YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Yitong Yin *March 1, 2009*

Solutions to Problem Set 2

Originally due Thursday, February 26, 2009.

Problem 1 One-way functions and collections of one-way functions

[Textbook, Chapter 2, Exercise 18.]

Solution

(a) The sampling algorithm (I, D) on input 1^n can be seen as a function mapping $q(n)$ bits (the internal coin tosses) to a pair (i, x), where $q(n)$ is polynomial of n (number of internal coin tosses), $i \in \bar{I} \cap \{0,1\}^n$ and $x \in D_i$. Denoting this function as S_n, it is easy to see that

$$S_n(U_{q(n)}) = (I_n, X_n)$$

where I_n is the output distribution of algorithm I on input 1^n, and X_n is the output of algorithm D on input I_n.

A new function g can be defined as such: for each $z \in \{0,1\}^{q(n)}$, represent that $S_n(z) = (i, x)$, thus $g(z) = (i, f_i(x))$. It also follows that $g(U_{q(n)}) = (I_n, f_{I_n}(X_n))$.

We will show that the above g is a one-way function.

Function g is polynomial-time computable since S_n can be computed by the polynomial-time algorithms I and D and f_i can be computed by the polynomial-time algorithm F.

Suppose that g is polynomial-time invertable, namely, there exists a polynomial-time algorithms A such that for some polynomial $p(n)$,

$$\Pr[A(1^{q(n)}, g(U_{q(n)})) \in g^{-1}(g(U_{q(n)}))] \geq \frac{1}{p(n)}.$$

Define a new algorithm B as such: for each input $y = (i, f_i(x))$ where $i \in \bar{I} \cap \{0,1\}^n$ and $x \in D_i$, represent that $S_n(A(1^{q(n)}, y)) = (j, z)$, where $j \in \{0,1\}^n$ and such that $B(y) = z$. We show that B inverts the collection (I, D, F).

Since $g(U_{q(n)}) = (I_n, f_{I_n}(X_n))$, the event that $A(1^{q(n)}, g(U_{q(n)})) \in g^{-1}(g(U_{q(n)}))$ is equivalent to the event that $A(1^{q(n)}, (I_n, f_{I_n}(X_n))) \in g^{-1}(I_n, f_{I_n}(X_n))$, which according to the definition of B, holds only if $B(I_n, f_{I_n}(X_n)) \in f^{-1}(f_{I_n}(X_n))$. Therefore,

$$\begin{aligned} & \Pr[B(I_n, f_{I_n}(X_n))] \\ \geq\ & \Pr[A(1^{q(n)}, g(U_{q(n)})) \in g^{-1}(g(U_{q(n)}))] \\ \geq\ & \frac{1}{p(n)}, \end{aligned}$$

which is contra to the assumption that (I, D, F) is a one-way collection.

(b) Given a one-way function f, a one-way collection (I, D, F) can be defined as such: $I(1^n)$ always outputs 0^n, $D(0^n)$ samples uniformly over the domain of f, and $F(0^n, x) = f(x)$.

Problem 2 Hard core of one-way function

[Textbook, Chapter 2, Exercise 24, as modified below.]

Note that the "Guideline" is poorly printed and easy to misinterpret. The second subscript "$x_{\bar{I}}$" on the right side of the definition of g is $\bar{I}$, not I, but it takes sharp eyes to spot the difference in the printed version of the book. Here, $\bar{I}$ is intended to denote the set difference $\{1, 2, \ldots, |x|\} - I$.

Also, we are used to dealing with functions on strings, yet the guideline defines g to take a set argument. There are many ways of representing finite sets by strings. For this problem, represent the set $I \subseteq \{1, 2, \ldots, |x|\}$ by the length-$|x|$ bit-vector u, where $u_i = 1$ iff $i \in I$. It follows that $\bar{I}$ is represented by $\neg u$, the bitwise complement of u.

Finally, we define $x[u] = x_{i_1} \ldots x_{i_k}$, where i_j is the position of the j^{th} 1-bit in u, and k is the number of 1-bits in u. Thus, if u represents the set S, then $x[u]$ denotes the string x_S defined in the guideline.

Using these conventions, the intended function g is defined by

$$g(x, u) = (f(x[u]), x[\neg u], u).$$

You may ignore the part of the guideline that talks about more "dramatic" predictability.

Solution

We take the function $g(x, u) = (f(x[u]), x[\neg u], u)$ defined in the problem guideline. For each input (x, u), let $b_i(x, u)$ denote the ith bit of the input.

Let A_i be such an algorithm: for each input in the form of $(f(x[u]), x[\neg u], u)$ if $i > |u|$ (size of the set u), return the $(i - |u|)$th bit of the binary representation of u; if $i \leq |u|$ and $i \notin u$, return x_i which is contained in $x[\neg u]$; and if otherwise, flip a fair coin and return the outcome.

It is easy to see that it always holds that $A_i(g(x, u)) = b_i(x, u)$ for $i > |u|$, i.e. $\Pr[A_i(g(U_n)) = b_i(x, u)] = 1$ for $i > |u|$.

