教養原論 数理の世界
数学とコンピュータ 2010 (第 7 回)
有限体の応用 – RSA 暗号

野呂 正行

神戸大学大学院理学研究科数学専攻

November 18, 2010

暗号とは

- **暗号化**
  ある規則で, 文字を別の文字に変換
- **復号**
  暗号文をもとの文 (平文; ひらぶん) に戻す
- **単純なもの : 表を使う**
  例 : アルファベットを単に何文字かずらす (シーザー暗号)
  ⇒ 文字の出現頻度を調べると解読できる
  参考 : 英文字の出現頻度 (COD; **数字は%**)
  E(11), A(8.5), R(7.6), I(7.5), O(7.2), T(7.0), N(6.7), S(5.7), L(5.5), C(4.5), U(3.6), D(3.4), P(3.2), M(3.0), H(3.0), G(2.5), B(2.1), F(1.8), Y(1.8), W(1.3), K(1.1), V(1.0), X(0.3), Z(0.3), J(0.2), Q(0.2)

暗号方式

暗号は, 一般にアルゴリズムにより暗号化, 復号
— 秘密保持のため, 鍵が必要

- **共通鍵暗号** (例 : DES, AES)
  暗号化, 復号に同じ鍵を使う
  鍵は, 通信者 A, B が何らかの方法で共有する
  $\Rightarrow$ 鍵の共有方法が問題

- **公開鍵暗号** (例 : RSA, 楕円曲線暗号)
  A から B にデータを送る場合, B が公開している鍵で暗号化してから送る
  B は, 自分が持つ秘密鍵で復号する
  $\Rightarrow$ 一時的に使う秘密鍵の送付に使える

例 : https://. . .

A が web shop B で物を買いたいとする
(B の URL は信用できるとする)
問題点 : インターネット上では常に盗聴の危険あり

- 初対面の相手と安全に通信するにはどうするか?
  データは暗号化する必要がある
  ⇒ 自分と相手だけが知る鍵が必要
- 初対面で, どうやって 鍵を共有するか?
  A が共通鍵を作り, B に渡せばよい
- 共通鍵を相手に安全に渡すにはどうすればよいか
  B が公開鍵をつくり, 公開し, A は共通鍵を, その公開鍵で暗号化して渡す

公開鍵暗号の要件

- **公開鍵から秘密鍵がバレない**
  暗号化方法は当然公開されている (復号方法も)
  公開鍵と秘密鍵は当然密接に関係している
  $\Rightarrow$ **公開鍵から秘密鍵がバレないような方式が必要**

- **暗号文から平文が簡単に復元できない**
  鍵がなくても平文が復元できるようでは意味がない
  例 : 一文字ずつ変換したのでは, 頻度解析などで解読できる
  $\Rightarrow$ **複数文字 (文字列) を変換できなければいけない**

RSA 暗号

Rivest, Shamir, Adleman により考案された (1977)
前述の問題点を解消できる暗号方式の一つ

- **公開鍵 n は素数 p, q を 2 つかけたもの** $(n = pq)$
  例 : 1024 ビット RSA 暗号の公開鍵は, 150 桁程度の素数を 2 つかけたもの
  $0 \leq x \leq n-1$ なる整数 x を, 同じ範囲の整数に変換, 逆変換できる.

- **文字列の変換**
  各文字を数に変換してつなげた大きい整数として変換
  アルファベット (ASCII) : 1 バイト
  漢字, かな : JIS なら 2 バイト, UTF-8 なら 3 バイト
  以上

RSA 暗号で使う有限体の性質

- **フェルマー (Fermat) の小定理**
  $a \in \mathbf{F}_p$ **ならば**, $a^p = a$. **さらに**, $a \neq 0$ **ならば** $a^{p-1} = 1$.
  言い替えれば, a **を整数**, p **を素数とすると**
  $a^p \equiv a \bmod p$.

- **ちなみに, フェルマーの大定理**
  n **が** 3 **以上の整数とするとき**, $x^n + y^n = z^n$ **を満たす整数** x, y, z **は存在しない**.
  (とんでもない難問だった $\Rightarrow$ 300 **年以上かかって解決)**

RSA 暗号：準備

- **公開鍵** (n, e) **(**$e = 65537$ **がよく用いられる)**
  大きい素数 p, q **(**p, q **は同じくらいの大きさで**, $p \neq q$**)**
  を用意する.
  実用上 p, q **は** 10^{150} **くらいで**, $p-1, q-1$ **が** $e = 65537$
  で割り切れないものを選ぶ.
  $\Rightarrow n = pq$ **に対し**, (n, e) **が公開鍵**

- **秘密鍵** d
  e **は** $n' = (p-1)(q-1)$ **を割り切らない素数なので**,
  $GCD(e, n') = 1$
  $\Rightarrow \mathbf{F}_e$ **での** n' **の逆数を** k **とすると** $kn' \equiv 1 \bmod e$
  $\Rightarrow kn' + de = 1$ **を満たす整数** d **がある**.
  $\Rightarrow de \equiv 1 \bmod n'$
  $\Rightarrow d$ **にいくつか** n' **を足せば** $d > 0$
  $\Rightarrow$ **この** d **を秘密鍵とする**.

