応用数学６ 2010 年 10 月 27 日

一方向性関数

講師: 安永憲司

1 効率的な計算

この講義において，**アルゴリズム**とは，文字列を入力として受け取り，文字列を出力するプログラムであると定義する．文字列としては，アルファベット $\{0,1\}$ 上の文字列を考えることが多い．アルゴリズムの効率を議論するには，本来は計算モデルを定義しないといけないが，計算モデルに関する細々とした議論を省略するため，ここでは曖昧な定義のまま進めていく．ただし，実際は，Turing 機械と呼ばれる計算モデルを暗に仮定している．

定義 1 (アルゴリズムによる計算) アルゴリズム A が関数 $f : \{0,1\}^* \to \{0,1\}^*$ を**計算する**とは，入力 x に対して，A が $f(x)$ を出力するときである．

定義 2 (アルゴリズムの実行時間) アルゴリズム A の実行時間が $T(n)$ であるとは，任意の入力 x に対して，$A(x)$ が $T(|x|)$ ステップ以内に停止するときである．ここで，$|x|$ は x の長さを表す．

定義 3 (効率的なアルゴリズム) アルゴリズム A が**多項式時間**で実行できるとは，ある定数 c が存在して，実行時間が $T(n) = n^c$ であるときである．多項式時間で実行できるアルゴリズムを**効率的なアルゴリズム**と呼ぶ．

この定義に従うと，n^{100} 時間かかるアルゴリズムも，効率的なアルゴリズムとなってしまう．しかし，以下のような理由で，多項式時間で実行できるアルゴリズムを効率的なものだと言っている．

1. 計算モデルやデータの表現方法に依存しない計算指標である．計算モデルとして，Turing 機械を選んでも，RAM モデルを選んでも，実行時間の差は多項式時間以内の差しかない．
2. 結合に関して閉じている．多項式時間アルゴリズムを多項式回数実行しても，多項式時間で実行できる．
3. 多くの多項式時間アルゴリズムは，実際のところ n^3 時間やそれ以下で実行可能である．
4. 多項式時間で実行できることが知られていない多くの関数は，多項式時間より大きな時間が必要だと考えられている．

定義 4 (確率的アルゴリズム) 確率的アルゴリズム A は，入力 x の他に，独立に投げられた偏りのないコイン投げの結果 $r \in \{0,1\}^*$ を読むことができるアルゴリズムである．コイン投げの結果 r を明示的に表す場合，A の出力は $A(x;r)$ と表す．多項式時間で実行できる確率的アルゴリズムを **PPT (probabilistic polynomial-time) アルゴリズム**と呼ぶ．

以降では，秘匿通信問題等において盗聴をする敵は，PPT アルゴリズムだと仮定する．つまり，敵はコイン投げをすることができるアルゴリズムであるが，多項式時間でしか計算できない．

補足 5 ここで定義した計算の指標は，漸近的なものである．つまり，アルゴリズムへの入力 x の長さが，どんどん大きくなった場合に，アルゴリズムがどのような振る舞いをするかで評価している．現実の世界では，漸近的な振る舞いではなく，具体的な実行時間を評価する必要がある．

2　計算能力の制限された敵と公開鍵暗号

Alice と Bob が秘匿通信を行うことを考える．以前の議論から，

- 暗号化アルゴリズムと復号アルゴリズムは公開情報
- Bob は必ず正しいメッセージを復元
- Eve の計算能力は高い

ことを仮定すると，以下のような考察が導かれた．

1. Bob は Eve の知らない秘密情報をもつ必要がある．
2. Alice は Eve の知らない秘密情報をもつ必要がある．さらに，Alice と Bob は Eve の知らない秘密情報を共有する必要がある．

Eve が PPT アルゴリズムだとすると，二つ目の考察は当てはまらない．また，Shannon の定理も当てはまらなくなる．このことから，Eve の計算能力を制限すると，

