned} T(\mathtt{c1\ and\ c2}) &= T(\mathtt{c1}) + T(\mathtt{c2}) \\ T(\mathtt{c1\ or\ c2}) &= T(\mathtt{c1}) + T(\mathtt{c2}) \\ T(\mathtt{not\ c1}) &= T(\mathtt{c1}) + 1 \end{aligned}$$

であるとする。

2.2 問題とアルゴリズム

定義 2.5 問題とはある関数が与えられた入力に対して `yes` あるいは`no` と出力するかどうかを決定することをいい, その出力を計算する方法をアルゴリズムという。 また入力する記号列の長さを入力のサイズと呼ぶ。

定義 2.6 ある問題に対してそれを解くアルゴリズムが存在するとき, その問題を計算可能 (computable) または決定可能 (decidable) といい, アルゴリズムが存在しないとき計算不能 (uncomputable) または決定不能 (undecidable) という。

[**実例**] ナップザック問題 (Knapsack problem)
入力: 自然数の列 $\{S, M_1, M_2, ..., M_n\}$
決定すること: $0, 1$ を要素とする列 $\{b_1, b_2, ..., b_n\}$ で

$$S = b_1 M_1 + b_2 M_2 + \cdots + b_n M_n$$

となるものが存在するか。

[**解答例**] 入力のサイズ n が小さければ容易に解ける。
$n = 1$: $S = 0$ または $S = M_1$ ならば `yes`, $S \neq 0$ かつ $S \neq M_1$ ならば `no`.
$n = 2$: $S \in \{0, M_1, M_2, M_1 + M_2\}$ ならば `yes`, $S \notin \{0, M_1, M_2, M_1 + M_2\}$ ならば `no`.
入力 $\{21, 9, 7, 23, 2, 3\}$ に対しては $21 = 9 + 7 + 2 + 3$ が見つかるから`yes`, 一方入力 $\{22, 9, 7, 23, 2, 3\}$ に対しては式

$$\begin{aligned} 22 &= b_1 9 + b_2 7 + b_3 23 + b_4 2 + b_5 3 \text{ を変形して} \\ 22 - b_4 2 &= b_1 9 + b_2 7 + b_3 23 + b_5 3 \end{aligned}$$

とすれば左辺は $22 \geq 22 - b_4 2 \geq 20$ で右辺は 23 以上か 19 以下だから no となる。ナップザック問題は効率が悪くてもそれを解くアルゴリズムを考えられるので計算可能であるといえる。

計算不能な問題の例としてつぎのようなものがある。詳しくは [2] 等を参照してほしい。
[実例] プログラム停止問題 (Programme terminality problem)
入力: プログラム P と入力データ x
決定すること: 入力 x に対して $P(x)$ は停止するか, ただし $P(x)$ はデータ x がプログラム P に入力された状態を示す。

2.3 多項式時間と指数時間

定義 2.7 入力のサイズ n に対してある定数 k が存在して $O(n^k)$[3] ステップ以下で解くアルゴリズムが存在する問題のクラスを決定性多項式時間 (deterministic polynomial time) あるいは単に多項式時間といい P で表す。またある定数 k が存在して $O(2^{n^k})$ ステップ以下で解くアルゴリズムが存在する問題のクラスを指数時間 (exponential time) といい EXPTIME で表す。

定義より P $\subset$ EXPTIME であるが

$$\mathrm{P} \neq \mathrm{EXPTIME}$$

であることが知られている。
[実例]「ナップザック問題
入力: 自然数の列 $\{S, M_1, M_2, ..., M_n\}$
決定すること: 0,1 を要素とする列 $\{b_1, b_2, ..., b_n\}$ で

$$S = b_1 M_1 + b_2 M_2 + \cdots + b_n M_n$$

となるものが存在するか。」は 2^n 通り考えられる列 $\{b_1, b_2, ..., b_n\}$ のすべてについての S の値を調べて解くことができるから EXPTIME に属することになる。

定義 2.8 自然数 n のビット数を n の入力のサイズと呼び size n と書く。

[3] n^k の正定数倍以下におさまるステップ数を $O(n^k)$ と書く。

補題 2.9 自然数 n に対してそのサイズは

$$\mathrm{size}\, n = 1 + [\log_2 n]$$

で与えられる。ただし [ ] はガウス記号と呼ばれ小数 $x_0.\, x_1x_2...(\geq 0)$ の整数部分を表す, すなわち $[x_0.\, x_1x_2...] = x_0$ である。

[証明] いま

$$n = a_t 2^{t-1} + a_{t-1}2^{t-2} + \cdots + a_2 2 + a_1$$

となっていたとする。ただし $a_i = 0$ または 1 で $a_t \neq 0$ とする。

$$2^{t-1} \leq n \leq 2^t - 1 < 2^t$$

だから $t - 1 \leq \log_2 n < t$ となり

$$\mathrm{size}\, n = 1 + [\log_2 n]$$

であることがわかる。

[証明終]

