合同式のはなし　—暗号理論入門—

小川 裕之　(大阪大学大学院 理学研究科)[†]

§1　序

数学 A で合同式を学びます. 合同式が織りなす数学の世界を覗いてみましょう. 講演会では, 合同式とその性質について説明し, 暗号理論の入門, RSA 暗号について説明し, Euclid の互除法の使い方を紹介します. この雑文は, 講演会のレジュメ (講演要旨) ではありません. 講演会の内容を補完し, 合同式のもつ可能性に気づいてもらうための文章です. 講演会ではこの文章をたどることはなく, ここに書いていない話もします. 以下の構成ですが, 2 章は合同式の話, 3 章は暗号理論の話です. 2.4 節までで合同式の定義と性質をまとめます. 2.5 節, 2.6 節は合同式の性質を単なる便利な公式ではなく, 数学的意味を明確にする立場から少し難しい説明を書いています. 暗号理論の基礎的な考え方は 3.3 節までに書いています. 3.4 節以降で RSA 暗号を紹介します. 3.3 節までを理解できれば, 必要な数学を勉強することで暗号理論を自修することもできると思います,

§2　合同式

2.1　合同式は, 整数の余りを許す割り算に関わる考え方です. 整数の割り算を整数の範囲で考えると, $13\div 3$ や $14\div 5$ などのように割り切れず, 余りが現れます. 有理数や小数を使うと $13\div 3=\frac{13}{3}=4.333\cdots$, $14\div 5=\frac{14}{5}=2.8$ で, 余りは要りません. 余りのある割り算は整数特有のものです.

ある整数で割った余りで整数全体を分類してみましょう. 例えば 7 で割ると余りとして $0,1,2,3,4,5,6$ の 7 つの整数が現れます. ところで今日は 10 月 28 日水曜日です. 1 週間は 7 日で, 7 日おきに曜日が決まっていますから, 21 日, 14 日, 7 日は水曜日です. それらの次の日の, 29 日, 22 日, 15 日, 8 日と 1 日は木曜日です. 今年の 10 月は 7 で割り切れる日が水曜日, 7 で割って 1 余る日が木曜日です. 曜日に注目するときは, 7 で割った余りを見ればいい. こういう計算はやったことがあると思います.

2.2　a,b,m は整数で, $m\neq 0$ とします. $a-b$ が m で割り切れるとき, **法 m に関して a と b は合同である** といい, $\boldsymbol{a\equiv b\pmod m}$ で表します. 例えば, $8-2(=6)$ は 3 で割り切れるので, $8\equiv 2 \pmod 3$ です. $8-(-6)(=14)$ は 7 で割り切れるので $8\equiv -6 \pmod 7$ です. また, $8\equiv 3 \pmod 5$, $8\equiv -2 \pmod 5$, $8\equiv -7 \pmod 5$ です. $7\equiv 1 \pmod 6$, $13\equiv 1 \pmod 6$, $19\equiv 1 \pmod 6$, $25\equiv 1 \pmod 6$ なので, 7, 13, 19, 25 は 6 を法として合同です.

2.3　余りに注目すると言いながら, 合同式の定義に余りが出てきません. $a\equiv b \pmod m$ は a を m で割った余りが b であるという, 余りを求める計算式ではありません. 敢えて余りを出すなら, a を m で割った余りと b を m で割った余りが等しいということで, 2 つの数 a,b の間の関係を表しています. 曜日の計算を例に見てみましょう. 余りを引き出す $28\equiv 0 \pmod 7$ は今日 (今年の 10 月 28 日) が水曜日 (余りが 0) であることを意味し, 余りによりその日の曜日がわかります. 今年の 10 月 28 日と 10 月 21 日は同じ曜日であることを表す $28\equiv 21 \pmod 7$ は, 今年の 10 月だけでなく, 同じ年同じ月であれば 28 日と 21 日が同じ曜日であることを表します. 法 7 に関する合同式 $a\equiv b \pmod 7$ で, 同じ年同じ月の a 日と b 日が同じ曜日であることを表せます.

2.4　日にちと曜日の関係ですぐに思いつきそうなことを挙げてみましょう:

(i)　a 日と b 日が同じ曜日で, b 日と c 日が同じ曜日なら, a 日と c 日も同じ曜日です.
(ii)　a 日と b 日が同じ曜日のとき, a 日の c 日後と b 日の c 日後も同じ曜日です.
(iii)　a 日と b 日が同じ曜日のとき, a 日の a 日後と b 日の b 日後も同じ曜日です.