1. 鍵長がメッセージ長よりも非常に短い秘密鍵暗号方式の可能性
2. 公開鍵暗号方式の可能性

が出てくる．

公開鍵暗号方式について簡単に説明する．「Bob は Eve の知らない秘密情報をもつ必要がある」という考察は，計算能力を制限したとしても必要である．そこで，Bob は Eve の知らない秘密 SK をもっていると仮定する．このとき，公開情報 PK が存在して，誰でも知ることができるとする．そして，この公開情報 PK を利用して暗号文を作り，秘密情報 SK をもっている Bob だけがメッセージを復元できるような仕組みをもつ暗号方式を公開鍵暗号方式と呼ぶ．

3　一方向性関数

公開鍵暗号を達成するために必要な性質を抜き出してみる:

1. メッセージ m から暗号文 $f(m)$ を計算できる（**効率的計算可能性**）
2. Eve は，メッセージ m を暗号化した $f(m)$ から，m を計算するのは困難（**逆計算の困難性**）
3. 暗号文からメッセージ m が復元可能である（**可逆性**）
4. Bob は SK を使えばメッセージを復元できる（**補助入力による逆計算可能性**）

上記の 1 と 2 の性質（計算できるが逆計算は出来ない）は，関数 f を構成するのに最も問題となる性質である．そのため，1 と 2 の性質をもつ関数 f を**一方向性関数**(one-way function; OWF) と呼ぶ．それに加え，3 の性質をもつ関数を**一方向性置換**(one-way permutation; OWP) と呼び，さらに加え， 4 の性質をもつ関数を**落し戸付き置換**(trapdoor permutation; TDP) と呼ぶ．

3.1　OWF, OWP, TDP の定義

一方向性という性質の定式化には様々な方法が考えられる．まず，以下のシンプルな定義を考える．

定義 6 (最悪時一方向性関数) 関数 $f : \{0,1\}^* \leftarrow \{0,1\}^*$ が最悪時一方向性をもつとは，以下を満たすときで

ある．

1. $f(x)$ を計算する PPT アルゴリズムが存在．
2. 以下の性質を満たす PPT アルゴリズム A は存在しない;

$$\forall x, \Pr[A(f(x)) \in f^{-1}(f(x))] = 1.$$

上記の定義に従うと，ほとんどの x に対しては逆計算ができるような関数も，一方向性をもつことになる．そのような関数が存在したとして，それが暗号に利用できるかどうか現時点ではよくわかっていない．そこで，より強い一方向性を定義する．この定義では，入力 x をランダムに選んだとき，$f(x)$ の逆計算が出来る確率がとても小さいものを一方向性関数と呼ぶ．一方向性関数の定義の前に，まず，**とても小さい**という概念を定式化する．

定義 7 (無視できる関数) 関数 $\varepsilon(n)$ が**無視できる**(negligible) とは，任意の $c > 0$ に対して，$n_0 \in \mathbb{N}$ が存在し，$n \geq n_0$ であるすべての n に対して，$\varepsilon(n) \leq 1/n^c$ が成り立つときである．

簡単にいうと，任意の多項式の逆数よりも漸近的に小さい関数のことを無視できる関数と呼ぶ．例として，$2^{-n}, 2^{-\log^2 n}$ は無視できる関数だが，$0.01, 1/n(= 2^{-\log n}), 1/n^3$ などは無視できる関数ではない．また，逆に，関数 $t(n)$ が無視できないとは，ある定数 c が存在して，無限に続く $\{n_0, n_1, \dots\}$ に対して，$t(n_i) > 1/n_i^c$ が成り立つときである．

なぜ，とても小さいという概念として，上記のような定義を与えたかというと，当てることができる確率が無視できる関数であれば，それを多項式回数繰り返したとしても，その間に 1 回でも当てることができる確率は，(無視できる関数) $\times$ (多項式) = (無視できる関数) が成り立つので，依然として無視できる程度だからである．つまり，PPT アルゴリズムにとって，無視できる関数で表される確率でしかできないことは，出来る確率が 0 であることとほぼ等価なのである．