自然対数を ln とすれば

$$\mathrm{size}\, n = 1 + [\log_2 n] = 1 + \left[\frac{\ln n}{\ln 2}\right]$$

であり小数 x に対して $[x] \leq x < [x] + 1$ が成り立つから, 自然数 n の入力サイズ $\mathrm{size}\, n$ は $O(\ln n)$ であると考えてよいことになる。2 進数の加法, 乗法, 除法はそれぞれ

$$\begin{array}{rr} & 11001 \\ + & 1111 \\ \hline & 101000 \end{array}$$

$$\begin{array}{rr} & 11001 \\ \times & 1101 \\ \hline & 11001 \\ & 11001 \\ & 11001 \\ \hline & 101000101 \end{array}$$

$$
\begin{array}{rl}
 & 1101 \\
\hline
111 &) 1011101 \\
 & -111000 \\
\hline
 & 100101 \\
 & -\ 11100 \\
\hline
 & 1001 \\
 & -\ 111 \\
\hline
 & 10
\end{array}
$$

のようにするから

命題 2.10 自然数 m, n の加法に要するステップ数は

$$O(\max(\ln m, \ln n))$$

乗法に要するステップ数は

$$O(\ln m \ln n)$$

除法に要するステップ数は $m \geq n$ と仮定して m を n で割った商を q とすれば

$$O(\ln m \ln q)$$

となる。

それでは実例によってステップ数を計算してみよう。はじめに変換器と呼ばれる計算モデルを定義しておく。

定義 2.11 (変換機) 決定性計算モデルとその上のプログラミング言語 $\mathcal{L}'$ を組にしたものを変換機と呼ぶ。ただし $\mathcal{L}'$ は { 代入文, `while` 文, `if` 文, 入力文, 出力文, `halt` 文 } で構成され出力文, `halt` 文 以外の実行文の形式, 意味, ステップ数は定義 2.2 と同じものとする。

出力文, `halt` 文

形式	意味	ステップ数
`output x`	変数または定数 `x` の値を出力	1 ステップ
`halt`	計算を停止する	1 ステップ

[**実例**] ユークリッドの互除法 (Euclidean algorithm)
ユークリッドの互除法とは, 2 つの自然数 a, b の最大公約数 (a, b) を求める計算法であり, 次の 2 つの補題より導かれた。

補題 2.12 a, b を整数とする。 $b>0$ かつ $b \mid a$ ならば

$$(b,a)=b.$$

補題 2.13 a, b を整数とする。 $b>0$ かつある整数 q, c があって

$$a=qb+c \quad \text{ならば} \quad (a,b)=(b,c).$$

どんな初等整数論の教科書にも書いてあるが復習すれば以下のようになる。

a, b を自然数とし, $a \geq b$ と仮定する。

$$a=r_0, \quad b=r_1$$

とおき a を b で割った商, 余りをそれぞれ q_0, r_2 とする。

$$a=r_0=q_0r_1+r_2 \quad 0 \leq r_2 < r_1$$

$r_2 \neq 0$ の場合には r_1 を r_2 で割り商を q_1 余りを r_3 とする。

$$r_1=q_1r_2+r_3 \quad 0 \leq r_3 < r_2$$

$r_3 \neq 0$ の場合には r_2 を r_3 で割り商を q_2 余りを r_4 とする。

$$r_2=q_2r_3+r_4 \quad 0 \leq r_4 < r_3$$

$r_4 \neq 0$ の場合にはさらに同じような計算を行う。この結果は

$$r_1 > r_2 > \cdots > r_k > \cdots \qquad (*)$$

$(*)$ の数列のうち正の数であるものを元とする集合 S を考えれば, S の元はすべて自然数であるから, S には最小数がある。それを r_{k_0} とすれば

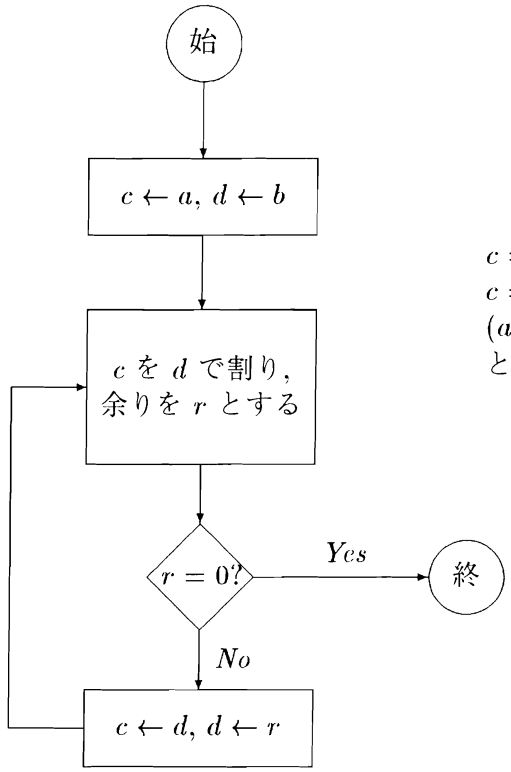
$$r_{k_0-1}=q_{k_0-1}r_{k_0}+r_{k_0+1}, \quad r_{k_0+1}=0$$

となっている。 もし $0<r_{k_0+1}<r_{k_0}$ とすれば, r_{k_0} が S の最小数であることに矛盾する。

ところで $r_{k_0}=(a,b)$ となっている。 なぜならば補題 2.13 より

$$\begin{aligned}(a,b) &= (r_0,r_1)=(r_1,r_2)=(r_2,r_3)\cdots \\ &= (r_{k_0-1},r_{k_0}).\end{aligned}$$

一方 $r_{k_0-1} = q_{k_0-1}r_{k_0}$ だから補題 2.12 より $(r_{k_0-1}, r_{k_0}) = r_{k_0}$ となり $(a,b) = r_{k_0}$ を得る。このアルゴリズムを流れ図にすれば次のようになる。



$c = a,\ d = b$
$c = dq + r$ より
$(a,b) = (c,d) = (d,r)$
となっている.