(i), (ii) は証明とか何とか言うものではなく, 当たり前のことに思えます. (iii) は少しギョッとするか

[†] 〒 560-0043　大阪府豊中市待兼山町 1-1　　ogawa@math.sci.osaka-u.ac.jp
2020 年 10 月 28 日　於 奈良県立 奈良高等学校

もしれません. なんとも不思議です. ともかくこれらを合同式にしてみます. 上で紹介していない形や, 上とはやや異なる表現もありますが, 合同式の性質としてまとめておきます.

[合同式の性質]

(1) $a \equiv a \pmod{m}$

(2) $a \equiv b \pmod{m}$ ならば $b \equiv a \pmod{m}$

(3) $a \equiv b \pmod{m}$ かつ $b \equiv c \pmod{m}$ ならば $a \equiv c \pmod{m}$

(4) $a \equiv b \pmod{m}$ かつ $c \equiv d \pmod{m}$ ならば $a + c \equiv b + d \pmod{m}$

(5) $a \equiv b \pmod{m}$ かつ $c \equiv d \pmod{m}$ ならば $ac \equiv bd \pmod{m}$

これらは, 合同式の定義から簡単に証明できます. 日にちと曜日の関係 (i) は合同式の性質 (3) で, 関係 (ii) は性質 (4) です. ちょっと不思議な関係 (iii) は, 性質 (5) で $c = d = 2$ の場合です. 性質 (5) は性質 (3), (4) から導かれます. $a \equiv b \pmod{m}$ の両辺に a を足して $2a \equiv b + a \pmod{m}$, b を足して $a + b \equiv 2b \pmod{m}$ なので, $2a \equiv a + b \equiv 2b \pmod{m}$ です. これが関係 (iii) です. 繰り返すことで $ac \equiv bc \pmod{m}$ (c は整数) を得ます. . $c \equiv d \pmod{m}$ について $bc \equiv bd \pmod{m}$ (b は整数) なので, $ac \equiv bc \equiv bd \pmod{m}$ を得ます.

2.5　合同式の性質の意味について, 少し難しい説明をします. 難しければ, ここことこの次の節を読み飛ばしてもかまいません. (1), (2), (3) は, 等号 ($=$) や同値 ($\Leftrightarrow$) のような, 両者が同等であることが満たすべき基本的関係式です. これら 3 条件を満たすものをある意味での「等号」(通常の等号と区別するため括弧をつけます) とみなすことができます. 通常の等号は, この「等号」の考え方のモデルです. 同値 ($\Leftrightarrow$) は命題 (数学的な文章) の間の「等号」です. 合同式も余りの世界の「等号」なのです. ここで更に難しい話をします. $28 \equiv 21 \pmod{7}$ ということは, 7 で割った余りの世界で 28 と 21 が等しい (合同である) ことを表しますが, 整数として $28 = 21$ ではありません. あたりまえですね. 法 7 に関する合同の世界では, 28 も 21 も, さらに 14, 7, 0, -7 などたくさんの整数が等しい (合同である) わけです. そこでは, 整数全体を余りに対応した 7 つの部分に分け, それぞれの部分は互いに合同なたくさんの整数からなります. 互いに合同な数の集まりを合同類あるいは剰余類と呼びます. ある合同類に属する整数は, その合同類を整数の世界の言葉で表すための代表者にあたります. 法 7 の 7 つの合同類の代表の集まりとして, 7 で割ったときの標準的な余り $0, 1, 2, 3, 4, 5, 6$ を取ることができます. $-3, -2, -1, 0, 1, 2, 3$ も法 7 の合同類の代表の集まりです. $0, 1, 10, 100, 1000, 10000, 100000, 1000000$ も法 7 の合同類の代表の集まりです.

2.6　合同類の代表の集まりという考え方で, 合同式の性質 (4) を見てみましょう. $a \equiv b \pmod{m}$ は, a と b が法 m に関する同じ合同類を代表していることを意味します. $c \equiv d \pmod{m}$ も同じことです. $a + c \equiv b + d \pmod{m}$ も同じで, $a + c$ と $b + d$ が同じ合同類を代表するということです. 代表として a と c を選んだ時の和 $a + c$ と, 代表として b と d を選んだ時の和 $b + d$ は, 同じ合同類を代表する. 合同類の代表として何を選んでも, それらの和は同じ合同類に属する. 合同類の集まり (法 m に関する合同の世界) に和が定義されたのです. 合同式の性質 (5) は, 法 m に関する合同の世界に積が定義されることを意味します.

