I240 暗号理論 2019
一方向性関数、擬似乱数生成器、擬似ランダム関数 (1)

2019/10/30, 11/1 講師 藤崎

1 はじめに

現代暗号理論の基礎概念である一方向性関数 (one-way function)、擬似乱数生成器 (pseudo-random (number) generator)、擬似ランダム関数 (pseudo-random function) を中心に現代暗号理論の根底を流れる考え方や証明法に慣れることを目的とする。一方向性関数、擬似乱数生成器、擬似ランダム関数の存在は全て等価である。なかでも特に知っておくべきは、Goldreich-Levin のハードコア述語の構成法、擬似乱数生成器の伸長定理、Goldreich-Golidwasser-Micali の擬似乱数生成器から擬似ランダム関数の構成法である。

2 よく使われる用語と定義

■確率的アルゴリズム アルゴリズム M が確率的であるとは、M の出力 y が入力 x と M の内部コイン投げ $r \in \{0,1\}^*$ によって決定される時に言う。すなわち、M は二入力関数として $M(x,r) = y$ と書ける。任意の（サイズ n の）入力 $x \in \{0,1\}^n$ に対して、（確率的）アルゴリズム M が常に時間 $t(n)$ で停止する時、M を $t(n)$-**時間（確率的）アルゴリズム**と呼ぶ。$M(x)$ は M の内部コイン投げによって定められる確率変数と思って良いので，$\Pr[M(x) = y]$ が定義できる。M の動作時間が限定されている時（例えば，$t(n)$-時間確率的アルゴリズムの時）内部コイン投げは有限なので一般性を失うことなく $r \in \{0,1\}^{m(n)}$ として、

$$\Pr[M(x) = y] = \frac{\#\{r \in \{0,1\}^{m(n)} | M(x,r) = y\}}{2^{m(n)}}$$

と考えてよい。ここで，$t(\cdot), m(\cdot)$ は，$\mathbb{N} \to \mathbb{N}$ の関数。**確定的** (deterministic) **なアルゴリズム**は内部コインの無い確率的なアルゴリズムの特別な場合として定義できる。

関数 $t : \mathbb{N} \to \mathbb{R}^+$ が**多項式制限である** (polynomial-bound) とは、十分大きな全ての n に対して，t がある多項式により上から抑えられることを言う。すなわち、ある $c > 0$、ある正整数 n_0 が存在して、全ての $n \geq n_0$ に対して $t(n) \leq n^c$ が成り立つことを言う。これを、しばしば $t(n) = O(\mathrm{poly}(n))$ と書く。

アルゴリズム M が**確率的多項式時間アルゴリズム** (probabilistic polynomial-time (PPT) algorithm) であるとは、M が $t(n)$-時間（確率的）アルゴリズムで，$t(n)$ がある多項式制限であることを言う。特に確定的多項式時間アルゴリズムであることを強調したいときは、deterministic polynomial-time (DPT) algorithm と書く。

注釈 1 我々は確率的アルゴリズムを主に扱うので、deterministic アルゴリズムをその対比で確定的アルゴリズムと呼ぶことにする（確定的（deterministic）⇔ 確率的（probabilistic））。この訳語は非標準的で，通常非決定性アルゴリズムとの対比から決定性アルゴリズムと良く訳されている（決定性（deterministic）⇔ 非決定性（non-deterministic））。

■一様/非一様なアルゴリズム Turing 機械のように, 任意の長さの入力を受け付けるアルゴリズムを**一様** (uniform) なアルゴリズムという。一方, ある固定の長さの入力に対してのみ動作するアルゴリズムを**非一様** (non-uniform) なアルゴリズムと呼ぶ。M_n を長さ n の問題に対してのみ動く確率的アルゴリズムとして $M := (M_1, \ldots, M_n, \ldots) = (M_n)_{n>0}$ のように非一様 (non-uniform) なアルゴリズムの族 M を定義する。非

一様な多項式時間アルゴリズム（の族）の力は、多項式サイズの回路（の族）と等しい。非一様な多項式時間アルゴリズムの族は、多項式時間制限の Turing 機械 T とその T に長さ n ごとに与えられる（n の）多項式の長さの補助入力 z_n の組の族 $(T, z_n)_{n>0}$ であらわすことが出来る。