さらに変換機 (定義 2.11) でのプログラムは

```
procedure GCD;
  1: a=0
  2: b=0
  3: r=0
  4: input x
  5:   a=x
  6: input y
  7:   b=y
  8: r=x mod y
  9: while r!=0 do
 10:   a=b
 11:   b=r
```

```
 12:   r=a mod b
 13: end
 14: output b
 15: halt
endproceudre
```

となる。最大公約数 r_{k_0} を見つけるまでに k_0 回割算を実行していて各割算のステップ数の合計は

$$O(\ln a(\ln q_0 + \ln q_1 + \cdots + \ln q_{k_0-1}))$$

以下だから

$$\ln q_0 + \ln q_1 + \cdots + \ln q_{k_0-1} = \ln \prod q_k \leq \ln a$$

よりユークリッド互除法のステップ数は $O(\ln a \ln a)$ 以下であり効率の良いことがわかる。入力のサイズ $\ln a$ に対してステップ数が $O(\ln^2 a)$ となったのだからこのアルゴリズムは P に属することになる。

[実例] 素因数分解のアルゴリズム
自然数 n を素因数分解する以下のようなアルゴリズムを考察しよう。

自然数 $n \neq 0$ の最小の真の約数は素数であるから, n が合成数ならば, n を 2 および小さい順に奇数 d で割っていくと, やがて最小の真の約数にたどりつく。

$$n = dq \quad 1 < d \leq q < n$$

とすれば

$$d^2 \leq dq = n$$

となっているから最小の真の約数を見つけるには $d \leq \sqrt{n}$ まで調べればよい。

$d \leq \sqrt{n}$ まで調べたかどうか判定するには n を d で順に

$$n = qd + r, \quad 0 \leq r < d, \quad q \leq d$$

となるまで割っていけば

$$(d+1)^2 = d^2 + 2d + 1 > qd + r = n \quad \text{すなわち} \quad d + 1 > \sqrt{n}$$

であるから $d \leq \sqrt{n}$ まで調べたことがわかる。

さらにこのとき

$$n = qd + r, \quad 1 \leq r < d, \quad q \leq d$$

であれば, n は $\sqrt{n}$ 以下の数で割れていないので素数と判定してよい。この計算では入力した数 n を順に $2, 3, 5, 7, ..., [\sqrt{n}]$ と割っていくのだから各割算のステップ数を合計すると

$$O(\ln n(\ln q_2 + \ln q_3 + \cdots + \ln q_{[\sqrt{n}]}))$$

となる, ただし q_d $(d = 2, 3, ..., [\sqrt{n}])$ は n を d で割ったときの商である。ところで

$$\ln q_2 + \ln q_3 + \cdots + \ln q_{[\sqrt{n}]} < n \ln n$$

であるから, ステップ数の上界の一つは $O(n \ln^2 n)$ となる。$n = e^{\ln n}$ であるからこのアルゴリズムの上界が指数時間でとらえられたことになる[4]。変換機 (定義 2.11) でのプログラムは

```
procedure Factoriztion;
  1: input n
  2: if n<2 then halt endif
  3: d=2
  4: while d*d<=n do
  5:   if n mod d==0 then
  6:        output d
  7:        n=n div d
  8:     else d=d+1
  9:   endif
 10: end
 11: output n
 12: halt
endprocedure
```

となる。

一般に指数時間かかるアルゴリズムは効率が悪くて実際には計算が不可能といえるほど時間がかかるといえる。

2.4 非決定性計算モデル

定義 2.1 で定義された計算モデル上のプログラミング言語では文 `s1` と `s2` のいずれかを選択して, その結果として `yes` 文を最も速く実行するというような計算はでき

[4] 素因数分解の解法は着実に進歩していて現在は準指数時間でとらえられている。

ない。 s1 と s2 をともに実行して初めて yes 文が実行できることになる。 このような決定性の計算モデル (通常のコンピュータ) に対し非決定性計算モデルを以下のように定義する。

定義 2.14 (非決定性計算モデル) 文 s1 と s2 のいずれか一方を選択して, その結果としてyes 文を最も速く実行するようにeither s1 or s2 文を使用する計算モデルを非決定性計算モデルと呼ぶ。 ただし s1 は 1 個以上の文の列, s2 は 0 個以上の文の列である。

