

## **Solution: Midterm Exam**

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on the top of every page of this quiz booklet.
- You have 80 minutes to earn a maximum of 80 points. Do not spend too much time on any one problem. Skim them all first, and attack them in the order that allows you to make the most progress.
- **You are allowed three double-sided letter-sized sheet with your own notes.** No calculators, laptops, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. Pages will be scanned and separated for grading. If you need more space, write “Continued on S1” (or S2, S3) and continue your solution on the referenced scratch page at the end of the exam.
- Do not waste time and paper rederiving facts that we have studied in lecture, recitation, or problem sets. Simply cite them.
- **Pay close attention to the instructions for each problem.** Depending on the problem, partial credit may be awarded for incomplete answers.

| Problem             | Parts | Points |
|---------------------|-------|--------|
| 0: Information      | 2     | 2      |
| 1: True or False?   | 7     | 28     |
| 2: Does It Permute? | 1     | 15     |
| 3: Fiesty Ciphers   | 3     | 20     |
| 4: Combiners!       | 1     | 15     |
| Total               |       | 80     |

Name: \_\_\_\_\_

School Email: \_\_\_\_\_

**Problem 1.** [28 points] **True or False?** (7 parts)

For each of the following questions, tick either **T** (True) or **F** (False) box. Each problem is worth 4 points. **You will get 4 points for a correct answer, 0 points for an incorrect answer, and 2 points if you leave it blank.** No justification is necessary, and no partial credit will be given.

- (a) ☐ **T** ☐ **F** The function  $\mu(n) := 1/2^{\sqrt{\log_2 n}}$  is negligible.

**Solution: False.** For  $n \geq 3$ ,  $\mu(n) = 1/2^{\sqrt{\log_2(n)}} \geq 1/2^{\log_2(n)} = 1/n$ .

- (b) ☐ **T** ☐ **F** For any one-way permutation  $f(x)$ ,  $f(f(x))$  is also a one-way permutation.

**Solution: True.** Clearly  $f(f(x))$  is a permutation. To see that it is one-way, suppose an adversary could invert  $y = f(f(x))$ . Then, to invert  $z = f(x)$ , the adversary can feed in  $f(z) = f(f(x))$  into the adversary, and non-negligibly often, the adversary finds the unique  $x$  such that  $y = f(f(x)) = f(z)$ . Since  $f$  is a permutation, this implies  $f(x) = z$ , so  $x$  is the pre-image for  $z$ .

- (c) ☐ **T** ☐ **F** Any length-doubling pseudo-random generator (PRG) family is also a one-way function (OWF) family.

**Solution: True.** Suppose not. Then, on input  $y = G(x)$ , an adversary finds  $x'$  such that  $G(x') = y$  non-negligibly often. To break the PRG, the distinguisher can output 1 if the adversary above succeeds, and 0 otherwise. It will output 1 with  $1/\text{poly}$  probability on  $G$ , but output 1 on only at most  $1/2^n$  fraction of truly random points. This is a non-negligible distinguishing advantage.

- (d) ☐ **T** ☐ **F** There is a known proof that public-key encryption is possible assuming only that one-way functions (OWF) exist.

**Solution: False.** This is a major open question.

- (e) **T** **F** Assume that one-way permutations exist. Then, there exists a one-way permutation  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  such that  $f(f(x)) = x$  for all  $x \in \{0, 1\}^n$ .

**Solution: False.** The function  $f$  itself is an efficient inverter for  $f$ .

- (f) **T** **F** There exists a function  $h : \{0, 1\}^n \rightarrow \{0, 1\}$  such that for all one-way functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ ,  $h$  is a hardcore predicate for  $f$ .

**Solution: False.** Consider the function  $f(x) = g(x) || h(x)$  where  $g$  is a one-way function.  $f$  is itself a one-way function, but clearly  $h$  is not hardcore for  $f$ .

- (g) **T** **F** The following two distributions are computationally indistinguishable:

$$(g, g^x, g^y, g^{xy}) \quad \text{and} \quad (g, g^x, g^y, g^z)$$

where  $g$  is uniformly drawn generator of the group  $\mathbb{Z}_p^*$ , and  $x, y$  and  $z$  are uniformly drawn from the set  $\{1, 2, \dots, p-1\}$ .