暗号の世界では一般に敵 (adversary) を非一様な確率的多項式時間アルゴリズムの族とみなす。以下、敵、識別器などは特に断らなくても非一様なアルゴリズム（の族）とみなす。

■無視できる関数　関数 $\epsilon : \mathbb{N} \to \mathbb{R}^+$ が**無視できる** (negligible) とは，ϵ がいかなる多項式（の逆数）より早く 0 に収束する時に言う：すなわち，任意の $c > 0$ に対して，ある正整数 n_0 が存在して，全ての $n \geq n_0$ に対して $\epsilon(n) \leq n^{-c}$ が成り立つことを言う。別の言い方をすると、$\epsilon(n)$ が無視できる関数とは $\epsilon(n) = n^{-\omega(1)}$ [*1]。

■統計的識別不可能性　$X = (X_n)_{n>0}$ を $\{0,1\}^n$ 上に値をとる確率変数 X_n の族とする。二つの確率変数の族 $X = (X_n)_{n>0}, Y = (Y_n)_{n>0}$ の統計的距離を

$$\mathsf{Dist}(X_n, Y_n) := \frac{1}{2} \sum_{a \in \{0,1\}^n} |\Pr_{X_n}[X_n = a] - \Pr_{Y_n}[Y_n = a]|$$

と定義する。ある無視できる関数 $\epsilon(n)$ があって, 十分大きな全ての n で，$\mathsf{Dist}(X_n, Y_n) \leq \epsilon(n)$ が成立する時，二つの確率変数の族 X, Y は, **統計的識別不可** (statistically indistinguishable) と言う（ $X \approx_s Y$ と書く)。

特に, 全ての n に対して, $\mathsf{Dist}(X_n, Y_n) = 0$ の時, X, Y は, 完全識別不可 (perfectly indistinguishable) と言う（ $X \equiv Y$)。

■計算量的識別不可能性　$X = (X_n)_{n>0}$ を $\{0,1\}^n$ 上に値をとる確率変数 X_n の族とする。$D : \{0,1\}^* \to \{0,1\}$ を（非一様な）$t(n)$-時間確率的アルゴリズムの族とする。二つの確率変数の族 $X = (X_n)_{n>0}, Y = (Y_n)_{n>0}$ に対して，D の識別能力を

$$\mathsf{Dist}_D(X_n, Y_n) := |\Pr_{D,X_n}[D(X_n) = 1] - \Pr_{D,Y_n}[D(Y_n) = 1]|$$

で定義する。ここで $\Pr_{D,X_n}[\cdots]$ とは，確率が D, X_n にのみ依存することの強調であり，特に確率がどのような分布に依存するか明らかな時は省略する。二つの確率変数族 X, Y が, (t, ϵ)-識別不可 (indistinguishable) とは, 全てのの $t(n)$-時間 PPT D に対して, 十分大きな全ての n で, $\mathsf{Dist}_D(X_n, Y_n)(n) \leq \epsilon(n)$ が成立することである（$X \approx_{(t,\epsilon)} Y$ と書く)。全ての多項式制限の関数 $t(n)$ に対して, ある無視できる関数 $\epsilon(n)$ が存在して, $X \approx_{(t,\epsilon)} Y$ のとき, 確率変数の族 X と確率変数の族 Y は**計算量的識別不可** (computationally indistinguishable) という（ $X \approx_c Y$ と書く)。

注釈 2　$\mathsf{Dist}_D(X_n, Y_n)$ とは確率変数 $D(X_n)$ と $D(Y_n)$ の統計的距離 $\mathsf{Dist}(D(X_n), D(Y_n))$ に等しい。

$$\begin{aligned}
\mathsf{Dist}_D(X_n, Y_n) &= |\Pr_{D,X_n}[D(X_n) = 1] - \Pr_{D,Y_n}[D(Y_n) = 1]| \\
&= \frac{1}{2} \sum_{a \in \{0,1\}} |\Pr_{D(X_n)}[D(X_n) = a] - \Pr_{D(Y_n)}[D(Y_n) = a]| \\
&= \mathsf{Dist}(D(X_n), D(Y_n)).
\end{aligned}$$

[*1] $f = \omega(g)$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$

3 一方向性関数とハードコア述語

3.1 一方向性関数 (One-Way Function)

