Foundations of Cryptography

Oded Goldreich and Shafi Goldwasser

Exercise 1: One-way functions and hard-core predicates

Due date: 13 Dec. 2001

1. Give a simple example of a function f such that
   - Given y, it is easy to find a preimage of y under f (i.e., x such that $f(x) = y$) if one exists.
   - Given $f(x)$ it is hard to find x: for any algorithm A, when x is chosen uniformly at random from $\{0,1\}^n$, the probability that A outputs x on input $f(x)$ is negligible.

   Is your function one-way?

2. (a) Prove that one-way functions cannot have polynomially bounded cycles. Namely, for every function f define $\mathrm{cyc}_f(x)$ to be the smallest *positive* integer i such that $f^i(x) = x$, where $f^{j+1}(x) = f(f^j(x))$ and $f^0(x) = x$. Prove that if f is (even weakly) one-way then for every polynomial $p(\cdot)$ and all sufficiently large n's the expected value of $\mathrm{cyc}_f(U_n)$ is greater than $p(n)$, where U_n is a random variable uniformly distributed over $\{0,1\}^n$.

   Hint: Recall Markov's inequality: for any non-negative random variable X and real number r:
   $$\Pr[X \geq r \cdot \mathrm{E}(X)] \leq \frac{1}{r}$$

   (b) Let f be an efficiently-computable function such that $\mathrm{E}(\mathrm{cyc}_f(U_n))$ is not bounded by any polynomial. Is f necessarily a weak one way-function? What about f for which $\min_{x\in\{0,1\}^n}(\mathrm{cyc}_f(x))$ is not bounded by any polynomial?

3. Assuming the existence of one-way functions, prove that there exist a one-way function f so that *no* single bit of the preimage constitutes a hard-core predicate.

   Hint: Given a one-way function f, consider the function $g(x, I) \stackrel{\text{def}}{=} (f(x_I), x_{\overline{I}}, I)$, where $I \subseteq \{1, 2, ...|x|\}$ and x_S denotes the string resulting by taking only the bits of x with positions in the set S (i.e., $x_{\{i_1,...,i_s\}} \stackrel{\text{def}}{=} x_{i_1} \cdots x_{i_s}$, where $x = x_1 \cdots x_{|x|}$).

continued...

4. Demetrius was overjoyed to see that given a weak one-way $f(x)$ we can construct a strong one-way function, namely $g(x_1, \ldots, x_{t(n)}) = (f(x_1), \ldots, f(x_{t(n)}))$ for an appropriate polynomial $t(\cdot)$. However, he is bothered by the use of a randomized algorithm in the reduction argument, and would like to alter the proof to use a deterministic algorithm instead.

   (a) Demetrius suggests the following variant:

   Assume there exists an algorithm B' and a polynomial $q(\cdot)$ such that B' inverts g with probability better than $\frac{1}{q(n)}$ for infinitely many n. Then the following algorithm A_{a} is supposed to invert f. On input y, A_{a} does the following:

   - Repeat $a(n)$ times (for an appropriate polynomial $a(\cdot)$):
     - for $i = 1$ to $t(n)$ do
       - Compute $(z_1, \ldots, z_{t(n)}) \leftarrow B'(f(0), \ldots, y, \ldots, f(0))$ where the y appears in the ith position. $(*)$
       - If $f(z_i) = y$ then halt and output z_i.

   Note that the only difference from the construction seen in class is in step $(*)$: when calling B', inputs other than y are set to $f(0)$ instead of random elements from the image of f.

   Can the proof seen in class be adapted to use A_{a}? If so, describe the necessary changes. If not, show that the modified reduction argument is wrong in the following sense: given a black box B' that inverts g with the required probability, A_{a} does *not* necessarily invert f with the claimed probability (in fact it may always fail).

   (b) Upon further reflection, Demetrius proposes another variant, A_{b}, which operates similarly to A_{a}, except that step $(*)$ is now

   - Compute $(z_1, \ldots, z_{t(n)}) \leftarrow B'(y, \ldots, y)$

   Evaluate A_{b} as before.

   (c) Consider the following variant, A_{c}. Fix some function $h : \{0,1\}^* \to \{0,1\}^*$. In A_{c}, step $(*)$ is as follows.

   - Compute $(z_1, \ldots, z_{t(n)}) \leftarrow B'(w_1, \ldots, w_{t(n)})$ where each w_j is a random image of f (i.e., $w_j = f(x_j)$ for $x_j \in_R \{0,1\}^n$), except for $w_i = y$ and $w_{t(n)} = h(w_1, \ldots, w_{t(n)-1})$.

   A_{c} is identical to the original construction seen in class, except that the last part of the input to B' is determined by the rest. Demetrius argues that if h is very "mixed up" then the input to B' looks "very random" so everything should work (and we saved saved n coin flips!). As before, evaluate this claim.

Please submit your answers typeset on a computer. New users of TEX may find the programs LYX (run `lyx` on the Unix host `verdi`) or WinEdt (`http://www.winedt.com`) useful.

Questions may be directed to Oded Goldreich `<oded@wisdom.weizmann.ac.il>` or Eran Tromer `<tromer@wisdom.weizmann.ac.il>`.