一般に either s1 or s2 文を 1 回実行するには s1 またはs2 を実行する 2 通りの場合があるから, k 回では 2^k 通りの実行の仕方があることになる。 非決定性計算モデルの特徴はこれらの実行の場合の選択を yes 文を最も速く実行するよう都合良く選択すると言う点にある。 このように s1 または s2 を非決定的に推測して実行した場合の either s1 or s2 文の実行に要するステップ数はそれぞれ s1 または s2 を選択した場合に応じて

$$T(s1)+1 \text{ または } T(s2)+1$$

ステップとなる。 非決定性計算モデル上でナップザック問題を解こう。
[実例] 非決定性計算モデル上でのナップザック問題:
入力: 自然数の列 $\{S, M_1, M_2, ..., M_n\}$
決定すること: 0,1 を要素とする列 $\{b_1, b_2, ..., b_n\}$ で

$$S = b_1M_1 + b_2M_2 + \cdots + b_nM_n$$

となるものが存在するか, は計算可能か。

[解答例]either or 文を使って以下のような procedure を考える。

```
procedure Knapsack;
  1: sum=0
  2: input S
  3: while 入力データ存在 do
  4:    input M
  5:    either sum=sum+M or
  6:    end
  7: if S==sum then yes else no
endprocedure
```

$\{S, M_1, M_2, ..., M_n\} = \{21, 9, 7, 23, 2, 3\}$ が入力されたとしよう。2 行目で `S=21` となり 3 行目から 6 行目では順に`sum=sum+9`, `sum=sum+7`, 空文, `sum=sum+2`, `sum=sum+3` と入力データ存在が存在しなくなるまで `sum` の計算が繰り返され 7 行目に来て `21=9+7+0+2+3` を判定して `yes` を出力する。つまりこの非決定性計算モデルでは `either or` 文を使って $n+1$ 個のデータのナップザック問題を $O(n)$ ステップで解いたことになる。

分割問題と呼ばれる問題も入力のサイズが n ならば非決定性計算モデル上で $O(n)$ ステップで解ける。

[**実例**] 非決定性計算モデル上で分割問題:

入力: 自然数の列 $\{M_1, M_2, ..., M_n\}$

決定すること: 部分集合 $A \subset \{1, 2, ..., n\}$ が存在して

$$\sum_{i \in A} M_i = \sum_{j \notin A} M_j$$

となるものが存在するか, は計算可能か。

[**解答例**]`either or` 文を使って以下のような `procedure` を考える。

```
procedure Partition;
  1: sum1=0
  2: sum2=0
  3: while 入力データ存在 do
  4:   input M
  5:   either sum1=sum1+M or sum2=sum2+M
  6:   end
  7: if sum1==sum2 then yes else no
endprocedure
```

$\{M_1, M_2, ..., M_n\} = \{1, 3, 7, 11, 13, 14, 15\}$ が入力されたとしよう。1 行目, 2 行目で `sum1`, `sum2` に初期値 0 が与えられ 3 行目から 6 行目では順に`sum1=sum1+1`, `sum2=sum2+3`, `sum1=sum1+7`, `sum1=sum1+11`, `sum1+13`, `sum2+14`, `sum2+15` と入力データ存在が存在しなくなるまで `sum1` あるいは`sum2` を非決定的に推測して実行し 7 行目に来て`sum1=1+7+11+13 =sum2=3+14+15` を判定して `yes` を出力する。すなわちこの非決定性計算モデルでは `either or` 文を使って n 個のデータの分割問題を $O(n)$ ステップで解くことになる。

定義 2.15 (非決定性多項式時間) 非決定性計算モデルによって多項式ステップで解くことのできる問題のクラスを非決定性多項式時間 (nondeterministic polynomial time) とよび NP で表す。

つまり
(i) 与えられた解が正しいかどうかは多項式時間で計算できても
(ii) 解の候補数が指数関数的に増加するので, 1 つ 1 つの解の候補をチェックするというプログラムでは解くことができない。
というような問題と思えばよい ([16])。この定義より決定性計算モデルによって多項式ステップで解ける問題のクラス P は当然 NP に属することになる。

$$\mathrm{P} \subset \mathrm{NP}$$

しかし $\mathrm{P} = \mathrm{NP}$ か $\mathrm{P} \neq \mathrm{NP}$ であるかは重要な未解決問題である。

ところで非決定性計算モデルでの計算を決定性計算モデルに移行して行なう場合 `either s1 or s2` 文を決定性計算モデルでは`s1`, `s2` の両方を計算してから `yes`, `no` を出力するから, もとの非決定性計算モデルのプログラムのステップ数 t に `either or` が n 回使用されていれば決定性計算モデルでのステップ数は $2^n + (t - n)$ となり

$$\mathrm{NP} \subset \mathrm{EXPTIME}$$

であることがわかる。

3 RSA 暗号と ElGamal 暗号

公開鍵暗号の考え方が提唱されて以来具体的な暗号系が数多く考案されてきた。しかしそれらの大部分は安全かつ実用的であるとはされていない ([22])。さまざまな問題点が指摘されているものの, 現在広く使用されている RSA 暗号と ElGamal 暗号の原理について解説する。これらの暗号はその安全性の根拠を素因数分解や離散対数問題といった NP 問題においている ([20]) 。

3.1 合同式 (Congruence expression)

定義 3.1 整数 a と b に対して $a - b$ が自然数 m で割り切れるとき, a と b は m を法として合同であると言い,

$$a \equiv b \pmod{m}$$

と書く。つまり

$$a \equiv b \pmod{m} \quad \Longleftrightarrow \quad m \mid a-b\,.$$

合同でないときは, $a \not\equiv b \pmod{m}$ と書く.