関数 $f : \{0,1\}^* \to \{0,1\}^*$ を考える。(a) 任意の x に対して，x の長さの多項式制限時間で $f(x)$ を計算する確定的アルゴリズムが存在する一方で，(b) $f^{-1}(y)$ を計算するのが平均的に難しい時，f を**一方向性関数** (one-way function) と呼ぶ。

より正確には, $t(n)$-時間アルゴリズムの敵 A の f の逆計算に関する成功確率を,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{owf}}_{A,f}(n) &:= \Pr_{x \leftarrow_{\mathrm{R}} \{0,1\}^n, A}[A(f(x)) \in f^{-1}(f(x))] \\ &= \Pr_{U_n, A}[A(f(U_n)) \in f^{-1}(f(U_n))]\end{aligned}$$

と定義する。ここで, U_n は $\{0,1\}^n$ 上の一様分布に従う確率変数である。A の成功確率は，A の内部コインと入力 x の $\{0,1\}^n$ 上の一様分布に依存する（つまり，A による f の逆計算の平均的な難しさを定義している)。

条件 (a) を満足する関数 $f : \{0,1\}^* \to \{0,1\}^*$ が，任意の $t(n)$-時間確率的アルゴリズム A に対して，十分大きな全ての n で $\mathsf{Adv}^{\mathsf{owf}}_{A,f}(n) \leq \epsilon(n)$ が成り立つ時，関数 f を，$(t(n), \epsilon(n))$-**一方向性関数**（又は，$(t(n), \epsilon(n))$-OWF ）と呼ぶ。

特に，$t(n)$ が多項式制限で，$\epsilon(n)$ が無視できる関数の時，関数 f を**一方向性関数**と呼ぶ。

現実に我々が知っている一方向性関数の候補はこの定義にやや合致しない。そこで, 代わりに一方向性関数の族 $F = \{f_i\}_{i \in I}$ を次のように定義する。(a) 全ての インデックス $i \in I_n (= I \cap \{0,1\}^n)$ に対して, 関数 $f_i : \{0,1\}^n \to \{0,1\}^*$ が定義される。f_i は多項式時間で計算できる（一様な確定的多項式時間アルゴリズムで f_i を実装できる)。

(b) $t(n)$-時間アルゴリズムの敵 A の成功確率を,

$$\mathsf{Adv}^{\mathsf{owf}}_{A,F}(n) := \Pr[i \leftarrow_{\mathrm{R}} I_n; x \leftarrow_{\mathrm{R}} \{0,1\}^n : A(f_i(x)) \in f_i^{-1}(f_i(x))]$$

と定義したとき, 任意の $t(n)$-時間確率的アルゴリズム A に対して，十分大きな全ての n で $\mathsf{Adv}^{\mathsf{owf}}_{A,F}(n) \leq \epsilon(n)$ が成り立つ。条件 (a), (b) を満たす関数族 F を，$(t(n), \epsilon(n))$-**一方向性関数族**と呼ぶ。特に，$t(n)$ が多項式制限で，$\epsilon(n)$ が無視できる関数の時，関数 f を**一方向性関数族**と呼ぶ。以下では, 一方向性関数と言っていても実は一方向性関数族のこともある。

3.2 一方向性落し戸付き関数族 (One-Way Trapdoor Function Family)

一方向性関数族 F の各インデックス i にたいして, f_i^{-1} を多項式時間で逆計算できるアルゴリズムが存在するとき, 関数族 F を **一方向性落し戸付き関数族** ((one-way) trapdoor function family) と呼ぶ。f_i^{-1} を f_i の落とし戸 (trapdoor) と呼ぶ。

注釈 3 族でない一方向性関数には落とし戸は存在しない。

注釈 4 RSA 関数は, $i = (n, e)$, $f_i(x) = x^e \bmod n$, $f^{-1}(y) = y^d \bmod n$ によって定義される一方向性落とし戸関数（置換）族の例になっている。ここで n を $n = pq$ なる, 大きな素数 p, q の積, $\phi(\cdot)$ をオイラー関数とする。$e \geq 3$ を, $e \in (\mathbb{Z}/\phi(n)\mathbb{Z})^\times$ なる整数, d を $e^{-1} \in (\mathbb{Z}/\phi(n)\mathbb{Z})^\times$ （すなわち $de = 1 \pmod{\phi(n)}$ ）なる整数とする。