定義 8 (一方向性関数) 関数 $f : \{0,1\}^* \leftarrow \{0,1\}^*$ が一方向性をもつとは，以下を満たすときである．

1. **効率的計算可能性．** $f(x)$ を計算する PPT アルゴリズムが存在．
2. **逆計算の困難性．** 任意の PPT アルゴリズム A に対して，無視できる関数 ε が存在して，すべての $n \in \mathbb{N}$ に対して，

$$\Pr[f(x') = y \mid x \xleftarrow{R} \{0,1\}^n, y = f(x), x' \leftarrow A(1^n, y)] \leq \varepsilon(n).$$

この定義では，逆計算が出来る確率を無視できる関数以下だとしているが，これを確率 0 に置き換えてもいいような気がしてくる．しかし，そうすると，答えをランダムに予想して出力するアルゴリズムを考えると，確率 $1/2^n$ で逆計算ができたことになり，確率 0 とはならない．つまり，確率 $1/2^n$ で逆計算が出来る PPT アルゴリズムは必ず存在するのである．そこで，無視できる関数程度という定義にすると，確率 $1/2^n$ で逆計算が出来るアルゴリズムは逆計算が出来たとはみなさないことになり，意味のある定義になる．

上の定義において，A の入力には，y だけでなく 1^n を含めている（1^n は，長さ n で要素がすべて 1 の系列 $1 \cdots 1$ を表している）．これは，アルゴリズム A の実行時間を，入力 x の長さに関しての多項式時間に制限したいからである．例えば，関数 $f_{\mathrm{len}}(x) =$ (x の長さの二進表現) を考えると，$x \in \{0,1\}^n$ のとき，$|f_{\mathrm{len}}(x)| = \log n$ であるので，どのような PPT アルゴリズムでも，$f_{\mathrm{len}}(x)$ を与えられて，$f_{\mathrm{len}}^{-1}(f_{\mathrm{len}}(x))$ の要素を出力することはできない（入力長 $\log n$ のとき，長さ n を出力するには指数時間必要）．しかし，$f_{\mathrm{len}}(x)$ が与えられたとき，$f_{\mathrm{len}}^{-1}(f_{\mathrm{len}}(x))$ の要素として，例えば 0^n という長さ n の系列を答えるという簡単なアルゴリズムが存在する．そのため，この関数 f_{len} が一方向性を満たすというのは，我々が考慮したい状況と異

なってくる．そこで，便宜的に，1^n という長さ n の系列を入力に加えている．そうすることで，敵のアルゴリズムを入力長の多項式時間に制限することが意味を持ってくる．

このような n は，今考えたい問題のサイズを表していると言える．そして，暗号システムでは，このような問題のサイズを表すパラメータ n をもつことが多い．例えば，使い捨て鍵暗号であれば，メッセージの長さ n から鍵の長さが決まり，一つのシステムが決定される．そして，より大きな n を選べば，また別のより大きな暗号システムに対応する．このパラメータ n のことを，**セキュリティパラメータ**と呼ぶ．そして，暗号システムにおけるアルゴリズムは n の多項式時間で動作することが想定され，攻撃する敵のアルゴリズムも，n の多項式時間で動作するものを対象としている．さらに，無視できる関数は n に関して無視できる関数を扱う．

上記の一方向性関数の定義は，使い勝手のよい定義ではあるが，実際に暗号技術を構成するには不向きな点がある．例えば，関数の定義域が $\{0,1\}^*$ である点は，現実的には扱いにくい．そこで，一方向性関数族を定義する．

定義 9 (一方向性関数族) 一方向性関数族とは，以下の性質を満たす関数族 $\mathcal{F} = \{f_i : \mathcal{D}_i \leftarrow \mathcal{R}_i\}_{i\in\mathcal{I}}$ である．