RSA 暗号：安全性

- **鍵の安全性**

  d を作るには $n' = (p-1)(q-1)$ が必要

  $\Rightarrow$ p, q が必要

  $\Rightarrow$ n の素因数分解が必要

  $\Rightarrow$ $n \simeq 10^{300}$ では事実上無理

  (最近では $\Rightarrow$ $n \simeq 10^{600}$ が推奨されている)

  すなわち，「数学者，計算機科学者，コンピュータの現在の能力では素因数分解ができない」ことに基づく暗号方式が RSA 暗号

  量子コンピュータが実用化されると危うい?

- **e の選び方**

  e はある程度大きい素数ならなんでもよい

RSA 暗号：暗号化

1. **データを数字の列に変換**

   例えば, JIS 2 **バイト文字 (漢字など) の列なら**

a_1	a_2	$\cdots$	a_l

   $0 \le a_i \le 2^{16} - 1$ (2 **バイト**)

2. 1. **で作った数の列を**, n **未満の数に分割する**.

a_1	$\cdots$	a_m	a_{m+1}	$\cdots$	a_{2m}	$\cdots$

   2 **バイトが** m **桁** $\Rightarrow$ 2 **進** 2^{16m} **桁**

   $\Rightarrow 2^{16m} \le n$ **となるように** m **をとればよい**.

   $A_1 =$ | a_1 | $\cdots$ | a_m |, $A_2 =$ | a_{m+1} | $\cdots$ | a_{2m} |, $\cdots$ **とおく**.

   すなわち,

   $$A_1 = a_1 \cdot 2^{16(m-1)} + \cdots + a_{m-1} \cdot 2^{16} + a_m, \ldots$$

   と見なす. **このとき** $0 \le A_i \le n - 1$ **である**.

3. **各** A_i **を暗号化** : $E_i \leftarrow A_i^e \bmod n$

RSA 暗号：復号

1. **各 E_i に対し $D_i \leftarrow E_i^d \bmod n$**
   実は, $D_i = A_i$ となっている.

2. **数字の列を元に戻す**
   作った D_i を, 2 バイトずつに区切り, 文字に戻す

注意：E_i から A_i を求めるのは困難.
e 乗して $\bmod n$ したら E_i になる数は何か?
$\Rightarrow 0, 1, \ldots, n-1$ を e 乗して $\bmod n$ すれば求まるが, n が巨大だと無理

簡単な例

$n = 7763996327$, $e = 65537$, $d = 5362894673$

暗号文 : 2581306503 **を復号してみる**.

$$2581306503^d \equiv 3220491773 \bmod n$$

$$3220491773 = (\text{bff4cdfd})_{16}$$

実は $(\text{bff4})_{16}$ **は「数」**, $(\text{cdfd})_{16}$ **は「理」** (EUC **で**)

実際, **「数理」を暗号化してみると**

$$3220491773^e \equiv 2581306503 \bmod n$$

となることがわかる.

数学的な証明が必要な部分

- p が素数なら $a^p \equiv a \bmod p$ (フェルマーの小定理)
- $E = (A^e \bmod n)$, $D = (E^d \bmod n)$, $0 \leq A, D \leq n-1$ ならば $D = A$
  これは $A^{ed} \equiv A \bmod n$ を示せばよい.

$\Rightarrow$ のちほど
基本は,

- ab が p で割り切れるなら a か b が p で割り切れる.
  これが p が「素」であることの基本的性質
- a が $n(= pq)$ で割り切れる $\Leftrightarrow$ a が p でも q でも割り切れる

$A^e \bmod n$, $E^d \bmod n$ の計算は大変そう

$\Rightarrow$ **繰り返し 2 乗法** (repeated squaring) **剰余計算の基本原理**

$$(ab) \bmod n = (a \bmod n)(b \bmod n) \bmod n$$

を繰り返し使う.