3.3 一方向性関数のハードコア述語 (Hard Core Predicate for One-Way Function)

f が一方向性関数とは，$x \in f^{-1}(y)$ を求めることが難しいと言うことであるが，これは例えば，x のいかなる部分情報も求めることが難しいという意味では無いことは次の例から明らかである：任意の一方向性関数 f から，別の一方向性関数 g を $g(x,z) := (f(x),z)$ で定義する（x,z はビット列)。明らかに，g は一方向性関数であるが，$x' = (x,z)$ の部分ビット列 z を求めるのは簡単である。入力 x に関する情報の内，$f(x)$ を漏らしていても計算することが難しい一ビット $b(x)$ を $f(x)$ のハードコア (hard core) という。以下，もう少し厳密に定義する。

関数 $b : \{0,1\}^* \to \{0,1\}$ を考える。関数 f, b に対する $t(n)$-時間アルゴリズム D の成功確率を

$$\begin{aligned}\mathsf{Adv}^{\mathsf{hc}}_{D,f,b}(n) &:= \mathsf{Dist}_D\Big((f(U_n),b(U_n)),(f(U_n),U_1)\Big)\\ &= |\Pr_{D,U_n}[D((f(U_n),b(U_n))) = 1] - \Pr_{D,U_n,U_1}[D((f(U_n),U_1)) = 1]|\end{aligned}$$

と定義する。ここで，U_n, U_1 はそれぞれ $\{0,1\}^n, \{0,1\}$ 上の一様分布に従う独立な確率変数である。$X_n := (f(U_n),b(U_n))$，$Y_n := (f(U_n),U_1)$ によって確率変数族 $X = (X_n)_{n>0}$，$Y = (Y_n)_{n>0}$ を定義する。$X \approx_{(t,\epsilon)} Y$ の時，関数 b を，関数 f の $(t(n),\epsilon(n))$-**ハードコア**（又は，$(t(n),\epsilon(n))$-HC ）と呼ぶ。$X \approx_c Y$ のとき（すなわち, いかなる PPT アルゴリズム D を考えても, ϵ が無視できる関数になる）, 関数 b を f の **ハードコア述語** (hard core predicate) と呼ぶ。

任意の一方向性関数から、新たな一方向性関数とそのハードコア述語を作り出せることが知られている。

■Goldreich-Levin の構成 [GL89]　一方向性関数 f に対して、関数 $g : \{0,1\}^* \to \{0,1\}^*$ を $g(x,r) := (f(x),r)$ と定義する。ただし g の入力 $(x,r) \in \{0,1\}^{n'}$ に対して、$n' = 2n$ のとき、$x, r \in \{0,1\}^n$。一方、$n' = 2n+1$ のとき, $x \in \{0,1\}^n$, $r \in \{0,1\}^{n+1}$ とする。さらに関数 $b : \{0,1\}^* \to \{0,1\}$ を $b(x,r) := \langle x,y\rangle = \sum x_i \cdot r_i \bmod 2$ により定義する。ここで x_i はビット列 x の i 番目のビットを意味する。ただし $n' = 2n+1$ のときは, $x_{n+1} = 0$ と定義する。

この時，次の定理が成り立つ。

定理 5（Goldreich-Levin [GL89])　任意の一方向性関数 f に対して，上記のように関数 g を定義する。この時関数 $b : \{0,1\}^* \to \{0,1\}$ は，g のハードコアである。

注釈 6　Goldreich-Levin の定理の証明のキモは, Hadamard 符号 (Walsh-Hadamard 符号) が local list-decodable 符号であることを示すことと同じである。

■他の構成法　他の構成法として，例えば，素因数分解の困難性を仮定した構成法，離散対数問題の困難性を仮定した構成法などが知られている。

4 擬似乱数生成器 (Pseudo Random Generator)

■擬似乱数生成器の定義　大まかに言うと，擬似乱数生成器とは，短い乱数列をその乱数列より長い（一見）ランダムな乱数列に伸ばす確定的アルゴリズムのことである。

任意の n ビット入力を $(n+l)$ ビット列に伸長する（n の多項式時間制限の）確定的アルゴリズム（又は関数）$G : \{0,1\}^* \to \{0,1\}^*$ を考える。入力を n ビットに制限すると，G は、$G : \{0,1\}^n \to \{0,1\}^{n+l}$ となる。