1. 関数のサンプルが容易．PPT アルゴリズム Gen が存在し，$\mathsf{Gen}(1^n)$ は $i \in \mathcal{I}$ を出力．
2. 定義域からのサンプルが容易．入力 $i, 1^n$ に対し，$\mathcal{D}_i$ の要素から一様ランダムにサンプルする PPT アルゴリズムが存在．
3. 評価が容易．入力 $i, x \in \mathcal{D}_i$ に対し，$f_i(x)$ を出力する PPT アルゴリズムが存在．
4. 逆計算が困難．任意の PPT アルゴリズム A に対して，無視できる関数 ε が存在して，

$$\Pr[f_i(x') = y \mid i \leftarrow \mathsf{Gen}(1^n), x \xleftarrow{R} \mathcal{D}_i, y = f_i(x), x' \leftarrow A(1^n, i, y)] \leq \varepsilon(n).$$

定義 10 (一方向性置換族) 関数族 $\mathcal{F} = \{f_i : \mathcal{D}_i \leftarrow \mathcal{R}_i\}_{i\in\mathcal{I}}$ が一方向性置換族であるとは，$\mathcal{F}$ が一方向性関数族であり，すべての $i \in \mathcal{I}$ について，f_i が置換である（すべての $y \in \mathcal{R}_i$ は唯一の現像 $x \in \mathcal{D}_i$ をもつ）とき．

定義 11 (落し戸付き置換族) 落とし戸付き置換族とは，以下の性質を満たす関数族 $\mathcal{F} = \{f_i : \mathcal{D}_i \leftarrow \mathcal{R}_i\}_{i\in\mathcal{I}}$ である．

1. すべての $i \in \mathcal{I}$ について，f_i が置換である．
2. 関数のサンプルが容易．PPT アルゴリズム Gen が存在し，$\mathsf{Gen}(1^n)$ は (i, t) を出力．ただし，$i \in \mathcal{I}$ であり，t は落し戸（trapdoor）である．
3. 定義域からのサンプルが容易．入力 $i, 1^n$ に対し，$\mathcal{D}_i$ の要素から一様ランダムにサンプルする PPT アルゴリズムが存在．
4. 評価が容易．入力 $i, x \in \mathcal{D}_i$ に対し，$f_i(x)$ を出力する PPT アルゴリズムが存在．
5. 逆計算が困難．任意の PPT アルゴリズム A に対して，無視できる関数 ε が存在して，

$$\Pr[f(x') = y \mid (i, t) \leftarrow \mathsf{Gen}(1^n), x \xleftarrow{R} D_i, y = f_i(x), x' \leftarrow A(1^n, i, y)] \leq \varepsilon(n).$$

6. 落し戸があれば逆計算が容易．Gen の出力 (i, t) と $y \in \mathcal{R}_i$ を入力として，$f_i^{-1}(y)$ を出力する PPT アルゴリズムが存在．

4 一方向性関数の具体的候補

現在，具体的にある関数が OWF であるということは証明されていない．また，もしそれが証明できたとすると，有名な未解決問題 $P = NP$ 問題を解決することになるので，OWF の存在性証明は，現状では与える

ことが難しい．しかし，その候補と思われている関数はたくさん存在する．ここでは，OWF，OWP，TDP に対して，具体的な候補を紹介する．

4.1 OWF の候補: 素数のかけ算

n ビットで表現される素数 p, q を考える．つまり，p, q は集合

$$\Pi_n = \{x \mid x < 2^n \text{かつ } x \text{ は素数 }\}$$

の要素である．ことのき，関数 $f(p,q) = p \cdot q$ を定義する．つまり，整数のかけ算である．$N = pq$ が与えられたとき，p もしくは q を求める多項式時間アルゴリズムの存在は知られていない．

予想 12 (素因数分解仮定) 任意の PPT アルゴリズム A に対して，無視できる関数 ε が存在して，

$$\Pr[A(N) \in \{p,q\} \mid p \xleftarrow{R} \Pi_n, q \xleftarrow{R} \Pi_n, N = pq\} \leq \varepsilon(n).$$

