

# Exercise 1 – Foundations of Cryptography 89-856

## Solutions

June 22, 2008

**Exercise 1:** Show that the addition function  $f(x, y) = x + y$  (where  $|x| = |y|$  and  $x$  and  $y$  are interpreted as natural numbers) is not one-way.

**Solution 1:** Given  $z = f(x, y)$ , return  $(z, 0)$ . Clearly,  $f(z, 0) = z = f(x, y)$  and so  $(z, 0)$  is a preimage. The purpose of this exercise is to demonstrate the fact that it suffices to return *any* preimage of  $z$  under  $f$  (and it is not necessary to return the preimage used in computing  $z$ ).

**Exercise 2:** Prove that if there exist one-way functions, then there exists a one-way function  $f$  such that for every  $n$ ,  $f(0^n) = 0^n$ . Provide a full (formal) proof of your answer.

**Solution 2:** I provide a painfully detailed proof.

Let  $f$  be one-way function (that exists by the assumption) and define  $g(x) = f(x)$  for every  $x \neq 0^{|x|}$  and  $g(0^n) = 0^n$  for every  $n$ . Clearly,  $g$  fulfills the requirements. It remains to prove that it is one-way. First,  $g$  is efficiently computable. Second, assume by contradiction that there exists a PPT algorithm  $A$  and a polynomial  $p(\cdot)$  such that for infinitely many  $n$ 's, algorithm  $A$  inverts  $g$  with probability at least  $1/p(n)$ . We begin by analyzing the probability that  $A$  succeeds in inverting  $g$  on non-zero inputs:

$$\begin{aligned} \Pr \left[ A(g(U_n)) \in g^{-1}(g(U_n)) \right] &= \Pr \left[ A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n \right] \cdot \Pr [U_n \neq 0^n] \\ &\quad + \Pr \left[ A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n = 0^n \right] \cdot \Pr [U_n = 0^n] \\ &\leq \Pr \left[ A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n \right] + \Pr [U_n = 0^n] \\ &= \Pr \left[ A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n \right] + \frac{1}{2^n} \end{aligned}$$

Therefore, for infinitely many  $n$ 's we have that:

$$\Pr \left[ A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n \right] \geq \frac{1}{p(n)} - \frac{1}{2^n}$$

We now construct  $B$  that inverts  $f$  as follows. Upon receiving an input  $y$ ,  $B$  invokes  $A$  and returns whatever  $A$  outputs. We analyze  $B$ 's success:

$$\begin{aligned} \Pr \left[ B(f(U_n)) \in f^{-1}(f(U_n)) \right] &= \Pr \left[ B(f(U_n)) \in f^{-1}(f(U_n)) \mid U_n \neq 0^n \right] \cdot \Pr [U_n \neq 0^n] \\ &\quad + \Pr \left[ B(f(U_n)) \in f^{-1}(f(U_n)) \mid U_n = 0^n \right] \cdot \Pr [U_n = 0^n] \end{aligned}$$

$$\begin{aligned}
&\geq \Pr [B(f(U_n)) \in f^{-1}(f(U_n)) \mid U_n \neq 0^n] \cdot \Pr [U_n \neq 0^n] \\
&= \Pr [A(g(U_n)) \in g^{-1}(g(U_n)) \mid U_n \neq 0^n] \cdot \left(1 - \frac{1}{2^n}\right) \\
&\geq \left(\frac{1}{p(n)} - \frac{1}{2^n}\right) \cdot \left(1 - \frac{1}{2^n}\right) > \frac{1}{2p(n)}
\end{aligned}$$

We conclude that for infinitely many  $n$ 's, algorithm  $B$  inverts  $f$  with probability greater than  $1/2p(n)$ , in contradiction to the one-wayness of  $f$ .

**Exercise 3:** A function  $f$  is said to be **length regular** if for every  $x, y \in \{0, 1\}^*$  such that  $|x| = |y|$ , it holds that  $|f(x)| = |f(y)|$ . Show that if there exist one-way functions, then there exist length-regular one-way functions. Provide a full (formal) proof of your answer.

**Solution 3:** Let  $f$  be a one-way function. Since  $f$  is one-way, it is efficiently computable. Thus, there exists a polynomial  $p(\cdot)$  such that for every  $x$ ,  $|f(x)| \leq p(|x|)$ . Define  $f'(x) = f(x)10^{p(|x|)-|f(x)|}$ . Clearly,  $f'$  is length-regular. This is due to the fact that the output-length of  $f'(x)$  for every  $x$  of length  $n$  is exactly  $p(n) + 1$ . We now prove that  $f'$  is one-way. Our proof here is somewhat more concise than in the previous exercise.