この G に対する $t(n)$-時間確率的アルゴリズム（族）D の識別成功確率を

$$\begin{aligned}\mathsf{Adv}^{\mathsf{prg}}_{D,G}(n) &:= \mathsf{Dist}_D\Big(G(U_n),\, U_{n+l}\Big)\\ &= |\Pr_{D,U_n}[D(G(U_n))=1] - \Pr_{D,U_{n+l}}[D(U_{n+l})=1]|\end{aligned}$$

と定義する。

確率変数族 $(G(U_n))_{n>0}$ が $(U_{n+l})_{n>0}$ から $(t(n),\epsilon(n))$-識別不可の時、G を $(t(n),\epsilon(n))$-**擬似乱数生成器**（又は、$(t(n),\epsilon(n))$-PRG）と呼ぶ。

特に $(G(U_n))_{n>0}$ が $(U_{n+l})_{n>0}$ から計算量的識別不可の時（すなわち $(G(U_n))_{n>0} \overset{\mathrm{c}}{\approx} (U_{n+l})_{n>0}$）、$G$ を**擬似乱数生成器**、確率変数 $G(U_n)$ を**擬似乱数**と呼ぶ。

4.1 擬似乱数生成器の伸長定理

固定された n に対して，$G^{(l)}:\{0,1\}^n \to \{0,1\}^{n+l}$ で自然に定義される関数 $G^{(l)}:\{0,1\}^* \to \{0,1\}^*$ を n-ビット列を $(n+l)$-ビット列に伸長する擬似乱数生成器とする。l を擬似乱数生成器の伸長度と呼ぶ。擬似乱数生成器の伸長度 $l := l(n)$ が n の多項式制限である限り二つの異なる伸長度を持つ擬似乱数生成器に本質的な違いは無い。

■擬似乱数生成器の伸長　$G^{(1)}$ を上記の記法に従って伸長度 1 の擬似乱数生成器とする。（入力 n を固定して）関数 $G^{(l)}:\{0,1\}^n \to \{0,1\}^{n+l}$ を次のように再帰的に定義する。

$$G^{(l)}(x) := \quad G^{(1)}(x)[1] \,||\, G^{(l-1)}\Big(G^{(1)}(x)[2,\ldots,n+1]\Big).$$

ここで，ビット列 x に対して $x[i]$ は x の i 番目のビットを，$x[i,\ldots,j]$ は x の i から j ビット目までのビット列を意味する。上記 $G^{(l(n))}$ は下記のようにも書き表せる。

$$G^{(l(n))}(x) = \sigma_1 \cdots \sigma_{l(n)} x_{l(n)}.$$

ただし、$x_0 := x$, $\sigma_i := G^{(1)}(x_{i-1})[1]$, $x_i := G^{(1)}(x_{i-1})[2,\ldots,n+1]$ である。

$G^{(1)}$ が擬似乱数生成器である時，$l(n)$ が多項式制限であるならば，$G^{(l(n))}$ も擬似乱数生成器であり，以下の定理が成立する。

定理 7　$G^{(1)}$ を $(t(n),\epsilon(n))$-擬似乱数生成器とする。すると、上記の方法で構成された $G^{(l(n))}$ は $(t'(n),\epsilon'(n))$-擬似乱数生成器であり、

$$\begin{aligned}t'(n) &\le t(n) - (l(n)-1)T_G(n) - T_R(l(n)-1),\\ \epsilon'(n) &\le l(n)\cdot\epsilon(n).\end{aligned}$$

を満足する。ただし、$G^{(1)}:\{0,1\}^n \to \{0,1\}^{n+1}$ を $T_G(n)$-時間確定アルゴリズム、n ビットの乱数列の生成時間を $T_R(n)$ で表す。

(証明)

■証明の概要　$G^{(l(n))}$ に対する t'-時間識別アルゴリズム D' を考える。次に、D' をオラクルに使う $G^{(1)}$ に対する識別アルゴリズムを考える。$G^{(1)}$ に対する t-時間識別アルゴリズムの識別確率は仮定により ϵ が上限であるから、結局 D' の識別アルゴリズムの上限を抑えることができる。