命題 3.2 整数 a,b,c,d 自然数 $m(\neq 0)$ に対して以下のことが成り立つ.

(i) $a \equiv a \pmod{m}$.

(ii) $a \equiv b \Rightarrow b \equiv a \pmod{m}$.

(iii) $a \equiv b,\ b \equiv c \Rightarrow a \equiv c \pmod{m}$.

(iv) $a \equiv b,\ c \equiv d \pmod{m} \Rightarrow$

$$\begin{aligned} a+c &\equiv b+d \pmod{m} \\ a-c &\equiv b-d \pmod{m} \\ ac &\equiv bd \pmod{m}. \end{aligned}$$

(v) もし $ac \equiv bc \pmod{m}$ かつ $(c,m)=1$ ならば $a \equiv b \pmod{m}$.

定理 3.3 (フェルマーの小定理) (Pierre de Fermat 1601-1665)
p を素数 a を整数とするとき

(i) $a^p \equiv a \pmod{p}$.

(ii) (フェルマーの小定理) $p \nmid a$ ならば $a^{p-1} \equiv 1 \pmod{p}$.

となる。

[証明] (i) p を素数とする。二項係数

$${}_pC_i = \frac{p(p-1)\cdots(p-i+1)}{1\cdot 2\cdots i} \quad (1 \le i \le p)$$

は, $i<p$ のとき分母は p で割れないので

$$p \mid {}_pC_i \quad (i=1,2,...,p-1)$$

となる。ただし ${}_pC_0=1,\ {}_pC_p=1$。

a に関する帰納法で証明する。$a=2$ の場合は,

$$\begin{aligned} 2^p &= (1+1)^p = {}_pC_0 + {}_pC_1 + \cdots + {}_pC_{p-1} + {}_pC_p \\ &\equiv (1+1) \pmod{p} \\ &\equiv 2 \pmod{p} \end{aligned}$$

となる。一般に $(a-1)^p \equiv a-1 \pmod{p}$ と仮定すれば,

$$\begin{aligned} a^p &= \{(a-1)+1\}^p \\ &= (a-1)^p + {}_pC_1(a-1)^{p-1} + {}_pC_2(a-1)^{p-2} \\ &\quad + \cdots + {}_pC_{p-1}(a-1) + 1 \\ &\equiv (a-1)+1 \pmod{p} \\ &\equiv a \pmod{p} \end{aligned}$$

となっている。

(ii) $p \nmid a$ かつ $p \mid a(a^{p-1}-1)$ であるから $p \mid a^{p-1}-1$. 故に

$$a^{p-1} \equiv 1 \pmod{p}.$$

(証明終)

系 3.4 (フェルマー テスト) $n > 2$ なる自然数があって n が素数か否かわからないとき, n の倍数でない a を用いて

$$a^{n-1} \not\equiv 1 \pmod{n}$$

となれば n は合成数である。

[証明] n が合成数でない, すなわち素数であると仮定すると, $n \nmid a$ であるからフェルマーの小定理より

$$a^{n-1} \equiv 1 \pmod{n}$$

となってしまう。 (証明終)

参考までにフェルマーの大定理とは: 「奇素数 p を固定したとき

$$x^p + y^p = z^p$$

となる自然数 $x, y, z > 0$ は存在しない。」であった。

3.2 オイラーの定理 (Leonhard Euler 1707-1783)

定義 3.5 自然数 n に対して $1, 2, 3, ..., n$ の中で n と互いに素な数の個数を $\varphi(n)$ で表す。$\varphi(n)$ はオイラー関数と呼ばれている。

[実例] (i) $\varphi(1)=1$

(ii) p が素数の場合, $1,2,...,p$ の中で p と互いに素な数は $1,2,...,p-1$ であるから

$$\varphi(p)=p-1$$

となる。

(iii) $n=p^m$ (p : 素数, $m\geq 1$) のときは

$$1,2,...,p,p+1,...,2p,...,p^m$$

の中で p^m と互いに素な数とは, p で割れない数である。p の倍数は p^{m-1} 個 ($\{p,2p,3p,...,p^{m-1}p\}$) あるので

$$\varphi(p^m)=p^m-p^{m-1}=p^m(1-\frac{1}{p})$$

となる。

(iv) $n=p^eq^f$ (p,q : 素数 かつ $p\neq q$, $e,f\geq 1$) のときは $1,2,...,p^eq^f$ の中で n と素でない数は, p の倍数または q の倍数である。p の倍数は p 番目ごとにあるので $\frac{n}{p}$ 個ある。同様に q の倍数は $\frac{n}{q}$ 個ある。n よりこれらを引くと pq の倍数は 2 回引かれてしまう。pq の倍数は $\frac{n}{pq}$ 個あるので

$$\varphi(n)=n-\frac{n}{p}-\frac{n}{q}+\frac{n}{pq}=n(1-\frac{1}{p})(1-\frac{1}{q})$$