For $i \leq |u|$, $A_i(g(x, u)) = b_i(x, u)$ when $i \notin u$, or if $i \in u$ and A_i makes a correct guess. According to the total probability, for $i \leq |u|$,

$$\begin{aligned} & \Pr[A_i(g(U_n)) = b_i(x, u)] \\ = \; & \Pr[\text{the outcome of coin flipping is 1}] \Pr[u_i = 1] + 1 \cdot \Pr[u_i = 0] \\ = \; & \frac{1}{2} \Pr[u_i = 1] + \Pr[u_i = 0] \end{aligned}$$

where u is a uniform random string of length $|x|$, thus $\Pr[u_i = 1] = \Pr[u_i = 0] = \frac{1}{2}$. The above probability is $3/4$.

Problem 3 Amplification

Amplification is the technique for reducing errors in probabilistic algorithms by repeating the computation many times. We have used amplification both for reducing the error probability in algorithms attempting to invert one-way functions and also for increasing the advantage of algorithms attempting to guess hard core predicates. However, the way it is used differs markedly in the two cases.

For inverting functions, we assume an algorithm $A(y)$ succeeds with probability at least $\epsilon(n)$ at returning a value $x \in f^{-1}(y)$. To amplify, we repeat $A(y)$ for $r(n)$ times and succeed if any of the runs succeed. This depends on our ability to feasibly test whether the value returned by $A(y)$ in a given run is correct or not. Hence, the amplified success probability is at least $1 - (1 - \epsilon(n))^{r(n)}$.

For guessing the value of a hard core predicate, we assume an algorithm $D(y)$ with advantage $\epsilon(n)$ at predicting $b(x)$. To amplify, we repeat $D(y)$ for $r(n)$ times. Because we do knot know which runs of $D(y)$ return correct answers, we select our final answer to be the majority of all returned answers. The advantage of the resulting algorithm D' is not easy to compute directly, so we use the Chernoff bound or other statistical techniques to get a lower bound on it.

Questions: Suppose $\epsilon(n) = \frac{1}{n^{\log_2 n}} = \frac{1}{2^{(\log_2 n)^2}}$.

(a) Show that for all positive polynomials $p(\cdot)$ and all sufficiently large n that $\epsilon(n) < \frac{1}{p(n)}$.

(b) Suppose $\epsilon(n)$ is the success probability for algorithm A. Find as small a function $r(n)$ as you can such that repeating A for $r(n)$ times results in a success probability greater than $1/2$ for all sufficiently large n.

(c) Suppose $\epsilon(n)$ is the advantage of algorithm D. Find as small a function $r(n)$ as you can such that repeating D for $r(n)$ times and taking the majority gives an advantage greater than $1/4$ for all sufficiently large n.

Solution

(a) Let $LHS(n) = -\log_2 \epsilon(n) = (\log_2 n)^2$, and $RHS(n) = -\log_2 p(n)$. Since $p(\cdot)$ is a polynomial, $RHS(n) \geq C \log_2 n$ for some constant C which is independent of n.

For all $n > 2^C$, it holds that $\log_2 n > C$, thus $RHS(n) = (\log_2 n)^2 < C \log_2 n \leq RHS(n)$. Since $-\log_2(\cdot)$ is monotonically decreasing, it must holds all the above n that $\epsilon(n) = 2^{-LHS(n)} < 2^{-RHS(n)} = \frac{1}{p(n)}$.

(b) Since the runnings of algorithm are independent, the probability that all $r = r(n)$ trails fail is $(1-\epsilon)^r$. We want this probability to be less than $1/2$.

$$\begin{aligned}(1-\epsilon)^r &= \left(1 - \frac{1}{n^{\log_2 n}}\right)^r \\ &= \left(1 - \frac{1}{n^{\log_2 n}}\right)^{n^{\log_2 n} \cdot \frac{r}{n^{\log_2 n}}} \\ &\geq \exp\left(-\frac{r}{n^{\log_2 n}}\right).\end{aligned}$$

To have $(1-\epsilon)^r < 1/2$, it requires that $\exp\left(-\frac{r}{n^{\log_2 n}}\right) < 1/2$. Solving this inequality we have that $r(n) > n^{\log_2 n} \cdot \ln 2$.

(c) Let X_i be the random variable which indicates ith trial of D succeeds, it holds that $X_i = 1$ with probability $\frac{1}{2} + \epsilon$ and $X_i = 0$ with probability $\frac{1}{2} - \epsilon$. The probability that the majority fails is

$$\begin{aligned}&\Pr\left[\frac{1}{r}\sum_{i=1}^{r} \leq \frac{1}{2}\right] \\ = \;&\Pr\left[\frac{1}{r}\sum_{i=1}^{r} - \left(\frac{1}{2} + \epsilon\right) \leq -\epsilon\right]\end{aligned}$$

$$
\begin{aligned}
&\leq \Pr\left[\left|\frac{1}{r}\sum_{i=1}^{r} - \left(\frac{1}{2}+\epsilon\right)\right| \geq \epsilon\right] \\
&\leq 2\exp\left(\frac{-\epsilon^2 r}{\left(\frac{1}{2}-2\epsilon^2\right)}\right) \qquad \text{(due to Chernoff bound)} \\
&= q.
\end{aligned}
$$

The advantage is $1/4$, thus the probability that bad thing happens is less than $1/2-1/4 = 1/4$ i.e. $q < 1/4$. To make $q < \frac{1}{4}$, it requires that $r(n) > (\frac{1}{2\epsilon(n)^2} - 2)\ln 8 = (\frac{1}{2}n^{2\log_2 n} - 2)\ln 8$.