■ハイブリットアーギュメント　$X_i\,(0 \le i \le l)$ を，$\{0,1\}^{n+l(n)}$ 上に値を取る，$X_i = U_{l-i}||G^{(i)}(U_n)$ と定義する確率変数とする（但し，$G^{(0)}(U_n) = U_n$）。

$$p_i := \Pr_{D',X_i}[D'(X_i) = 1]$$

と定義する。特に $X_0 = U_{l(n)}\,||\,U_n$，$X_l = G^{(l(n))}(U_n)$ であるので、定義より

$$\mathsf{Adv}^{\mathsf{prg}}_{D',G^{(l)}}(n) = |p_0 - p_{l(n)}|$$

である。三角不等式から

$$|p_0 - p_{l(n)}| \le \sum_{i=1}^{l(n)} |p_{i-1} - p_i|$$

であることに注意する。次に、$|p_{i-1} - p_i|$ の確率の上限を抑えるため、次のような $G^{(1)}$ の識別アルゴリズム S^i を考える。

■$G^{(1)}$ の識別アルゴリズム S^i　S^i は $n+1$ ビット列 $u \in \{0,1\}^{n+1}$ を入力としてとる（u は、一様ランダムな $n+1$ ビット列、または $G^{(1)}$ の出力である）。

[**アルゴリズム** S^i]**:**

1. $(l(n)-i)$-ビットの一様ランダムな乱数列 $r \in \{0,1\}^{l(n)-i}$ を選ぶ。
2. ビット列 $\alpha := r\,||\,u[1]\,||\,G^{(i-1)}(u[2,\ldots,n+1])$ を作る（$u = u[1]\,||\,u[2,\ldots,n+1]$）。
3. $G^{(l(n))}$ に対する識別器 D' にビット列 α を入力する（α が $(n+l(n))$-ビット列であることに注意）。
4. $D'(\alpha)$ の出力 b を S^i の出力として出力する。

■識別確率の解析　S^i は、$G^{(1)}$ に対する識別器であるので

$$\mathsf{Adv}^{\mathsf{prg}}_{S^i,G^{(1)}}(n) = |\Pr_{U_{n+1}}[S^i(U_{n+1}) = 1] - \Pr_{U_n}[S^i(G^{(1)}(U_n)) = 1]|.$$

ここで、

$$G^{(i)}(x) = \quad G^{(1)}(x)[1]\,||\,G^{(i-1)}\Big(G^{(1)}(x)[2,\ldots,n+1]\Big)$$

に注目すると、u が $(n+1)$-ビットの乱数列の時は $\alpha = X_{i-1}$ であり、一方 $u = G^{(1)}(x)$ (x は n-ビットの乱数列) の時は $\alpha = X_i$ である。よって、

$$p_{i-1} = \Pr[S^i(U_{n+1}) = 1], \quad p_i = \Pr[S^i(G^{(1)}(U_n)) = 1]$$

が成り立つので、$\mathsf{Adv}^{\mathsf{prg}}_{S^i,G^{(1)}}(n) = |p_{i-1} - p_i|$。

S^i の走行時間を t で制限すると、擬似乱数生成器 $G^{(1)}$ の仮定から、

$$\mathsf{Adv}^{\mathsf{prg}}_{S^i,G^{(1)}}(n) = |p_{i-1} - p_i| \le \epsilon(n)$$

となる。S^i の走行時間を t に制限するには、$(l-i)$-ビット乱数列 r、D' の走行時間 t'、$G^{(i-1)}$ の実行時間 $(i-1)T_G(n)$ から、

$$t(n) \ge t'(n) + (l(n)-i)T_G(n) + T_R(l(n)-i)$$

であればよい。$S^1,\ldots,S^l$ の全ての制限時間を t で抑えるには、

$$t(n) \ge t'(n) + (l(n)-1)T_G(n) + T_R(l(n)-1)$$

とすればよい。このとき、$S^1, \ldots, S^l$ の識別確率は全て ϵ で抑えられるから、

$$\mathsf{Adv}^{\mathsf{prg}}_{S^i, G^{(1)}}(n) \leq l(n) \cdot \epsilon(n)$$

が成り立つ。よって、$G^{(l)}$ の識別器 D' は、走行時間が

$$t'(n) \leq t(n) - (l(n)-1)T_G(n) - T_R(l(n)-1)$$

のとき、識別確率は

$$\epsilon'(n) \leq l(n) \cdot \epsilon(n)$$

で抑えられることが示された。 ■

系 8 $f : \{0,1\}^n \to \{0,1\}^n$ を一方向性置換とする。その時、

$$G^{(l)}(x) := f^{(l)}(x) \,||\, b(x) \,||\, b(f(x)) \,||\, \cdots\cdots \,||\, b(f^{(l-1)}(x))$$

