Handout 3: Computational Indistinguishability

Boaz Barak

Exercises due Tuesday October 11th, 2005 1:30pm
Total of 120 points (not including additional extra credit question of 20 points).

Recall that X and Y are (T, ϵ)-computationally indistinguishable, denoted by $X \approx_{T,\epsilon} Y$ if for every T-sized Boolean circuit C, $|\Pr_X[C(X) = 1] - \Pr_Y[C(Y) = 1]| \leq \epsilon$

Exercise 1 (15 points)**.** Prove that computational indistinguishability is preserved under multiple independent samples. That is, suppose that X, Y, X', Y' are four independent distributions over $\{0,1\}^n$ and that $X \approx_{T,\epsilon} X'$ and $Y \approx_{T,\epsilon} Y'$ (for $T > 100n$). Let $\circ$ denote the concatenation operator (that is for two strings x, y $x \circ y$ is the concatenation of x and y). Then, $X \circ Y \approx_{T/5, 5\epsilon} X' \circ Y'$
Hint: Find a suitable intermediate distribution and then use the transitivity / triangle inequality property of computational indistinguishable.

Exercise 2 (15 points)**.** Prove the existence of a (not necessarily efficiently computable) pseudorandom generator. That is, prove that for every (sufficiently large) n, there exist a $(2^{n/10}, 2^{-n/10})$-pseudorandom generator $g : \{0,1\}^n \to \{0,1\}^{2^{n/20}}$. See footnote for hint[1]

Is the output of your function (i.e., $g(U_n)$) also *statistically close* to the uniform distribution on $2^{n/20}$ bits?

Exercise 3 (15 points)**.** The RSA SecurID card (see Figure 1) is a credit-card sized device that displays 6 digits that change every minute. The idea is that when you log into your account remotely (say when you want to log into your UNIX account in Princeton from an Internet Cafe) then you have to type the numbers that appear in the card in addition to your PIN or password.

1. What is the security advantage of such a card over traditional password? That is, what sort of attack can this card resist which cannot be resisted using a standard password mechanism. (Assume that it's possible for users to remember a 6-digits PIN or a password with similar security.)

2. Describe how you would implement such a scheme using pseudorandom functions.

   Assume that the PRF family takes a seed of size n, and that the number of possible devices is m (for $m < 2^n$). How many bits of storage does your implementation use at the server and each of the devices? (there is an implementation that uses at most $O(n)$ bits).

3. Try to *define* what it means that such a scheme is *secure* and sketch a proof that your construction satisfies it (you don't have to formally define and prove if you don't want to — you can use English but try to be precise). Say how the security depends on n - the number of bits that the device stores in memory (where its running time is polynomial in n) and on k - the number of digits that we display to the user.

Exercise 4 (15 points)**.** Recall that in class we gave a construction of a *probabilistic* CPA-secure encryption scheme (i.e., the function E used extra randomness in computing the encryption).

[1]**Hint:** Show that a random function $g(\cdot)$ will satisfy this property with high probability. Do this by using the *Chernoff bound* (see first handout) to bound the probability of $g(\cdot)$ failing with respect to a fixed $2^{n/10}$-sized circuit C (that is, consider the experiment where first C is fixed and *then* g is chosen at random). Since the probability that $g(\cdot)$ is not a PRG — i.e. that it fails for *some* circuit (that can depend on the choice of $g(\cdot)$) — is the union of these bad events with respect to all possible circuits, you can use the *union bound* to bound it.

Figure 1: RSA SecurID Device.

1. Show that there is no *deterministic* and *stateless* CPA-secure encryption scheme.
2. Give a construction for a *deterministic stateful* CPA-secure encryption scheme. (Note that CPA security implies in particular that the total number of messages is longer than the key length.) A stateful deterministic encryption scheme is often called a *stream cipher*.

Exercise 5 (15 points)**.** In these two questions you'll show that if we have a pseudorandom function family with particular input and output sizes, we can easily obtain a family that handles larger inputs and outputs. (It's easy to handle smaller outputs and inputs by truncation and padding.)