2.7　話が一気に難しくなってしまいました. 合同式の定義と性質を使ってちょっと気の利いたことをしてみましょう. 突然ですが, 整数を 3 で割った余りの簡単な計算方法を知ってますか. すべての桁の数字をそのまま加えて, それを 3 で割ればよい. どうしてこれでいいのでしょう. $10 = 3 \times 3 + 1$ だから $a \times 10 = 3 \times (3a) + a$ で, $100 = 3 \times 33 + 1$ だから $a \times 100 = 3 \times (33a) + a$ となります. 同様にして $a \times 10 \cdots 0 = 3 \times (3 \cdots 3\, a) + a$ です. ですから例えば "abc" (少し表記が紛らわしいですが, 百の位が a, 十の位が b, 一の位が c の 3 桁の整数) について, "abc"$= 3 \times (33a + 3b) + (a + b + c)$ なので, "abc" を 3 で割った余りは $a + b + c$ を 3 で割った余りに等しい.

合同式の立場で見てみましょう. $10 - 1 = 9 = 3 \times 3$ なので $10 \equiv 1 \pmod{3}$ です. $100 = 10^2 = 10 \times 10 \equiv 1 \times 1 = 1^2 = 1 \pmod{3}$ で, $1000 = 10^3 \equiv 1^3 = 1 \pmod{3}$, $10000 = 10^4 \equiv 1^4 = 1 \pmod{3}$ ですから, $10^n \equiv 1 \pmod{3}$ (n は正の整数) です. 先ほどの 3 桁の整数 "abc" について, "abc"$= a \times 10^2 + b \times 10 + c \equiv a \times 1 + b \times 1 + c = a + b + c \pmod{3}$ となります. $10 \equiv 1 \pmod{3}$

から始まったことで, "abc"$\equiv a+b+c \pmod 9$ が成り立ちます.

少し応用です. 11 では, $10 \equiv -1 \pmod{11}$ です. $10^n \equiv (-1)^n \pmod{11}$ なので, n が奇数のとき $10^n \equiv -1 \pmod{11}$ で, n が偶数のとき $10^n \equiv 1 \pmod{11}$ です. 従って 11 で割った余りは, 各桁の数を下の位から符号を交互に変えて和をとった数を 11 で割った余りに等しい. 3 桁の整数 "abc" について, "abc"$\equiv a-b+c \pmod{11}$ なので, $111 \equiv 1-1+1 = 1 \pmod{11}$, $123 \equiv 1^2+3 = 2 \pmod{11}$, $946 \equiv 9-4+6 = 11 \equiv 0 \pmod{11}$ です.

2 や 5 の割り算は簡単なので, 7 で割った余りを何とか簡単に計算できないでしょうか. $10 \equiv 3 \pmod 7$ だから, 十の位はその桁の数を 3 倍して, 百の位は 9 倍して, 千の位は 27 倍して... でも $27 \equiv 6 \pmod 7$ だから 6 倍でもいいけど, $27 \equiv -1 \pmod 7$ だから -1 倍でもいい. 従って $a \times 1000 + b \equiv b - a \pmod 7$ です. 例えば $123456 \equiv 123 \times 1000 + 456 \equiv 456 - 123 = 333 \pmod 7$ です, また $100 \equiv 9 \equiv 2 \pmod 7$ だから, $333 \equiv 3 \times 100 + 33 \equiv 3 \times 2 + 33 = 6 + 33 = 39 \pmod 7$ です. こうして 2 桁以下の数になりますが, 2 桁ぐらいの数なら 7 で割った余りの計算は簡単でしょう. まとめると, 整数を下から 3 桁ずつに分けて, 符号を交互に変えて和を取って, 必要ならこれを繰り返して 3 桁以下の数にします. そしてその 3 桁の数の百の位を 2 倍して残りの 2 桁の数に足して, 必要ならもう一回繰り返して 2 桁の数にします. あとはそれを 7 で割ればよい.

