

Problem 1

Prove that if  $(E, D)$  is an encryption scheme with message size  $m$  and key size  $n < m$ , then there exist two messages  $x, x' \in \{0, 1\}^m$  such that  $\mathbf{E}_{U_n}(x)$  is not the same distribution as  $\mathbf{E}_{U_n}(x')$ . (In other words, perfect secrecy is only possible if the key length is at least the message length.)

*Proof.* Suppose, to the contrary, that  $\mathbf{E}_{U_n}(x) = \mathbf{E}_{U_n}(x')$  for all plaintexts  $x, x'$ . Let  $y$  be some ciphertext in the range of  $E$ . For every plaintext  $x$ , there is some key  $k \in \{0, 1\}^n$  such that  $E_k(x) = y$ . Now,  $E_k(x) = E_k(x')$  implies that  $x = x'$ . Hence, the available  $2^n$  keys are insufficient to map all  $2^m$  distinct plaintexts to the ciphertext  $y$ . Therefore, it must be that there exist two messages  $x, x' \in \{0, 1\}^m$  such that  $\mathbf{E}_{U_n}(x)$  is not the same distribution as  $\mathbf{E}_{U_n}(x')$ .  $\square$

Problem 2

Show that, if  $P = NP$ , then one-way functions do not exist.

*Proof.* Let  $f$  be a one-way function. Given any output  $y$  of  $f$ , define the language  $L$  to be the set of all  $x$  such that  $f(x) = y$ . We claim that  $L$  belongs to NP. A certificate for membership in  $L$  is simply the input  $x$ . To verify that  $x$  is indeed a member of  $L$ , one need only confirm that  $f(x) = y$ . Since  $f$  is polynomial-time computable, the verification can be done in polynomial-time. Hence,  $L$  belongs to NP. Now, since  $P = NP$ , we can discover  $x$  such that  $f(x) = y$  in polynomial time when we are given  $f$  and  $y$ . Therefore,  $f$  is not one-way.  $\square$

Problem 3

Prove: If  $\{f_k\}_{k \in \{0,1\}^*}$  is a pseudorandom function family, then for every polynomial  $\ell(n)$ , the function  $G$  that maps  $k \in \{0, 1\}^n$  to  $f_k(1), \dots, f_k(\ell(n))$  is a secure pseudorandom generator.

*Proof.* Suppose not. Then there is an evil polynomial time algorithm  $A$  so that for all  $\epsilon$  there exists  $n$  such that

$$|\Pr[A(G(U_n)) = 1] - \Pr[A(U_{nl(n)}) = 1]| \geq \epsilon(n)$$

We create the oracle equipped polynomial time algorithm  $B^{f_k}$  which computes  $f_k(1), \dots, f_k(\ell(n))$  then shoves it into  $A$ .  $A(G(k)) = B^{f_k}(1^n)$ , so

$$\Pr_{k \in_R \{0,1\}^n} [A(G(k)) = 1] = \Pr_{k \in_R \{0,1\}^n} [B^{f_k}(1^n) = 1]$$

Also if  $g$  is randomly selected then the distribution of  $g(1), \dots, g(\ell(n))$  is uniform. Thus,

$$\Pr_{g \in_R F_n} [B^g(1^n) = 1] = \Pr_{k \in_R \{0,1\}^{nl(n)}} [A(k) = 1]$$

Finally,

$$\left| \Pr_{k \in_R \{0,1\}^n} [B^{f_k}(1^n) = 1] - \Pr_{g \in_R F_n} [B^g(1^n) = 1] \right| = |\Pr[A(G(U_n)) = 1] - \Pr[A(U_{nl(n)}) = 1]| \geq \epsilon(n)$$

So  $\{f_k\}$  is not a pseudorandom function family, causing a contradiction.  $\square$

Problem 4

Show that if  $f$  is a one-way permutation then so is

$$f^{(k)} = \underbrace{f \circ f \circ \dots \circ f}_k$$

where  $k = n^c$  for some fixed  $c > 0$ . Furthermore, show that the assumption that  $f$  is a permutation is necessary.