1. *(Changing PRFs output size)* Prove that if there exists a collection $\{f_s\}$ of pseudorandom functions with $f_s : \{0,1\}^{|s|} \to \{0,1\}$ (i.e., one-bit output) then there exists a collection $\{f'_s\}$ with $f'_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$. See footnote for hint.[2]
2. *(Changing PRFs input size)* Prove that if there exists a collection $\{f_s\}$ of pseudorandom functions with $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$ then there exists a collection $\{f'_s\}$ with $f'_s : \{0,1\}^* \to \{0,1\}^{|s|}$ (i.e., f'_s for a random $s \in \{0,1\}^n$ is indistinguishable from a random function from $\{0,1\}^*$ to $\{0,1\}^n$. See footnote for hint[3]

Additional Exercises.

Exercise 6 (15 points)**.** Recall that we defined a function $T : \mathbb{N} \to \mathbb{N}$ to be *super-polynomial* if $T(n) = n^{\omega(1)}$ or in other words, for every constant $c > 0$ there exists $N > 0$ such that for every $n > N$, $T(n) > cn^c$.[4]

[2]**Hint:** First come up with a pseudorandom family with output longer than 1 but shorter than $|s|$. For example, if $s \in \{0,1\}^{n^2}$ then the output can be n. Then show that existence of PRF implies existence of pseudorandom generators and use that to expand your output.

[3]**Hint:** (This is definitely not the only approach to do this.) First note that such a PRF family implies immediately a family where $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|/2}$. Then try to use this to get a function f'_s that works only for inputs whose size is a multiple of $|s|/2$. Then try to get a function that works for every finite length string.

[4]**Reminder of O notations:** (We don't need all of these but it's good to know.) Let $f, g : \mathbb{N} \to \mathbb{N}$ be two functions. We say that $f(n) = O(g(n))$ (or sometimes just $f = O(g)$) if there is a constant $c > 0$ and a number N such that for every $n > N$, $f(n) \le c \cdot g(n)$. We say that $f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$ or equivalently, there's a constant $c > 0$ such that for every large enough n, $f(n) \ge c \cdot g(n)$. We say that $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. We say that $f(n) = o(g(n))$ if $f(n) = O(g(n))$ but $f(n) \ne \Omega(g(n))$. In other words $f(n) = o(g(n))$ if for *every* constant $c > 0$ there's an N such that for every $n > N$, $f(n) < c \cdot g(n)$. We say that $f(n) = \omega(g(n))$ if $g(n) = o(f(n))$. That is, for every $c > 0$ if n is large enough then $f(n) > c \cdot g(n)$.

1. For each the following functions say (no need to prove) whether it is super-polynomial or not.

   (a) $f_1(n) = 2^{\sqrt{n}}$.

   (b) $f_2(n) = n^{\log n}$

   (c) $f_3(n) = n \log n$.

   (d)
   $$f_4(n) = \begin{cases} 2^n & n \text{ even} \\ n^2 & n \text{ odd} \end{cases}$$

2. Prove that for every super-polynomial function $T : \mathbb{N} \to \mathbb{N}$ the function $T' : \mathbb{N} \to \mathbb{N}$ defined as follows $T'(n) = \frac{T(n^{1/3})^{1/3}}{n^3}$ (with all values rounded to integers if they are not integers) is also super-polynomial.