$N = pq$ の素因数分解を行う単純な方法がある．N を 2 から $\sqrt{N}$ までのすべての整数で，割り切れるかどうかを調べるという方法である．この方法は，実行時間が $\mathrm{O}(\sqrt{N})$ であり，N の多項式時間で実行できている．しかし，上の予想が破られる訳ではない．注意しておきたい点は，上記の予想においてアルゴリズムが入力として受け取るのは N であり，ビットで表現すると $2n$ ビット（p, q がそれぞれ n ビットなので）という点である．すると，$\sqrt{N} \approx \sqrt{2^{2n}} = 2^n$ であるので，実行時間は $\mathrm{O}(2^n)$ になり，入力長の指数時間がかかっているのである．

命題 13 関数族 $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ を以下のように定義する．

$$\begin{aligned} \mathcal{I} &= \mathbb{N} \\ \mathcal{D}_i &= \{(p,q) \mid p, q \in \Pi_i\} \\ f_i(p,q) &= p \cdot q \end{aligned}$$

このとき，素因数分解仮定のもとで，$\mathcal{F}$ は一方向性関数族である．

4.2 OWP の候補: ベキ剰余

まず，群の定義を与える．

定義 14 (群) 群 G とは，二要素間の演算 $\oplus : G \times G \to G$ をともなう要素集合 G で，以下の性質を満たすものである．

1. 閉包性．任意の $a, b \in G$ に対して，$a \oplus b \in G$.
2. 単位元の存在．単位元 $i \in G$ が存在し，任意の $a \in G$ に対して，$a \oplus i = i \oplus a = a$ である．
3. 結合性．任意の $a, b, c \in G$ に対して，$(a \oplus b) \oplus c = a \oplus (b \oplus c)$.
4. 逆元の存在．任意の $a \in G$ に対して，ある $b \in G$ が存在して，$a \oplus b = b \oplus a = i$.

整数 N に対し，$\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ と定義する．このとき，$\mathbb{Z}_N$ は，N を法とする加法演算に関して群をなすことが簡単に確認できる．

また，$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(x, N) = 1\}$ と定義する．ここで，$\gcd(x, y)$ は整数 x と y の最大公約数（greatest common divisor）を表す．以下に示す性質より，$\mathbb{Z}_N^*$ は $\mathbb{Z}_N$ の乗法群と呼ばれる．

命題 15 $\mathbb{Z}_N^*$ は N を法とする乗法演算に関して群をなす．

証明: まず，1 という要素が単位元の性質を満たすことがわかる．また，結合性が成り立つことも自明である．以下では閉包性と逆元の存在を証明する．

閉包性を証明するため，$ab \notin \mathbb{Z}_N^*$ である $a, b \in \mathbb{Z}_N^*$ の存在を仮定する．このとき，$\gcd(a, N) = 1, \gcd(b, N) = 1, \gcd(ab, N) = d > 1$ が成り立っている．最後の式より，d は ab と N の両方を割り切る．このとき，d は a または b を割り切るということがわかるので，$\gcd(a, N) = 1$ または $\gcd(b, N) = 1$ という仮定に矛盾する．

要素 $a \in \mathbb{Z}_N^*$ を考える．定義より，$\gcd(a, N) = 1$ であるので，Euclid の互除法を使うと，$ax + Ny = 1$ を満たす整数 x, y を求めることができる．このとき，$ax = 1 \mod N$ であることがわかり，x が逆元となる．また，$ax + Ny = 1$ であることから，$\gcd(x, N) = 1$ であり，$x \in \mathbb{Z}_N^*$ であることがわかる． □

定理 16 (Fermat の小定理) 素数 p と $x \in \mathbb{Z}_p^*$ に対して，$x^{p-1} = 1 \mod p$.