最後に難敵 13 です. $10 \equiv -3 \pmod{13}$ なので, $100 = 10^2 \equiv 3^2 = 9 \pmod{13}$, $1000 = 10 \times 100 \equiv 3 \times 9 \pmod{13}$ です. ここで $9 \equiv -4 \pmod{13}$ に気づけば, $1000 \equiv 3 \times (-4) = -12 \equiv 1 \pmod{13}$! 下から 3 桁ずつに分けて全部足すことを繰り返して 3 桁の数にします. あとは, 百の位を 9 倍して残り 2 桁の数に足すことを繰り返して, 2 桁の数にします. 九九に 13 の段はないので, 暗算は難しいかもしれませんが, 十の位を 3 倍して一の位の数から引いたら, -27 から 9 までの整数になるので, 後は何とかなるでしょう.

§ 3　暗号理論への入門

3.1　皆さんの多くがパソコンやスマホなどを使い, LINE やメールなどいろいろな情報通信サービスを利用していると思います. ドコモ口座を介した不正送金から派生した, 詐取事件が起こりました. 報道な少なくなっていますが, その全貌は未だ明らかになっておらず, 今も被害は拡大し続け, 終息の目途もたっていません. 情報通信サービスにアクセスするために重要な ID やパスワード (パスコード) などを含めた個人情報の漏洩や, 不十分な認証システムの欠陥など, 誰でも被害に遭う可能性があります. 情報漏洩を防ぐ最も簡単かつ強固な方法は, 使わないことですが, そういうわけにもいきません. 情報漏洩を含めた通信の秘匿性を守るため, 多くの情報伝達過程が暗号化されています.

3.2　通信によりやり取りされる情報には, メールなどの文字情報, 写真やビデオなどの画像, 動画, 音声や, その他さまざまなデータがあります. 最初にこれらを情報処理機器で扱えるデータ (主に整数の列) に変換します. **符号化**と言います. 符号化されたデータをやり取りする通信の過程で, 情報が他者に漏れないように工夫する技術のひとつが暗号化技術です. 暗号化技術は, 情報を含んだデータを元の情報を抽出し難い形に変換する**暗号化**と, 暗号化されたデータからもとのデータや情報を再現する**復号化**からなります. 情報を伝えたい人 (発信者) は, 情報をデジタルデータに符号化し, 暗号化し, 暗号化されたデータを送信します. 情報を受け取る人 (受信者) は, 受け取った暗号化データを元のデジタルデータに復号化し, そこから情報を取り出します. 通信過程で暗号化されたデータが他者に渡ったとしても, 復号化が難しいか, 復号化に時間がかかる (数十年, 数百年) なら, 伝えたかった情報自体が漏れたことにはなりません. 受信者には簡単な復号化が他者には困難であることが, 暗号化技術の肝になります. 受信者のみが情報を収めた箱を開けるための鍵をもっていればいいのです. 暗号化と復号化はデータを逆向きに処理するので, 基本的には暗号化を行う手続き自体も秘密にしておきたいところです. 昔の暗号化は乱数表 (乱数の表ではなく, ランダムに並んだ数の表) といって, 変換規則の表を発信者と受信者で共有して, 暗号化と復号化を行っていました. 推理小説などで出てくる暗号は, 基本的にこの方法で, 探偵などが乱数表にあたる表を推理するのが暗号解読です. 暗号化と復号化で同じ鍵 (乱数表など) を使う暗号化技術を**共通鍵暗号方式** (**秘密鍵暗号方式**) といいます. 発信者と受信者が出会って鍵を共有できればいいですが, そうでなければ秘密にしたい鍵を通信で送ら

ねばならなくなります. また, 鍵は一組の発信者受信者間で一つずつ必要なので, 通信する相手の数 (通信は人対人だけではないので, 情報機器では膨大な数になる) の鍵を安全に管理しなければなりません. そこで, 暗号化する鍵と復号化する鍵を別にし, 暗号化する鍵を公開し, 復号化する鍵を秘密にする**公開鍵暗号方式**が提唱されました. 公開鍵暗号方式の場合, 暗号化する鍵 (**公開鍵**という) は隠さずに公開するので, その鍵の受け渡しに気を使う必要はありません. 他者に知られないように管理に気を使う, 復号化する鍵 (**秘密鍵**という) は, 自分のものだけなので, 鍵管理の手間も大幅に軽減されます. 公開鍵暗号方式では, 公開鍵など公開した情報から秘密鍵を知られてはいけないので, やや複雑なアルゴリズム (演繹手続き) を使います. そのため, 公開鍵暗号方式の方が共通鍵暗号方式より暗号化復号化の処理に時間がかかります. 共通鍵暗号方式はかなり高速に処理することが可能なのです. そこで, 最初に公開鍵暗号方式で共通鍵暗号方式の鍵を共有し, その後は共通鍵暗号方式で通信を行うことが多くあります.