となる。特に

$$\varphi(pq)=(p-1)(q-1)$$

となる。

定理 3.6 一般に $n=p_1^{e_1}p_2^{e_2}\cdots p_r^{e_r}$, ただし $p_1,...,p_r$ は相異なる素数で, $e_1,...,e_r\geq 1$, とするとき

$$\varphi(n)=n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_r})$$

となる。

[証明]

$$\begin{aligned} & n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_r}) \\ = \; & n-\frac{n}{p_1}-\frac{n}{p_2}-\cdots-\frac{n}{p_r}+\frac{n}{p_1p_2}+\frac{n}{p_1p_3}+\cdots+\frac{n}{p_{r-1}p_r} \\ & -\frac{n}{p_1p_2p_3}-\cdots+(-1)^r\frac{n}{p_1p_2\cdots p_r} \end{aligned}$$

となっている。ここで

$\frac{n}{p_1}$ は p_1 で割れる数の個数,

$\frac{n}{p_1p_2}$ は p_1p_2 で割れる数の個数,

.....

である。このときたとえば $p_1, p_2, ..., p_i$ $(i \geq 1)$ では割れるが, $p_{i+1}, ..., p_r$ では割れない数は何回数えられているか考える。

n では 1 回.

$\frac{n}{p_1}, ..., \frac{n}{p_r}$ では $\frac{n}{p_1}, ..., \frac{n}{p_i}$ の i 回, つまり i 個の素数より 1 個選び出す組み合わせの回数 ${}_iC_1$

となる。

$\frac{n}{p_1p_2}, ..., \frac{n}{p_{r-1}p_r}$ では i 個の素数より 2 個選ぶ回数, ${}_iC_2$ となる。

.....

以下同様に考えていき, 符号を考慮すれば

$$1 - {}_iC_1 + {}_iC_2 - {}_iC_3 + \cdots + (-1)^i {}_iC_i = (1-1)^i = 0$$

となり, $p_1, p_2, ..., p_i$ で割れて $p_{i+1}, ..., p_r$ で割れない数は数えられていないことになる。

$p_1, ..., p_r$ で割れない数は, 始めに展開された最初の項 n の中で 1 回だけ数えられる。つまり

$$n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_r})$$

では n と素な数が 1 回だけ数えられている。故に

$$\varphi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_r})$$

となる。 （証明終）

定理 3.7 (オイラーの定理) 自然数 n に対して $1, 2, 3, ..., n$ の中で n と互いに素な数の個数を $\varphi(n)$ とすれば, $(a, n) = 1$ なる整数 a に対して

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

となる。

上の定理において $n = p$ (素数) とすれば $\varphi(p) = p - 1$ であるからフェルマーの小定理

$$a^{p-1} \equiv 1 \pmod{p}$$

が得られる。

［証明］ $1, 2, 3, ..., n$ の中で n と互いに素な $m = \varphi(n)$ 個の数を

$a_1, a_2, ..., a_m$ とする。$(a, n) = 1$ なる a に対して, aa_i を n で割った余りを b_i とする。ただし n で割り切れる場合は余り n と考える。つまり

$$aa_i \equiv b_i \pmod{n}, \qquad 1 \leq b_i \leq n$$

とする。$aa_i = nt + b_i$ ただし t は整数であるから,

$$(aa_i, n) = (b_i, n)$$

となる。$(aa_i, n) = 1$ より $(b_i, n) = 1$ であるから

$$b_i \in \{a_1, a_2, ..., a_m\}$$

となる。一方 $i \neq j$ ならば $b_i \neq b_j$ であることは, 次のようにしてわかる。

$$\begin{aligned} b_i = b_j &\implies aa_i \equiv aa_j \pmod{n} \\ &\implies a_i \equiv a_j \pmod{n} \\ &\implies i = j. \end{aligned}$$

故に $b_1, b_2, ..., b_m$ は $a_1, a_2, ..., a_m$ を並べ換えただけである。よって

$$\begin{aligned} a_1 a_2 \cdots a_m &= b_1 b_2 \cdots b_m \\ &\equiv (aa_1)(aa_2) \cdots (aa_m) \pmod{n} \\ &= a^m a_1 a_2 \cdots a_m \end{aligned}$$

となり

$$a_1 a_2 \cdots a_m \equiv a^m a_1 a_2 \cdots a_m \pmod{n}$$

であることがわかる。ところで $(a_1 a_2 \cdots a_m, n) = 1$ だったから

$$a^{\varphi(n)} = a^m \equiv 1 \pmod{n}$$

となる。 （証明終）

3.3 RSA 暗号 (Rivest, Shamir, Adleman, 1977)

RSA 暗号について簡単に説明すると以下のようになる。参考書によって多少の違いはある。p, q を非常に大きな 201 桁程度の 2 つの異なる公表されていない素数 (1