**Solution: False.** The probability that  $g^{xy}$  is a square mod  $p$  is  $3/4$ ; happens if at least one of  $x$  or  $y$  are even. But of course,  $g^z$  is a square with probability  $1/2$ . Since squareness mod  $p$  can be detected in polynomial time, these distributions are distinguishable.

**Problem 2.** [15 points] **Does It Permute?**

Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a one-way permutation. Construct a one-way permutation  $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$  such that  $g(0^n) = 0^n$ . (Here,  $0^n$  refers to the  $n$ -bit string of zeroes.) Prove that your construction is a permutation and that it is one-way.

**Solution:** Let  $g(x) = f(x) \oplus f(0^n)$ . Clearly,  $g$  is efficiently computable.

To see that  $g$  is a permutation, let  $h : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be defined by  $h(x) := f^{-1}(x \oplus f(0^n))$ . We have

$$h(g(x)) = h(f(x) \oplus f(0^n)) = f^{-1}(f(x) \oplus f(0^n) \oplus f(0^n)) = x,$$

showing that  $g$  is injective. Since the domain and codomain of  $g$  are the same size, this implies  $g$  is a permutation.

Furthermore, note that  $g(0^n) = f(0^n) \oplus f(0^n) = 0^n$ .

Lastly, to see that  $g$  is one-way, suppose not. Then, there is some p.p.t. adversary  $A$  such that  $A(g(x)) = x$  non-negligibly often over  $x \leftarrow \{0, 1\}^n$ . For  $z = f(x')$ , define the adversary  $B(z)$  by outputting  $A(z \oplus f(0^n))$ . Since  $z \oplus f(0^n) = f(x') \oplus f(0^n) = g(x')$ , we know that  $A$  will output  $x'$  non-negligibly often. This is the inverse of  $z$ , as desired.

**Problem 3.** [20 points] **Fiesty Ciphers** (3 parts)

Let  $\mathcal{F} = \{F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n : k \in \{0, 1\}^n\}$  be a pseudorandom function (PRF) family. We say that  $\mathcal{G} = \{G_k : \{0, 1\}^n \rightarrow \{0, 1\}^n : k \in \{0, 1\}^m\}$ , for  $m = \text{poly}(n)$ , is a pseudorandom *permutation* (PRP) family if it is a PRF family, and additionally, (1) each  $G_k$  is a permutation; and (2) each  $G_k$  is invertible in polynomial time given  $k$ . The security definition for a PRP is exactly the same as that for a PRF.

(a) [4 points] Define  $G_k : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$  to be

$$G_k(x||y) := y||(F_k(y) \oplus x) .$$

Show that it is a permutation by writing down the inverse function  $G_k^{-1}(\cdot||\cdot)$  explicitly (Note: We are **not yet** asking you to show that this is a *pseudorandom* permutation.)

**Solution:**  $G_k^{-1}(x||y) := (y \oplus F_k(x))||x$ .

(b) [4 points] Show that  $G_k$  is **not** a PRP family.

**Solution:**  $G_k$  is not a PRP family because it is not a PRF family. To see this,  $G_k(x||y)$  outputs  $y$  in the clear. Consider a distinguisher that feeds in  $0^n||0^n$  to the oracle and checks that the first  $n$  output bits are  $0^n$ . This happens with probability 1 for  $G_k$  and probability  $1/2^n$  for a truly random function. This advantage is  $1 - 1/2^n$ , which is non-negligible.

(c) [12 points] Now, let's try doing this transformation one more time. Define

$$H_{k_1, k_2}(x \| y) := G_{k_2}(G_{k_1}(x \| y)) = (F_{k_1}(y) \oplus x) \| (F_{k_2}(F_{k_1}(y) \oplus x) \oplus y).$$

Prove that  $H_{k_1, k_2}$  is **not** a PRP family.

**Solution:**  $H_{k_1, k_2}$  is not a PRP family because it is not a PRF family. To see this, the first  $n$  output bits are affine in  $x$  (over  $\mathbb{F}_2^n$ ). Consider the adversary that feeds in  $0^n \| 0^n$  and  $1 \| 0^{n-1} \| 0^n$  to the oracle, and checks whether first  $n$  output bits only differ in the first coordinate. This will happen with probability 1 for  $H_{k_1, k_2}$  but probability  $1/2^n$  for a truly random function.