Assume by contradiction that there exists a PPT adversary  $\mathcal{A}'$  and a polynomial  $q$  such that  $\mathcal{A}'$  inverts  $f'$  with probability at least  $1/q(n)$  for infinitely many  $n$ 's. We construct an adversary  $\mathcal{A}$  as follows:  $\mathcal{A}$  receives  $(y, 1^n)$  for input, invokes  $\mathcal{A}'$  on  $(y10^{p(|x|)-|f(x)|}, 1^n)$  and returns whatever  $\mathcal{A}'$  returns (we can assume that  $\mathcal{A}$  knows the polynomial  $p(\cdot)$ ). An important point to note here is that due to the fact that “padding” includes a single one followed by zeroes, the output of  $f'(x)$  uniquely defines the length of the portion that is  $f(x)$ . Therefore, the set of preimages of  $f'(x)$  equals the set of preimages of  $f(x)$ . This implies that if  $\mathcal{A}'$  inverts  $f'$ , then  $\mathcal{A}$  will have inverted  $f$ . By our assumption,  $\mathcal{A}'$  inverts with success probability of  $1/q(n)$ ; therefore the same is true of  $\mathcal{A}$ .

*Note: I expect a more detailed proof in your solutions.*

**Exercise 4:** Prove that if there exist collections of one-way functions, then there also exist one-way functions. Can you say the same for 1–1 one-way functions? Explain.

**Solution 3:** Let  $(I, D, F)$  be a collection of one-way functions. Let  $p_I(\cdot)$  denote the (polynomial) running-time of  $I$  and let  $p_D(\cdot)$  denote the (polynomial) running-time of  $D$ . Note that  $p_I(n)$  and  $p_D(n)$  constitute an upper bound on the number of random coins used by  $I$  and  $D$ , respectively, upon input of length  $n$ . Furthermore,  $p_D(n)$  also constitutes an upper bound on the length of values in the range of  $D$ .

We define a (single) one-way function  $g$  as follows. Upon input  $x$  of length  $n$ , first divide  $x$  into two parts  $x_1$  and  $x_2$  such that for some  $k$ ,  $|x_1| = p_I(k)$ ,  $|x_2| = p_D(k)$  and  $n/2 \leq |x_1| + |x_2| \leq n$ . (Note that it is always possible to find such a  $k$ . In order to do this, choose the larger polynomial, w.l.o.g. let it be  $p_I$ . Then, choose  $k$  such that  $p_I(k) = n/2$ . This  $k$  suffices.) The function  $g$  is defined as:

$$g(U_n) = g(x_1, x_2) = \left( I(1^k; x_1), F \left( I(1^k; x_1), D(I(1^k; x_1); x_2) \right) \right)$$

where  $I(1^k; x_1)$  denotes the output of algorithm  $I$  upon input  $1^k$  and *random-tape*  $x_1$ , and  $D(I(1^k; x_1); x_2)$  denotes the output of  $D$  upon input  $I(1^k; x_1)$  and *random-tape*  $x_2$ .

Note that  $I(1^k; x_1)$  is included in the output to force any inverting algorithm to find a preimage of  $F(I(1^k; x_1), D(I(1^k; x_1); x_2))$  under  $I(1^k; x_1)$  and not under some other function in the collection.

It remains to prove that  $g$  is a one-way function. First, it is clearly an efficiently computable function. Second, assume by contradiction that for infinitely many  $n$ 's, it can be inverted with probability  $1/q(n)$ . By the construction, this implies that for infinitely many  $k$ 's, the collection  $(I, D, F)$  can be inverted with probability  $1/q(n)$ . This holds because for every  $n$  there exists a  $k$  that results from the above derivation via  $p_I$  and  $p_D$ , and  $g$  can be inverted for these  $k$ 's with probability  $1/q(n)$ . Noting that for some constant,  $n = k^c$ , we have that for infinitely many  $k$ 's, the collection  $(I, D, F)$  can be inverted with probability  $1/q(k^c) = 1/\text{poly}(k)$ . This contradicts the one-wayness of  $(I, D, F)$ .

Notice that the above argument does *not* go through regarding 1–1 one-way functions. This is due to the fact that the mapping from random tapes to inputs may not be 1–1. Indeed, the question of obtaining 1–1 one-way functions (and *not* collections), or especially length-preserving 1–1 one-way functions (i.e., one-way permutations), is an interesting question. See <http://www.wisdom.weizmann.ac.il/oded/PS/gln.ps> for some prior work on this topic.

**Exercise 5:** Assume that  $\mathcal{P} \neq \mathcal{NP}$ . Show that there exists a function that is easy to compute and hard to invert in the worst case, but is not one-way.

