10.2　公開鍵暗号の例

公開鍵暗号とは、暗号化するための鍵が公開されているものを指す。鍵が公開されているので、だれでも**平文（ひらぶん）**を暗号化することができる。しかし、それならだれでも復号もできてしまうと思うだろうが、現実的には無理である。なぜなら、暗号を平文に戻すためには、公開鍵として与えられた数を素因数分解できなくてはいけないからだ。

ちょっと待って。素因数分解ならこれまでにスクリプトを書いたじゃないの、と思うかもしれない。でも、実際に公開鍵として用いられる数は数百桁の数である。こうなると世界一の高速コンピュータでも何万年かそれ以上かかるのである。だから、“現実的に” 復号は不可能なのだ。そもそも理論的に復号できなければ暗号の意味がないので、理論で復号できても現実的に不可能な要素が何かあればよいのである。

そこで、ここでは公開鍵暗号の雰囲気に少し浸ることを目標に、以前われわれが書いた非力なスクリプトでも素因数分解できるような数で体験しようという計画である。当然、そのような小さな桁の数では実用にならないけれど。いま暗号として

$$\boxed{\text{（暗号）}28705：\quad \text{（鍵）}n=1120213、r=3} \quad \ldots\text{（※）}$$

が与えられたとしよう。詳しい説明はあとからするので、いまは復号の手順を見てもらおう。

D1:	鍵 n の素因数分解	$\to$	$1120213 = 251 \times 4463$
D2:	$\varphi(n)$ の計算	$\to$	$(251-1)\times(4463-1) = 1115500$
D3:	$3t \equiv 1 \pmod{1115500}$ を解く	$\to$	$t = 743667$
D4:	$(\text{暗号})^t \div n$ の余りを求める	$\to$	余り 858373

数回の計算を経て得られた値 858373 が平文である。この平文は、鍵を用いて再び暗号化することができる。その手順は以下のとおりである。

C0:　$(\text{平文})^r \div n$ の余りを求める　$\to$　（暗号）28705

暗号化は実に簡単だ。これは単に **Python3** で計算できるので確認しておこう。

[py script]

```
>>> 858373**3 % 1120213
28705
```

結局、C0 と D1 から D4 までのサイクルで

$$\text{（平文）}\ 858373 \xrightarrow{[C0]} \text{（暗号）}\ 28705 \xrightarrow{[D1][D2][D3][D4]} \text{（平文）}\ 858373$$

となったことが分かるだろう。

さて、なぜこれでうまくいくのか簡単に説明しておこう。暗号と復号には**合同式**が利用されるのだが、合同式についての詳細は省かせてもらう。ざっくり言えば、記号 “≡” を用いて

$$x^r \equiv y \pmod{n} \quad \Leftrightarrow \quad x^r \text{を} n \text{で割った余りが} y$$

という関係にある式のことである。「$x^r - y$ は n で割り切れる」と言い換えても同じことだ。いまの例では

$$858373^3 \equiv 28705 \pmod{1120213} \quad \Leftrightarrow \quad 858373^3 \div 1120213\text{、余り } 28705$$

という計算をして暗号文 y を作ったことになる。

それに対して、与えられた条件（※）を用いて復号するのだが、それは x の方程式

$$x^3 \equiv 28705 \pmod{1120213}$$

を解くことにほかならない。これだけを見れば、ちょっとスクリプトを書いて

[py script]

```
>>> x=1
>>> while 1:
...   if (x**3 - 28705) % 1120213 == 0:
...     print(x)
...     break
...   else:
...     x += 1
...
858373
```

こんな風に復号できると思うだろうが、本当の公開鍵暗号では鍵が数百桁の数であることを忘れないように。場当たり的では無理なのだ。

実は暗号文 y がきちんと平文 x に復号できるのは、今頃になって言うのだが、鍵にあたる数 n がある条件を満たしているからである。それは

1. n は、x と互いに素で、ふたつの素数 p, q を用いて $n = pq > x$ となっていること
2. r は、オイラーの関数 $\varphi(n)$ で得た値と互いに素であること

というものである。オイラーの関数は素数 p, q について $\varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$ となる性質がある。いまの例では $n = 1120213 = 251 \times 4463$ であったから、$1120213 > 858373$ は満た

している。858373 はたまたま素数なので 1120213 とは互いに素でもある。また

$$\varphi(1120213) = \varphi(251) \times \varphi(4463) = 250 \times 4462 = 1115500 = 2^2 \times 5^3 \times 23 \times 97$$

であるから、r としてここに含まれない素因数 3 を選んだ。7 でも 11 でもよいけど小さい方が計算しやすい。

さて、以上の条件を満たした場合、t に関する合同式の方程式

$$rt \equiv 1 \pmod{\varphi(n)}$$

は t の解を持つ。なぜなら r と $\varphi(n)$ は互いに素である—そうなるように r を選んだ—からだ。よって解は $rt - 1 = k\varphi(n)$（k は整数）と書ける。

一方で、暗号化に用いた式 $y \equiv x^r \pmod{n}$ の両辺を t 乗し、いま求めた解の式を利用すると

$$y^t \equiv x^{rt} = x^{1+k\varphi(n)} = x\left(x^{\varphi(n)}\right)^k \pmod{n} \quad \ldots (☆)$$

となる。散歩道が突然急坂になったようだが、単に代入して指数法則を用いているだけであることに注意しよう。そしてオイラーの定理によると

$$x^{\varphi(n)} \equiv 1 \pmod{n}、\quad （x,\ n \text{ は互いに素}）$$

が成り立つので（☆）は

$$y^t \equiv x \pmod{n} \quad \Leftrightarrow \quad y^t \text{を } n \text{ で割った余りが } x$$

ということであり、それは取りも直さず $rt \equiv 1 \pmod{\varphi(n)}$ を満たす t を y に用いれば復号できるということなのである。

実際 $n = 1120213$、$r = 3$ であったから、$\varphi(1120213) = 1115500$ として $3t \equiv 1 \pmod{1115500}$ を解いた $t = 743667$ によって暗号 28705 は

[py script]

```
>>> 28705**743667 % 1120213
858373
```

と復号されるのである。