Pseudorandom permutations. Recall that we define a collection of permutations $\{p_k\}_{k\in\{0,1\}^*}$ where for $k \in \{0,1\}^n$, p_k is a permutation over $\{0,1\}^{m(n)}$ to be a *pseudorandom-permutation collection* if it satisfies:[5]

- *(Efficient computation)* The functions $(k,x) \mapsto p_k(x)$ and $(k,y) \mapsto p_k^{-1}(y)$ are efficiently computable (i.e., in polynomial time).
- *(Pseudorandomness)* There are super polynomial functions T, ϵ such that for every $T(n)$ time adversary A,

$$\left| \Pr_{k \leftarrow_{\mathrm{R}} \{0,1\}^n} \left[\mathsf{A}^{p_k(\cdot), p_k^{-1}(\cdot)}(1^m) = 1 \right] - \Pr_{P \leftarrow_{\mathrm{R}} \{0,1\}^m \xrightarrow{1\text{-}1} \{0,1\}^m} \left[\mathsf{A}^{P(\cdot), P^{-1}(\cdot)}(1^m) = 1 \right] \right| < \epsilon(n)$$

  (The notation $\mathsf{A}^{f,g}$ means that the adversary is given oracle access to the functions $f(\cdot)$, $g(\cdot)$ which naturally it can query for at most T times. If an event happens with probability at most $1/n^{\omega(1)}$ then we say it happens with *negligible* probability.)

Exercise 7 (15 points)**.** Let $\{p_k\}_{k\in\{0,1\}^*}$ be a pseudorandom permutation collection, where for $k \in \{0,1\}^n$, p_k is a permutation over $\{0,1\}^m$.

1. Consider the following encryption scheme (E, D): $\mathsf{E}_k(x) = p_k(x)$, $\mathsf{D}_k(y) = p_k^{-1}(y)$. Prove that this scheme is *not* a CPA-secure encryption.

2. Consider the following scheme (E, D) that encrypts $m/2$-bit messages in the following way: on input $x \in \{0,1\}^{m/2}$, E_k chooses $r \leftarrow_{\mathrm{R}} \{0,1\}^{m/2}$ and outputs $p_k(x,r)$ (where comma denotes concatenation), on input $y \in \{0,1\}^{m/2}$, D_k computes $(x,r) = p_k^{-1}(y)$ and outputs x. Prove that (E, D) *is* a CPA-secure encryption scheme.

   Hint: You can prove first that this scheme satisfies the weaker notion of *multiple message security*. That is, for every polynomial $p = p(n)$ and $x_1, \ldots, x_p, x'_1, \ldots, x'_p \in \{0,1\}^{m/2}$ adversaries the two sequences of random variables $\langle Enc_K(x_1), \ldots, \mathsf{E}_K(x_p)\rangle$ and $\langle \mathsf{E}_K(x'_1), \ldots, \mathsf{E}_K(x'_p)\rangle$ are computationally indistinguishable (where K and K' are two independent random variables distributed uniformly over $\{0,1\}^n$).

[5]We assume that m and n are polynomially related. That is, $n^{1/c} < m(n) < n^c$ for some constant c.

Exercise 8 (15 points)**.** The CBC construction is often used to get an encryption for larger message size. If $p : \{0,1\}^m \to \{0,1\}^m$ is a permutation, then $\mathsf{CBC}_\ell\langle p\rangle$ is a permutation from $\{0,1\}^{\ell\cdot m}$ to $\{0,1\}^{\ell\cdot m}$ defined in the following way: for $x_1,\ldots,x_\ell \in \{0,1\}^m$, let $y_0 = 0^n$ and define $y_i = p(y_{i-1} \oplus x_i)$. Then, $\mathsf{CBC}_\ell\langle p\rangle(x_1,\ldots,x_\ell) = (y_1,\ldots,y_\ell)$.[6] Note that the inverse of $\mathsf{CBC}_\ell\langle p\rangle$ can be computed in a similar way using the inverse of $p(\cdot)$.

Let $\{p_k\}$ be a pseudorandom permutation collection. Determine the CPA-security of the following two encryption schemes which are based on the CBC construction. That is, for each scheme either prove that it is CPA-secure or give an attack showing that it is not. For simplicity, we consider only the 3-block variant of the scheme (i.e. $\ell = 3$).