Let  $G$  be a graph and let  $\phi$  be a 3-colouring of  $G$ . Then, consider the function  $f(G, \phi) = (G, 1)$  if  $\phi$  is a valid colouring of  $G$  and  $f(G, \phi) = (G, 0)$  if  $\phi$  is not a valid colouring of  $G$ . (Note, that  $G$  can be represented as a consecutive string of size approximately  $n^2/2$  where each bit in the string denote the existence or nonexistence of the appropriate edge.)

First,  $f$  is easy to compute, because the validity of a colouring can be checked in polynomial-time. Next, in the worst case, the function  $f$  is not invertible. This is due to the assumption that  $\mathcal{P} \neq \mathcal{NP}$  and so given  $(G, 1)$  it is hard – in the worst case – to find a colouring for  $G$ . Finally, note that  $f$  is *not* a one-way function. This is due to the fact that a random colouring of a random graph will be valid with very low probability. Thus, with high probability over a random input, the output of  $f$  will be some pair  $(G, 0)$  which is easily invertible (just compute an invalid colouring).

\* **Exercise 6:** Let  $x \in \{0, 1\}^n$  and denote  $x = x_1 \cdots x_n$ . Prove that if there exist one-way functions, then there exists a one-way function  $f$  such that for every  $i$  there exists an algorithm  $A_i$  such that,

$$\Pr_{x \leftarrow U_n} [A_i(f(x)) = x_i] \geq \frac{1}{2} + \frac{1}{2n}$$

We note that  $x \leftarrow U_n$  means that  $x$  is chosen according to the uniform distribution over  $\{0, 1\}^n$ .

This exercise demonstrates that it is not possible to claim that every one-way function hides at least one *specific* bit of the input.

**Solution 4:** Let  $f$  be a one-way function. Then, define  $g$  as follows:

$$g(U_{n+\log n}) = g(U_n^{(1)}, U_{\log n}^{(2)}) = f(U_n^{(1)}, U_{\log n}^{(2)}, [U_n^{(1)}]_{U_{\log n}^{(2)}})$$

where  $[x]_i$  denotes the  $i^{\text{th}}$  bit of  $x$ . In words,  $g$  receives an input of length  $n + \log n$  and outputs  $f$  applied to the first  $n$  bits, along with  $i$  and the  $i^{\text{th}}$  bit of the input, where  $i$  is the value of the binary number encoded in the last  $\log n$  bits of its input.

Fix  $i$ . We begin by constructing an algorithm  $A_i$  such that  $A_i(g(x))$  outputs  $x_i$  with probability  $1/2 + 1/2n$ . Algorithm  $A_i$  receives  $g(U_{n+\log n})$  and checks second  $\log n$  bits of the output. If they encode the value  $i$ , then  $A_i$  outputs the last bit of the output. If they encode some other value  $j \neq i$ , then  $A_i$  outputs a random bit  $b \in_R \{0, 1\}$ . Now,

$$\begin{aligned} \Pr [A_i(g(U_n^{(1)}, U_{\log n}^{(2)})) = [U_n^{(1)}]_i] &= \frac{1}{2} \cdot \Pr[U_{\log n}^{(2)} \neq i] + 1 \cdot \Pr[U_{\log n}^{(2)} = i] \\ &= \frac{1}{2} \left(1 - \frac{1}{n}\right) + \frac{1}{n} = \frac{1}{2} + \frac{1}{2n} \\ &> \frac{1}{2} + \frac{1}{2(n + \log n)} \end{aligned}$$

where this last annoying inequality is due to the fact that the input length is  $n + \log n$  and not  $n$  (and it holds because  $2n < 2(n + \log n)$ ). We conclude that on inputs of length  $k$ ,  $A_i$  succeeds in guessing the  $i^{\text{th}}$  bit with probability at least  $1/2 + 1/2k$ , as required.

It remains to prove that  $g$  is one-way. However, this is easy. Assume by contradiction that it can be inverted with probability  $1/\text{poly}(n)$  by some PPT algorithm  $A$ , then we can invert  $f$  with at least the same probability as follows. Upon receiving input  $y = f(x)$ , invoke  $A$  upon input  $(y, i, 0)$  and  $(y, i, 1)$  for all  $i = 1, \dots, n$ . Note that at least one of these tuples  $(y, i, b)$  is correct according to  $g$ . Therefore,  $A$  inverts this tuple, returning  $x$ , with the same probability that it inverts  $g$ . We can check all of the results and return the one that is correct, if there is such a one. (Note that we invoke  $A$  exactly  $2n$  times so the inversion procedure remains polynomial.) This inversion procedure succeeds with probability at least as high as the probability that  $A$  inverts  $g$ , in contradiction.