また，$x \in \mathbb{Z}_N^*$ に対して，$x^k = 1 \mod N$ となる最小の k を，$\mathbb{Z}_N^*$ における x の位数と呼ぶ．一般的には，位数が $N-1$ よりも小さい要素 $x \in \mathbb{Z}_N^*$ は存在する．例えば，3 以上の素数 p に対して，$(p-1)^2 = (-1)^2 = 1 \mod p$ であるので，$\mathbb{Z}_p^*$ における $p-1$ の位数は 2 である．

事実 17 素数 p に対して，$\mathbb{Z}_p^*$ は位数 $p-1$ の要素を少なくとも一つ含む．

そして，$\mathbb{Z}_p^*$ において位数が $p-1$ である要素のことを，$\mathbb{Z}_p^*$ の**生成元**と呼ぶ．生成元 $g \in \mathbb{Z}_p^*$ に対して，$\{g, g^2, g^3, \ldots, g^{p-1}\} = \mathbb{Z}_p^*$ が成り立つ点に注意．つまり，g のベキ乗を取っていくと，$\mathbb{Z}_p^*$ のすべての要素が現れる．

話を OWP に戻す．候補となる関数は，$f_{p,g}(x) = g^x \bmod p$ である．ここで，p は素数，g は $\mathbb{Z}_p^*$ の生成元，$x \in \mathbb{Z}_p \setminus \{0\}$ である．この関数が置換になっている理由は，g が生成元であり，素数 p に対して，$\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ が成り立つからである．

関数 f の順方向演算，つまり，p, g, x が与えられて $g^x \bmod p$ を求めることは，効率的に可能である．単純に x 回 g の掛け算を行なおうとすると，指数時間かかる．そこで，x を二進展開して $x = 2^\ell x_\ell + 2^{\ell-1} x_{\ell-1} + \cdots + 2x_1 + x_0$, $x_i \in \{0, 1\}$ と表すと，

$$\begin{aligned} g^x \bmod p &= a^{2^\ell x_\ell + 2^{\ell-1} x_{\ell-1} + \cdots + 2x_1 + x_0} \bmod p \\ &= \prod_{i=0}^{\ell} x_i a^{2^i} \bmod p \end{aligned}$$

となる．$g^{2^i} \bmod p$ をすべての $i \in \{0, 1, \ldots, \ell\}$ に対して計算するには，平方演算を ℓ 回繰り返せばよい．各 $g^{2^i} \bmod p$ が求まれば，最大 ℓ 回の掛け算を行うことで，$g^x \bmod p$ は求まる．p を法とした掛け算や平方は $\mathrm{O}(\log^2 p)$ 時間で可能なので，合計で $\mathrm{O}((\log^2 p) \cdot \ell) = \mathrm{O}(\log^3 p)$ 時間で計算が可能である．

そして，$g, p, g^x \bmod p$ が与えられたときに，x を求める問題は離散対数問題として知られており，効率的に計算できないと考えられている．乗法群 $\mathbb{Z}_p^*$ の生成元の集合を G_p とする．

予想 18 (離散対数仮定) 任意の PPT アルゴリズム A に対して，無視できる関数 ε が存在して，

$$\Pr[A(y) = x \mid p \xleftarrow{R} \Pi_n, g \xleftarrow{R} \mathrm{G}_p, y = g^x \bmod p] < \varepsilon(n).$$

命題 19 関数族 $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ を以下のように定義する．

$$\begin{aligned} \mathcal{I} &= \{(p, g) \mid p \in \Pi_n, g \in \mathrm{G}_p\} \\ \mathcal{D}_i &= \{x \mid x \in \mathbb{Z}_p \setminus \{0\}\} \\ f_{p,g}(x) &= g^x \bmod p \end{aligned}$$

このとき，離散対数仮定のもとで，$\mathcal{F}$ は一方向性置換族である．

4.3 TDP の候補: RSA

整数 N に対して，N と互いに素である N 以下の整数の数は，Euler の φ 関数と呼ばれ，$\varphi(N)$ で表される．$\mathbb{Z}_N^*$ に含まれる要素の数は $\varphi(N)$ である．また，素数 p に対して，$\varphi(p) = p - 1$ であり, 素数 p, q に対して，$\varphi(pq) = (p-1)(q-1)$ である．

定理 20 (Euler) 任意の $a \in \mathbb{Z}_N^*$ に対して，$a^{\varphi(N)} = 1 \mod N$.