*Proof.* Assume  $f^{(k)}$  is not a one-way function. Then let  $A$  be a polynomial time algorithm so that for all  $\epsilon$ , there exists an  $n$  such that

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = f^{(k)}(x)}} [f^{(k)}(A(y)) = y] \geq \epsilon(n)$$

Since  $f$  and  $f^{(k)}$  are both permutations,

$$\begin{aligned} \Pr_{\substack{x \in_R \{0,1\}^n \\ y = f^{(k)}(x)}} [f^{(k)}(A(y)) = y] &= \Pr_{y \in_R \{0,1\}^n} [f^{(k)}(A(y)) = y] \\ &= \Pr_{\substack{x \in_R \{0,1\}^n \\ y = f(x)}} [f^{(k)}(A(y)) = y] \\ &= \Pr_{\substack{x \in_R \{0,1\}^n \\ y = f(x)}} [f(f^{(k-1)}(A(y))) = y] \end{aligned}$$

$f^{(k-1)}(A(y))$  can be computed in polynomial time. First compute  $A(y)$ , which is polynomial time, then apply  $f$   $k-1$  times, which takes  $n^c$  times the running time of  $f$ , which is polynomial time. This contradicts that  $f$  is a one-way function.

Furthermore, if a one-way function exists, then a one-way function exists whose outputs and inputs are the same length, specifically  $f_U$ . Let  $g$  be the function that computes  $f_U$ , then outputs a string of length  $2n$ , consisting of  $f_U(x)$  listed twice. Then the length of the output of  $f^{(k)}$  is  $2^{n^c} n$ , which is not printable in polynomial time, much less computable in polynomial time.  $\square$

#### Problem 5

Suppose  $x \in \{0,1\}^m$  is an unknown vector. Let  $r_1, \dots, r_m \in \{0,1\}^m$  be randomly chosen, and  $x \odot r_i$  revealed to us for all  $i \in [m]$ . Describe a deterministic algorithm to reconstruct  $x$  from this information, and show that the probability (over the choice of the  $r_i$ 's) is at least  $1/4$  that it works. (This shows that if  $r_1, \dots, r_m$  are fully independent then we cannot guess  $x \odot r_1, \dots, x \odot r_m$  with probability much better than  $2^{-m}$ , and hence it was crucial to move to a merely pairwise independent collection of vectors in the proof of the Goldreich-Levin Theorem).

*Proof.* If  $r_1, \dots, r_m$  are linearly independent vectors then for each  $i \in [m]$ :

- Determine the linear combination  $\sum_{j=1}^m c_j r_j$  which gives the vector  $e_i$  with a 1 in the  $i$ th position and 0 elsewhere.
- Compute the  $i$ th coordinate of  $x$  as follows:

$$x_i = x \odot e_i = x \odot \sum_{j=1}^m c_j r_j = \sum_{j=1}^m c_j (x \odot r_j)$$

Now the probability that  $r_1, \dots, r_m$  are linearly independent is:

$$\prod_{i=1}^m \left( \frac{2^i - 1}{2^i} \right) \geq \frac{1}{2} \prod_{i=2}^m \frac{2^{i-1} + 1}{2^{i-1} + 2} = \frac{2^{m-1} + 1}{8} \prod_{i=2}^{m-1} \frac{2^{i-1} + 1}{2^i + 2} = \frac{2^{m-1} + 1}{2^{m+1}} > \frac{1}{4}$$

$\square$

#### Problem 6

Let  $V_n$  denote the binomial distribution on  $n$  points with probability  $1/3$ , i.e.,

$$\mathbf{P}_{V_n}(w) = \left( \frac{1}{3} \right)^{\text{wt}(w)} \left( \frac{2}{3} \right)^{n - \text{wt}(w)}$$

for each  $w \in \{0,1\}^n$ , where  $\text{wt}(w)$  denotes the number of 1's in  $w$ . Call a polynomial-time computable function  $G : \{0,1\}^* \rightarrow \{0,1\}^*$  of stretch  $\ell(n)$  a "secure biased pseudorandom generator" if, for every