より大きな自然数で, 正の約数が 1 と自分自身のみであるもの) とする。このとき次のような自然数や自然数の集合を用意する。

$$\begin{aligned} N &= pq \ (\text{この } p \text{ と } q \text{ は非公開とする}) \\ L &= \varphi(pq) = (p-1)(q-1) \\ \ell &: p-1 \text{ と } q-1 \text{ の最小公倍数} \\ \mathcal{M}(=\mathcal{C}) &= \{400 \text{ 桁以下の自然数で } N \text{ と互いに素なもの}\} \end{aligned}$$

上の数を元にして $i \in I$ さんの公開鍵 $\{c_i, N\}$ 秘密鍵 $\{d_i\}$ を $1, 2, ..., \ell$ のなかから以下のように作る。

$$\begin{aligned} c_i &: L \text{ と互いに素な自然数} \\ d_i &: c_i d_i \equiv 1 \pmod{L} \text{ となるような } d_i \end{aligned}$$

d_i が存在するのは c_i と L が互いに素であることによる。オイラーの定理より任意の $m \in \mathcal{M}$ に対して

$$m^L = m^{\varphi(N)} \equiv 1 \pmod{N} \quad (*)$$

が成り立っていることに注意して $i \in I$ さんの組立関数 E_i を

$$\begin{aligned} E_i : \mathcal{M} &\longrightarrow \mathcal{C} \\ m &\longmapsto m^{e_i} \pmod{N} \end{aligned}$$

翻訳関数 D_i を

$$\begin{aligned} D_i : \mathcal{C} &\longrightarrow \mathcal{M} \\ c &\longmapsto c^{d_i} \pmod{N} \end{aligned}$$

と定義する。このとき

$$c_i d_i \equiv 1 \pmod{L} \Leftrightarrow \text{ ある自然数 } t \text{ が存在して } c_i d_i = 1 + tL$$

であるから $(*)$ より

$$m^{e_i d_i} = m^{1+tL} = m \cdot (m^L)^t \equiv m \pmod{N} \quad (m \in \mathcal{M})$$

となり, 任意の $m \in \mathcal{M}$, $c \in \mathcal{C}$ に対して

$$D_i(E_i(m)) = D_i(m^{e_i}) = (m^{e_i})^{d_i} = m^{e_i d_i} \equiv m \pmod{N}$$

$$E_i(D_i(c)) = E_i(c^{d_i}) = (c^{d_i})^{e_i} = c^{e_i d_i} \equiv c \pmod{N}$$

となっている。大きな自然数 N (400 桁程度) の素因数分解は一般には非常に難しいから, $i \in I$ さんの鍵として $\{c_i, N\}$ を公開してもそれを解く鍵 $\{d_i\}$ は N の素因数分解

$$N = pq$$

がわからない限り求められない これで公開鍵暗号が実現できたことになる。
[**実例**] 簡単のため

$$\begin{aligned} p &= 19,\ q = 23 \\ N &= pq = 437 \\ L &= \varphi(pq) = 396 \\ \ell &= 198 \\ \mathcal{M}(=\mathcal{C}) &= \{1, 2, 3, ..., 436\} \text{ (ただし } N \text{ と互いに素)} \end{aligned}$$

とする。$i \in I$ さんの公開鍵 $\{c_i, N\}$ 秘密鍵 $\{d_i\}$ を以下のように作ることができる。

$$\begin{aligned} c_i &= 13 \ (L \text{ と互いに素}) \\ d_i &= 61 \end{aligned}$$

このときたとえば $m = 2$ とすれば

$$E_i : 2 \longmapsto 2^{13} = 8192 \equiv 326 \pmod{437}$$

$$\begin{aligned} D_i : 326 &\longmapsto 326^{61} \\ &= 20242940758841557382.....576 \ (154 \text{ 桁}) \\ &\equiv 2 \pmod{437} \end{aligned}$$

となって

$$D_i(E_i(2)) \equiv 2 \pmod{437}$$

であることがわかる。

3.4 ElGamal 暗号 (T. ElGamal, 1985)

この暗号は有限体 (加減乗除が定義された有限集合) における離散対数 (discrete logarithm) の計算:「有限体 F の元 g, y が与えられたとき

$$g^x = y$$

となる $x \in \mathsf{Z}$ を見つけること。」
の困難さを利用して電子署名と公開暗号を実現するものである。実際には大きな素数 p と 自然数 $g < p$ に対して, 自然数 $x < p$ が与えられて $y \equiv g^x \pmod{p}$ を計算するのは簡単であるが, 逆に p, g, y から x を求めるのは困難であるという事実を利用することが多い。

ElGamal 暗号で平文 m に署名するには以下のようにする。まず公開鍵として次の 3 つの整数 p, g, y と 秘密鍵 x を用意する。すなわち

(i) p は 150 桁以上の素数。

(ii) g はランダムな自然数で $g < p$ を満足するもの。

(iii) $y \equiv g^x \pmod{p}$ ただし

(iv) x はランダムな自然数で $x < p$ を満足するもの。

(i), (ii), (iii) は特定のグループ B で共有され (iv) はそのグループ B に署名付きの平文を送りたい者 A に秘密鍵として渡される。A は