1. $k = 2k' + r$ ($r = 0$ **または** $r = 1$) **とする**
2. $a^k \bmod n = a^{2k'+r} \bmod n = a^{k'} \cdot a^{k'} \cdot a^r \bmod n$
   $r = 1$ **なら** $a^r = a$ $r = 0$ **なら** $a^r = 1$ **に注意する.**
   $\Rightarrow a^{k'} \bmod n$ **が計算できていれば,** $a^k \bmod n$ **も, 基本原理の適用で計算できる.**
3. $a^{k'} \bmod n$ **は, 同じ方法で計算する**
   このような計算を再帰 (recursion) **という**

繰り返し 2 乗法の効率

$a^k \bmod n$ を計算する場合

$k = (b_m \ldots b_1)_2$ ($b_i = 0$ または 1) とすれば, $k' = (b_m \ldots b_2)_2$.

$\Rightarrow m$ ステップで終了する.

$\Rightarrow m \simeq \log_2 k$ だから, $d \simeq 10^{300}$ でも 1000 ステップ程度で

$E^d \bmod n$ は計算できる.

(間違っても E をひとつずつかけたりしてはいけない.)

例: $8^{100} \bmod 15$ の計算

$8^{100} \bmod 15 = (8^{50} \bmod 15)^2 \bmod 15 = 4^2 \bmod 15 = 1$

$8^{50} \bmod 15 = (8^{25} \bmod 15)^2 \bmod 15 = 8^2 \bmod 15 = 4$

$8^{25} \bmod 15 = (((8^{12} \bmod 15)^2 \bmod 15) \cdot 8) \bmod 15 = 8$

$8^{12} \bmod 15 = (8^6 \bmod 15)^2 \bmod 15 = 4^2 \bmod 15 = 1$

$8^6 \bmod 15 = (8^3 \bmod 15)^2 \bmod 15 = 2^2 \bmod 15 = 4$

$8^3 \bmod 15 = ((8^2 \bmod 15) \cdot 8) \bmod 15 = 4 \cdot 8 \bmod 15 = 2$

(**下から上へ遡る**)

素数 p, q の作り方

- **整数をランダムに発生させて, 素数かどうかチェック**
- **素数チェック**
  確定的ではないが, 通ればほとんど間違いなく素数, というチェック方法がある
  (実用上, 「ほとんど間違いなく素数」で問題ない, とされる)
- **整数をランダムに発生させる**
  より乱数発生アルゴリズムがいろいろある

フェルマーの小定理の証明

a は p で割り切れないとする. $x \bmod p$ を $\overline{x}$ と書く.

1. $1 \leq i, j \leq p-1$ のとき, $ai \equiv aj \bmod p$ なら $i = j$
   $ai \equiv aj \bmod p$ なら $a(i-j)$ が p で割り切れる. a が p で割り切れないので $i-j$ が p で割り切れる. これは $i = j$ の場合のみ成り立つ.

2. $\{\overline{a \cdot 1}, \ldots, \overline{a \cdot (p-1)}\} = \{1, \ldots, p-1\}$ (集合として等しい)
   1. より $\overline{a \cdot i}$ はすべて異なり, 0 とも異なるので OK.

3. $a^{p-1} \cdot (p-1)! \equiv (p-1)! \bmod p$
   2. の両辺の要素をそれぞれかけた.

4. $a^{p-1} \equiv 1 \bmod p$
   $(a^{p-1}-1)(p-1)!$ が p で割り切れる. $(p-1)!$ は p で割り切れないので $a^{p-1}-1)$ の方が p で割り切れる.

$A^{de} \equiv A \bmod n$

$de \equiv 1 \bmod n'$ **より** $de - kn' = 1$ **を満たす** k **がある.**

$A^{de} = A^{1+kn'}$ **より** $A^{1+kn'} \equiv A \bmod n$ **をいえばよい.**

$n = pq$ **より次をいえばよい:**

$$A^{1+kn'} \equiv A \bmod p, \quad A^{1+kn'} \equiv A \bmod q$$

1. $A \equiv 0 \bmod p$ **のとき**

   $A^{1+kn'} \equiv 0 \bmod p$ **より** $A^{1+kn'} \equiv A \bmod p$

2. $A \equiv 0 \bmod p$ **でないとき (**A **が** p **で割り切れないとき)**

   $A^{1+kn'} = A \cdot A^{k(p-1)q-1} = A \cdot (A^{p-1})^{k(q-1)}$ **で,**

   A **が** p **で割り切れないから** $A^{p-1} \equiv 1 \bmod p$.

   よって, $A^{1+kn'} \equiv A \bmod p$.

(mod$q$ **も同様)**