**Problem 4.** [15 points]

Noga and Zoe are participants in an encryption design competition. They each came up with a novel public-key bit encryption scheme: let's call Noga's scheme  $\mathcal{E}_N = (\text{Gen}_N, \text{Enc}_N, \text{Dec}_N)$  and Zoe's scheme  $\mathcal{E}_Z = (\text{Gen}_Z, \text{Enc}_Z, \text{Dec}_Z)$ . Shafi, the judge, inspected both schemes and figured out that one of them is IND-CPA secure but the other one isn't. Unfortunately, she disappeared before telling them which one is secure!

Noga and Zoe decided to join forces and submit the following scheme  $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec})$  to the competition. Your task is to use a hybrid argument to prove that if either  $\mathcal{E}_N$  or  $\mathcal{E}_Z$  is IND-CPA secure, so is  $\mathcal{E}$ . State your hybrids clearly.

- $\text{Gen}(1^\lambda)$ : Generate keys  $(pk_N, sk_N) \leftarrow \text{Gen}_N(1^\lambda)$  and  $(pk_Z, sk_Z) \leftarrow \text{Gen}_Z(1^\lambda)$ . Output

$$pk = (pk_N, pk_Z) \quad \text{and} \quad sk = (sk_N, sk_Z).$$

- $\text{Enc}(pk, b)$  where  $b$  is a bit: generate a random bit  $r \leftarrow \{0, 1\}$  and output  $c = (c_N, c_Z)$  where

$$c_N \leftarrow \text{Enc}_N(pk_N, r) \quad \text{and} \quad c_Z \leftarrow \text{Enc}_Z(pk_Z, r \oplus b)$$

- $\text{Dec}(sk, c)$ : let  $b_N \leftarrow \text{Dec}_N(c_N)$  and  $b_Z \leftarrow \text{Dec}_Z(c_Z)$ . Output  $b_N \oplus b_Z$ .

*Note:* Make sure you handle both cases in your proof.

**Solution:** We split the proof into two parts, one in a world where  $\mathcal{E}_N$  is IND-CPA secure, and the other in a world where  $\mathcal{E}_Z$  is IND-CPA secure.

Part 1. Imagine a world where  $\mathcal{E}_N$  is IND-CPA secure. Our proof proceeds as follows. In all hybrid experiments, the adversary gets a pair  $(pk, c)$ .

Hybrid 0: Here  $pk$  and  $c$  are generated by running  $\text{Gen}(1^\lambda)$  and  $\text{Enc}(pk, 0)$  respectively. Note that  $c$  consists of  $(\text{Enc}_N(pk_N, r), \text{Enc}_Z(pk_Z, r))$  for a random bit  $r$ .

Hybrid 1: The only change from hybrid 0 is how we generate  $c$ . Here, we generate  $c$  as  $(\text{Enc}_N(pk_N, r \oplus 1), \text{Enc}_Z(pk_Z, r))$  for a random bit  $r$ .

Hybrid 2: Once again, the only change from hybrid 1 is how we generate  $c$ . Here, we generate  $c$  as  $(\text{Enc}_N(pk_N, r), \text{Enc}_Z(pk_Z, r \oplus 1))$  for a random bit  $r$ . Note that  $c$  here is identical to a ciphertext generated by running  $\text{Enc}(pk, 1)$ .

We now show why adjacent hybrids are indistinguishable. Hybrids 0 and 1 are computationally indistinguishable by the IND-CPA security of  $\mathcal{E}_N$ . Hybrids 1 and 2 are distributed identically.

Part 2. Imagine a world where  $\mathcal{E}_Z$  is IND-CPA secure. Our proof proceeds as follows. In all hybrid experiments, the adversary gets a pair  $(pk, c)$ .

Hybrid 0: Here  $pk$  and  $c$  are generated by running  $\text{Gen}(1^\lambda)$  and  $\text{Enc}(pk, 0)$  respectively. Note that  $c$  consists of  $(\text{Enc}_N(pk_N, r), \text{Enc}_Z(pk_Z, r))$  for a random bit  $r$ .

Hybrid 1: Here, we generate  $c$  as  $(\text{Enc}_N(pk_N, r), \text{Enc}_Z(pk_Z, r \oplus 1))$  for a random bit  $r$ . Note that  $c$  here is identical to a ciphertext generated by running  $\text{Enc}(pk, 1)$ .

Hybrids 0 and 1 are computationally indistinguishable by the IND-CPA security of  $\mathcal{E}_Z$ . This finishes the proof.