probabilistic polynomial time algorithm  $A$ , there exists a negligible function  $\epsilon : \mathbb{N} \rightarrow [0, 1]$  such that, for all  $n$ ,

$$|\mathbf{P}[A(G(U_n)) = 1] - \mathbf{P}[A(V_{l(n)}) = 1]| < \epsilon(n).$$

Show that, if a secure biased pseudorandom generator exists, then a secure pseudorandom generator of stretch  $\alpha l(n)$  exists for some fixed  $\alpha > 0$ . (Hint: You may want to use Chernoff-type bounds: see Corollary A.15 in Arora-Barak.) In particular, provide a *deterministic* polynomial-time algorithm which transforms the output of a secure biased pseudorandom generator into that of a secure pseudorandom generator.

*Proof.* Let  $\alpha = 7/18$  (We'll use it later). To transform the biased pseudorandom number  $x$  into a pseudorandom number  $y$ , we proceed along  $x$  taking two digits at a time. If we find a 00, then the next digit of  $y$  is 0, if we find a 01 or 10, then the next digit of  $y$  is 1, otherwise do nothing. We continue until either we have  $\alpha l(n)$  bits, in which case we output the bits we've collected, or we get to the end of  $x$ , in which case we output all zeros. Call this pseudorandom generator  $g$ . We need to show that this a secure pseudorandom generator.

Suppose not. Then there is an algorithm  $A$  that ruins its day. Let's take an arbitrary negligible  $\epsilon$ .  $\epsilon(n) + 2e^{n/72}$  is also a negligible function so there exists an  $n$  such that

$$\Pr[A(g(U_n)) = 1] - \Pr[A(U_{\alpha l(n)}) = 1] \geq \epsilon(n) + 2e^{n/72}$$

Build algorithm  $B$  to be the algorithm that takes a biased pseudorandom number, uses our method to compute a pseudorandom number  $y$ , then return  $A(y)$ . Then  $B(G(x)) = A(g(x))$ , so  $\Pr[A(g(U_n)) = 1] = \Pr[B(G(U_n)) = 1]$ .

Now  $\Pr[B(V_{l(n)}) = 1]$  depends on  $A(0^{\alpha l(n)})$ . Let  $E$  be the event that the input for  $B$  has too many 11 pairs, and so our conversion process returns  $0^{\alpha l(n)}$ . Then if  $A(0^{\alpha l(n)}) = 0$

$$\Pr[B(V_{l(n)}) = 1] = (1 - \Pr(E))\Pr[A(U_{\alpha l(n)}) = 1]$$

otherwise  $A(0^{\alpha l(n)}) = 1$  and

$$\Pr[B(V_{l(n)}) = 1] = (1 - \Pr(E))\Pr[A(U_{\alpha l(n)}) = 1] + \Pr(E)$$

Either way

$$\Pr[A(U_{\alpha l(n)}) = 1] - \Pr(E) \leq \Pr[B(V_{l(n)}) = 1] \leq \Pr[A(U_{\alpha l(n)}) = 1] + \Pr(E)$$

Now the expectation for the number of 11 pairs is  $l(n)/18$ . And I rigged  $\alpha$  (this is where we use  $7/18$ ) so that event  $E$  is having more than  $l(n)/9$  pairs. So we use Chernoff bounds to estimate  $\Pr(E)$  with  $c = 1$  and  $\mu = l(n)/18$ .

$$\Pr(E) \leq 2e^{n/72}$$

Hence

$$\Pr[A(U_{\alpha l(n)}) = 1] - 2e^{n/72} \leq \Pr[B(V_{l(n)}) = 1] \leq \Pr[A(U_{\alpha l(n)}) = 1] + 2e^{n/72}$$

Now to drive it home,

$$\begin{aligned} |\Pr[B(G(U_n)) = 1] - \Pr[B(V_{l(n)}) = 1]| &\geq |\Pr[A(g(U_n)) = 1] - \Pr[A(U_{\alpha l(n)}) = 1]| - 2e^{n/72} \\ &\geq \epsilon(n) \end{aligned}$$

Thus  $G$  is not a secure biased pseudorandom generator, which is a contradiction.  $\square$