が擬似乱数生成器である。ただし、$f^{(i+1)}(x) := f(f^{(i)}(x))$ と定義する（$f^{(1)}(x) = f(x)$）。

（ヒント）定理 7 と同様に証明できる。ハイブリッドアーギュメントがうまく構成できるような乱数列を探してくればよい。$X_0, \ldots, X_l \in \{0,1\}^n$ を一様分布に従う独立な確率変数とし、確率変数 $Y_0, \ldots, Y_l \in \{0,1\}^{n+l}$ を $Y_0 := X_0 \,||\, U_l$, $Y_1 := f(X_1) \,||\, U'_l$, $Y_2 := f^{(2)}(X_2) \,||\, U_{l-1} \,||\, b(X_2)$, ..., $Y_l := f^{(l)}(X_l) \,||\, b(X_l) \,||\, \cdots \,||\, b(f^{(l-1)}(X_l))$ と定義する。すると、特に

$$\begin{array}{rccccccc} Y_i = & f^{(i)}(X_i)\,|| & U_{l-i-1}\,|| & U_1|| & b(X_i)\,|| & \ldots\ldots|| & b(f^{(i-1)}(X_i)) \\ Y_{i+1} = & f^{(i+1)}(X_{i+1})\,|| & U'_{l-i-1}\,|| & b(X_{i+1})|| & b(f(X_{i+1}))|| & \ldots\ldots|| & b(f^{(i)}(X_{i+1})). \end{array}$$

$u \in \{0,1\}^{n+1}$ を、一様ランダムな $n+1$-ビット列（$u = U_{n+1}$）、または、$f(U_n)||b(U_n)$ とする。$\alpha := u[1, \ldots, n] \in \{0,1\}^n$, $\beta := u[n+1] \in \{0,1\}$ とすると、

$$Z := f^{(i)}(\alpha) \,||\, U_{l-i-1} \,||\, \beta \,||\, b(f(\alpha)) \,||\, \ldots\ldots \,||\, b(f^{(i-1)}(\alpha)).$$

と置いてやる。$u = U_{n+1}$ なら、$Z = Y_i$。一方、 $u = f(U_n)||b(U_n)$ なら $Z = Y_{i+1}$ であることに注意せよ。

問題 9 上記補題を証明せよ。

4.2 一方向性関数 ↔ 擬似乱数生成器

■一方向性関数 → 擬似乱数生成器 関数 f が長さ保存で無い場合，構成法は大変非効率な上，証明はとても面倒であるので今回は省略する。一方，関数 f が長さ保存の一方向性置換の時は，構成法はほぼ自明である：f に対するハードコアを b とする。その時，$f(x)$ の最後にハードコアビット $b(x)$ を追加してやれば，この $f(x)b(x)$ は擬似乱数である。

定理 10 関数 b を長さ保存の一方向性置換 f に対する (t,ϵ)-ハードコアとする。その時，$G(x) := f(x)||b(x)$ は 1 ビット伸長の (t,ϵ)-擬似乱数生成器である。

問題 11 上記定理を証明せよ。

■擬似乱数生成器 → 一方向性関数

定理 12　$G : \{0,1\}^n \to \{0,1\}^{n+l}$ が (t,ϵ)-擬似乱数生成器とする。すると、G は、$(t, \frac{\epsilon}{1-2^{-l}})$-一方向性関数である。

(証明) (ヒント) G の一方向性を破る敵を A とする。A を使って G に対する識別アルゴリズム D を作ってみる。するとその識別確率は A の成功確率 ϵ' で表現できる。D の走行時間が t であれば、識別確率は高々 ϵ であるので、ϵ' を ϵ の関数で抑えられる。

参考文献

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.