(v) $p-1$ と互いに素な自然数 k をランダムに選び (拡張) ユークリッド互除法を使って k の逆数 $k^{-1} \pmod{p-1}$

$$k^{-1}k \equiv 1 \pmod{p-1}$$

を計算し,

(vi) 次の a, b を m への署名とみなして (m, a, b) を B へ送る。

$$\begin{aligned} a &\equiv g^k \pmod{p} \\ b &\equiv k^{-1}(m - xa) \pmod{p-1} \end{aligned}$$

b の定義から

$$m \equiv xa + kb \pmod{p-1}$$

であることに注意してほしい。a, g を知っていても k を求めることは困難であり b の計算には x を必要としていて, しかも y から x を求めることも困難であるから, B では式

$$y^a a^b \equiv g^{ax} g^{kb} \equiv g^{xa+kb} \equiv g^m \pmod{p}$$

を計算することで (m,a,b) の送信元は A であると強く推測できる。

ElGamal 暗号で平文 m を暗文 c に変換するには以下のようにする。署名するときと同様に公開鍵として次の 3 つの整数 p,g,y と秘密鍵 x を用意する。 すなわち

(i) p は素数。

(ii) g はランダムな自然数で $g<p$ を満足するもの。

(iii) $y \equiv g^x \pmod{p}$ ただし

(iv) x はランダムな自然数で $x<p$ を満足するもの。

(i), (ii), (iii) は特定のグループ B で共有され, そのグループ B に暗文を送りたい者 A は

(v) $p-1$ と互いに素な自然数 k をランダムに選び

(vi) 次の a,b を m の暗文 c とみなして $c=(a,b)$ を B へ送る。

$$\begin{aligned} a &\equiv g^k \pmod{p} \\ b &\equiv y^k m \pmod{p} \end{aligned}$$

a,g を知っていても k を求めることは困難であり b の計算にも k を必要としているから, B では式

$$b/(a^x) \equiv (y^k m)/(g^k)^x \equiv ((g^x)^k m)/(g^{kx}) \equiv m \pmod{p}$$

を計算することで暗文 $c=(a,b)$ を原文 m に翻訳できる。

$\star\star\star\star\star\star\star$

以上がわかりやすい部分だけをとらえた暗号理論のごく一部分の概説であるが, 楕円曲線群といったより複雑な代数的構造を利用する暗号も一般的なものになりつつある。実用的な暗号理論を基礎から勉強したい方は [1], [4], [5], [6], [22] などを, 数学にも興味のある方は [19], [20], [23] などを, 実際に暗号を使用しながら学びたい方は [8] や [15] を, 計算の基礎について瞑想したい方は [2], [14] や [16] を参考にするとよいだろう。その前に [3] や [12] などの入門書を読んでおくのも全体像が浮かび上がってよいと思う。[23] の日本語版は共立出版から出ている。

参考文献

[1] 池野信一, 小山謙二. 現代暗号理論 (第 5 版). 電子情報通信学会 1995

[2] 岩田茂樹. **NP** 完全問題入門. 共立出版 1995

[3] 太田和夫, 黒澤馨, 渡辺治, 情報セキュリティの科学. 講談社ブルーバックス 1995

[4] 岡本栄司, 暗号理論入門. 共立出版 1993

[5] 岡本龍明, 太田和夫 編, 暗号・ゼロ知識証明・数論. 共立出版 1995

[6] 岡本龍明, 山本博資, 現代暗号. 産業図書 1997

[7] 加藤正隆, 基礎暗号学 I,II. サイエンス社 1989

[8] 川原稔, What is Unix 66-71. *Unix User. 1998.2 - 1998.7* (株) アスキー

[9] 木田祐二, 牧野潔夫, UBASIC によるコンピュータ整数論. 日本評論社 1994

[10] 澤田秀樹, 暗号理論と代数学, 海文堂 1997

[11] 高木貞治, 初等整数論講義. 共立出版 1981

[12] 一松信, 暗号の数理. 講談社ブルーバックス 1980

[13] 松井甲子雄, コンピュータの為の暗号組立法入門. 森北出版 1986

[14] 守屋悦朗, チューリングマシンと計算量の理論. 培風館 1997

[15] 山本和彦, 転ばぬ先のセキュリティ 1-26. *Unix Magazin, 1994.5 - 1996.8* (株) アスキー

[16] 渡辺治, 計算可能性 • 計算の複雑さ入門. 近代科学社 1992

[17] 和田秀男, コンピュータと素因子分解. 遊星社 1987

[18] D.E.R.Denning, Cryptography and data security. *Addison-Wesley 1983*

[19] N.Koblitz, A course in number theory and cryptography, 2nd edition. *Springer-Verlag, 1994*

[20] N.Koblitz, Algebraic aspects of cryptography. *Springer-Verlag, 1998*

[21] W.Patterson, Mathematical cryptology for computer scientists and mathematicians.*Rowman & Littlefield 1987*

[22] B.Schneier, Applied cryptography, 2nd edition. *John Wiley & Sons, Inc. 1996*

[23] D.G.Stinson, Cryptography. Theory and Practice. *CRC Press, 1995*

(さわだ ひでき, 山形大学理学部)