3.3　公開鍵暗号方式は, 公開した情報から秘密鍵を得るのにかなり時間がかかることを安全性の根拠にしています. 難しくはないけれど答えを得るのに時間がかかる問題で, 公開鍵暗号方式に応用されているものに, 素因数分解と離散対数問題があります。素因数分解を基にした RSA 暗号, 離散対数問題を基にした ElGamal 暗号があります. ElGamal 暗号は, 離散対数問題を内包する様々な代数系を使った方式があり, 現在, 楕円曲線と呼ばれる曲線上の演算を利用した楕円曲線暗号が広く使われています. これら暗号方式の安全性の根拠は, 現在知られている数学上の知識では解読に膨大な時間がかかることであり, 新たな発見などで安全でなくなる可能性があります. また, 計算技術の進展により計算時間が大幅に短縮されることでも, 安全性が揺らぎます. 以下, RSA 暗号を紹介し, この雑文を閉じることにします.

3.4　最初に 2 つの異なる素数 p, q を用意します. $n = pq$ とし, 正の整数 e として $(p-1)(q-1)$ と互いに素で比較的小さな数を適当に選びます. 正の整数 d を $de \equiv 1 \pmod{(p-1)(q-1)}$ を満たすように取ります. ここで, 組 (n, d) が公開鍵で, d が秘密鍵になります. 送りたいメッセージを m (m は n より小さい整数とする. n より大きい場合は分割する) とします. 送信者は m^e を n で割った余り (c とおく) を計算します. この c が送信される暗号文で, この計算が暗号化アルゴリズムです. 暗号文 c を受け取った受信者は c^d を n で割った余りを計算すると, 不思議なことにその余りが元のメッセージ m になります. これが復号化アルゴリズムで, 以上の手続きが RSA 暗号です. 少し難しい話ですが, n $(= pq)$ の余りの世界において, どのような整数 m に対しても $m^{(p-1)(q-1)} \equiv 1 \pmod{pq}$ が成り立ちます. $de = (p-1)(q-1)k + 1$ (k は整数) と表せるので,

$$c^d \equiv (m^e)^d \equiv m^{de} = m^{(p-1)(q-1)k+1} = (m^{(p-1)(q-1)})^k m \equiv 1^k m = m \pmod{pq}$$

となります. これが復号化アルゴリズムがうまく働く根拠です. 秘密鍵 d が計算できれば, 復号化は難しくありません. ここで Euclid の互除法が登場します. Euclid の互除法は, 最大公約数を計算するためのアルゴリズムです. e と $(p-1)(q-1)$ は互いに素 (最大公約数が 1) なので, Euclid の互除法により $ex + (p-1)(q-1)y = 1$ を満たす整数 x, y を求めることができます. この式を法 $(p-1)(q-1)$ の合同式にすると, 左辺の第 2 項は 0 になるので, $ex \equiv 1 \pmod{(p-1)(q-1)}$ となります. この x が知りたかった秘密鍵 d です.

3.5　秘密鍵を知るためには p, q がわかればよい. 公開されている情報は p, q の積 n なので, ここで素因数分解が登場します. 整数 n の素因数分解は $\sqrt{n}$ 以下の奇数で割ってみればわかるので, 計算方法は簡単ですね. では, 例えば 62773913 の素因数分解がわかりますか？ 1689259081189 の素因数分解がわかりますか？ 184260325078815714743569676 3 の素因数分解がわかりますか？ 最後の 28 桁の数ではその平方根の 14 桁の数 42925554752247 まで割り算を実行しなければなりません. 現在の計算機と計算理論, 様々な素因数分解法の開発などにより 100 桁程度の数なら数分から数時間程度で素因数分解できますが, 200 桁, 300 桁と桁数の大きな数の場合, 今の技術では生きている間に素因数分解が終わらない. これが, 素因数分解を基にした RSA 暗号の安全性の根拠なのです.