この Euler の定理は，前節で紹介した Fermat の小定理の一般化である．

ここで考える TDP の候補である RSA 関数とは，$f_{N,e}(x) = x^e \bmod N$ である．ただし，N は二つの素数 p, q の積（つまり，$N = pq$）であり，$x \in \mathbb{Z}_N^*$, $e \in \mathbb{Z}_{\varphi(N)}^*$ である．

まず，この関数が置換であることを示す．e は群 $\mathbb{Z}_{\varphi(N)}^*$ の要素であるので，逆元 d が存在して，$ed = 1 \bmod \varphi(N)$ を満たす．ここで，逆計算写像 $g_{N,e}(x) = x^d \bmod N$ を定義する．すると，任意の $x \in \mathbb{Z}_N^*$ に対して，

$$\begin{aligned} g_{N,e}(f_{N,e}(x)) &= g_{N,e}(x^e \bmod N) \\ &= (x^e \bmod N)^e \bmod N \\ &= x^{ed} \bmod N \\ &= x^{c\varphi(N)+1} \bmod N \end{aligned}$$

となる．ただし，c はある定数．Euler の定理より，$x^{\varphi(N)} = 1 \mod N$ であるため，上記の式は，

$$\begin{aligned} &= x^{c\varphi(N)} \cdot x \bmod N \\ &= x \bmod N \end{aligned}$$

となる．また，$y = f_{N,e}(x) = x^e \bmod N$ としたとき，$x \in \mathbb{Z}_N^*$ であり，$\mathbb{Z}_N^*$ が乗法群であることから，x のベキ乗を計算した値である y も $\mathbb{Z}_N^*$ の要素である．よって，$f_{N,e} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ であり，RSA 関数は置換である．

関数 $f_{N,e}$ の順方向演算は，前節での議論と同様，効率的に計算できる．

また，上記の議論から，$f_{N,e}(x)$ が与えられたとき，e の逆元 d が分かれば，関数 $g_{N,e}$ を使うことで，x を逆計算することができる．したがって，この d を落し戸 (trapdoor) として使うことができる．

RSA 関数での f と前節のベキ剰余関数での f との違いは，ベキ剰余では，指数部分を秘密にして，基底を公開しているのに対し，RSA では，基底を秘密にして，指数部分を公開している点である．

そして，RSA 関数の逆計算は困難だと信じられている．逆計算を行なうための一つの方法は，N を素因数分解して p と q を求めることである．すると，$\varphi(N)$ が求まり，群 $\mathbb{Z}_{\varphi(N)}^*$ における e の逆元も計算できる．しかし，素因数分解は効率的に出来ないと信じられているため，この方法は適用できない．

予想 21 (RSA 仮定) 任意の PPT アルゴリズム A に対して，無視できる関数 ε が存在して，

$$\Pr[A(N, e, y) = x \mid p, q \xleftarrow{R} \Pi_n, N = pq, e \xleftarrow{R} \mathbb{Z}_{\varphi(N)}^*, x \xleftarrow{R} \mathbb{Z}_N^*, y = x^e \bmod N] < \varepsilon(n).$$

命題 22 関数族 $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ を以下のように定義する.

$$\begin{aligned} \mathcal{I} &= \{(N, e) \mid p, q \xleftarrow{R} \Pi_n, N = pq, e \xleftarrow{R} \mathbb{Z}^*_{\varphi(N)}\} \\ \mathcal{D}_i &= \{x \mid x \in \mathbb{Z}^*_N\} \\ f_{N,e}(x) &= x^e \bmod N \end{aligned}$$

このとき，RSA 仮定のもとで，$\mathcal{F}$ は落し戸付き置換族である.