1. *(Padding in the end)* Given $p_k : \{0,1\}^m \to \{0,1\}^m$ and a message $x = x_1, x_2 \in \{0,1\}^{2m}$, E_k chooses $r \leftarrow_{\mathrm{R}} \{0,1\}^m$ and outputs $\mathsf{CBC}_3\langle p_k\rangle(x_1,x_2,r)$. Decrypting done in the obvious way.

2. *(Padding in the start)* Given $p_k : \{0,1\}^m \to \{0,1\}^m$ and a message $x = x_1, x_2 \in \{0,1\}^{2m}$, E_k chooses $r \leftarrow_{\mathrm{R}} \{0,1\}^m$ and outputs $\mathsf{CBC}_3\langle p_k\rangle(r,x_1,x_2)$. Decrypting done in the obvious way.

Exercise 9 (Extra credit question — do only if you have energy and time — 20 points.)**.** Suppose that we designed a cryptographic chip implementing a pseudorandom permutation and wanted to insert a hidden "back door" into it. More formally, we want to construct a collection $\{P_{k_{\mathsf{master}},k}\}$ that has two keys: the "normal" key k and a *master* key k_{master}. We will choose k_{master} at random and know it, and give to our unsuspecting clients two black-boxes $F_{k_{\mathsf{master}}}(\cdot,\cdot)$ and $B_{k_{\mathsf{master}}}(\cdot,\cdot)$ (for forward and backward) that allow the client to specify k and compute $P_{k_{\mathsf{master}},k}$ forwards and backwards (i.e., $F_{k_{\mathsf{master}}}(k,x) = P_{k_{\mathsf{master}},k}(x)$ and $B_{k_{\mathsf{master}}}(k,y) = P^{-1}_{k_{\mathsf{master}},k}(y)$).[7] To anyone not knowing k_{master} this should behave like a normal pseudorandom permutation, but we should be able to break it.

That is, on the one hand if k_{master} and k are chosen at random then access to the algorithms $F_{k_{\mathsf{master}}}(k,\cdot)$ and $B_{k_{\mathsf{master}}}(k,\cdot)$ is indistinguishable from access to a random permutation and its inverse.

On the other hand, we'd like to be able to break the scheme and recover the client's key k using the master key k_{master}. This naturally leads to the following questions:

1. *(Insecurity for chosen inputs)* Show a construction for such a scheme where there exists a polynomial-time algorithm A that if k_{master} and k are chosen at random and A is given k_{master} as input and access to a black box computing $P_{k_{\mathsf{master}},k}$ then it can recover k with probability at least 0.9.

2. *(Insecurity for known inputs)*[8] Show a construction for such a scheme where there exists a polynomial-time algorithm A and a polynomial $q = q(n)$ such that if k_{master} and k are chosen at random, and $x_1,\ldots,x_q$ are also chosen uniformly and independently at random, and A is given k_{master} and the sequence of pairs $\langle x_1, P_{k_{\mathsf{master}},k}(x_1)\rangle$, …, $\langle x_q, P_{k_{\mathsf{master}},k}(x_q)\rangle$ as input, then A outputs k with probability at least 0.9.

[6] Often a different public value instead of 0^m is chosen for y_0 although this does not make a lot of difference for security. This value is called IV or initialization vector.

[7] Actually, we can allow a slight relaxation: $P_{k_{\mathsf{master}},k}$ does not necessarily have to be a permutation as long as without knowledge of k_{master} it is not possible to find an x on which the inversion algorithm fails. That is, for some x's it may be $B_{k_{\mathsf{master}}}(k, F_{k_{\mathsf{master}}}(k,x)) \neq x$ but these inputs should be hard to find by the client.

[8] Solving this took me a bit of time, a slightly cumbersome construction, and also required me to use the relaxation that $P_{k_{\mathsf{master}},k}$ may not be completely a permutation (but rather for a random k_{master}, polynomial algorithms can't find inputs k,x on which $B_{k_{\mathsf{master}}}(F_{k_{\mathsf{master}}}(k,x)) \neq x$ with non-negligible probability).