CS 6846: Quantum Algorithms and Cryptography Sep 13, 2022

Homework 2

Instructor: Shweta Agrawal *Due: Sep 23, 5 pm.*

Instructions:

1. Please type up your solutions using latex.
2. You may collaborate with other students. Please mention the names of your collaborators or any other source that you use for the solution.
3. Please type up your solutions *individually* without any help.

Problem 1: Negligible Functions (2+16+3 pts)

In cryptography, we usually define security by requiring that the probability of some undesirable event (e.g. Eve guesses the message) be so small that one would never notice it. To that end, we define a negligible function as follows:

Definition 1. *(Negligible function) A function* $\nu(k) : \mathbb{N} \mapsto [0,1]$ *is called* negligible *if for every polynomial* p*, there exists some* $k_0 \geq 1$ *such that for all* $k > k_0$*,* $\nu(k) < |1/p(k)|$*.*

In this problem we will develop some intuition for this useful concept and how to work with it.

a. Give an example of a negligible function $\nu(k)$ where $\nu(k) > 0$ for all k.

b. Suppose that ν is a negligible function. Let p be a polynomial such that $p(k) \geq 0$ for all $k > 0$. Which of the following functions are negligible?

   1) $\nu(p(k))$
   2) $p(\nu(k))$
   3) $\sum_{i=1}^{p(k)} \nu_i(k)$, where each ν_i is negligible
   4) $\nu(k) * p(k)$
   5) $\nu(k)^{\frac{1}{p(k)}}$
   6) $\nu(k)^{\frac{1}{c}}$, for some positive constant c
   7) $\frac{1}{p(k)} - \nu(k)$
   8) $\nu(k)^{-c}$, for some positive constant c

c. Suppose that $\epsilon : \mathbb{N} \mapsto [0,1]$ is not a negligible function. Does it follow that for some polynomial p (where $p(k) > 0$ for all k) and some k_0, $\epsilon(k) > 1/p(k)$ for all $k > k_0$? If your answer is yes, prove it. If your answer is no, give a counter-example.

Problem 2: One-Way Function: Definition (3+3 pts)

Recall the standard definition for a one-way function: A function $f : \{0,1\}^* \to \{0,1\}^*$ is called **one-way** if the following two conditions hold:

1. **Easy to compute:** There exists a deterministic polynomial-time algorithm A such that on input x, algorithm A outputs $f(x)$ (i.e. $(A(x) = f(x))$.
2. **Hard to invert:** For every probabilistic polynomial-time algorithm A, there exists a negligible function ν such that :

$$\Pr\left(A(1^k, f(x)) \to x' \mid x \xleftarrow{R} \{0,1\}^k \wedge f(x') = f(x)\right) \leq \nu(k)$$

   Notation: The above notation $x \xleftarrow{R} \{0,1\}^k$ means that x of length k is chosen uniformly at random from the set of k bit strings. The notation $A(1^k, f(x)) \to x'$ denotes that A takes as input $(1^k, f(x))$ and returns x'. The probability that A succeeds (i.e. $f(x') = f(x)$) is negligible.

Suppose we define the "hard to invert" part differently: A function $f : \{0,1\}^* \to \{0,1\}^*$ is called **uninvertible** if it is easy to compute f (as defined above), but there does *not* exist a probabilistic polynomial-time algorithm A such that, for every string x, on input $(1^k, f(x))$, A outputs x' such that $f(x) = f(x')$.

a. Show that if f is a one-way function, then it is an uninvertible function.

b. Below is a proof that an uninvertible function is also one-way. Is this proof correct? If not, describe where it went wrong (potentially in more than one place).

   Reduction: We show that an algorithm A that breaks the "one-wayness" of f also breaks that "uninvertibleness" of A. Thus, the reduction accomplishes the contrapositive: not one-way implies not uninvertible.

   The reduction proceeds as follows: on input $y = f(x)$, run A, giving it input y. With non-negligible probability, A outputs x' such that $f(x') = y = f(x)$. If A outputs such x', output it. Else, run A again until it does.

   Analysis of the reduction: Correctness follows because the reduction does not halt until it finds a correct x'. Expected polynomial-time follows because A outputs a correct x' with non-negligible probability $\epsilon(k)$, and $\epsilon(k) \geq 1/p(k)$ for some polynomial $p(k)$, so we need to run A $1/\epsilon(k) \leq p(k)$ times before it produces a correct x'.

   Therefore, if f is an uninvertible function, then it is also a one-way function.

Problem 3: Combining OWF (3+3+3 pts)

Let f, g be length preserving one way functions, i.e. $|f(x)| = |x|$. We will construct new functions f' using arbitrary one-way f, g. Prove or disprove that f' is one way for each of the following constructions. If it is, prove it, else provide a counter example.

a. $f'(x) = f(x) \oplus g(x)$

b. $f'(x) = f(f(x))$

c. $f'(x_1||x_2) = f(x_1)||g(x_2)$ (here $||$ denotes concatenation)

Problem 4: RSA(4 pts)

Recall the textbook RSA encryption scheme that we saw in class. Recall that we have a public modulus $n = p \cdot q$ where p, q are large primes. A user's public key is $e \in \mathbb{Z}^*_{\phi(n)}$ and secret key is d s.t. $e \cdot d = 1 \mod \phi(n)$. To encrypt a message m, a user computes the ciphertext as $\mathsf{CT} = m^e \mod n$ and to decrypt she computes $\mathsf{CT}^d \mod n$.

Assume that Anita and Brijesh have RSA keys with the *same* public modulus n but with different public exponents e_s and e_r respectively where e_s and e_r are relatively prime. Say that RSA encryption is used to send the *same* message m to both Anita and Brijesh. Prove that if Esha knows n, e_s, e_r and sees the two ciphertexts $c_s = m^{e_s} \mod n$ and $c_r = m^{e_r} \mod n$, she can reconstruct the message m.

Problem 5: More RSA (4 pts)

This problem shows why it is unsafe to use a very small public key in RSA. Suppose Ananya, Bharat and Chandra have the following RSA public keys — $(3, N_A)$, $(3, N_B)$, and $(3, N_C)$ respectively. Divya sends the message m to each one of them, encrypted using their respective public keys. Suppose that Esha is eavesdropping on the conversation and gets the three encrypted messages. Show how she can use these to reconstruct the original message m.

Problem 6: Inverting ElGamal (4 pts)

Consider (a variant of) an ElGamal cryptosystem, where $\mathsf{SK} = (p, g, y, x)$ where p is a large prime of size polynomial in the security parameter τ, g is a generator of group $\mathbb{Z}^*_p$, x is a random number in $\mathbb{Z}_{p-1}$, and $y = g^x \mod p$[1]. The public key used for encryption is $\mathsf{PK} = (p, q, y)$ and $\mathsf{SK} = (p, q, y, x)$ is used to decrypt.

The encryption works as follows. The input message is broken-down into blocks m s.t. each $m \in \mathbb{Z}^*_p$ and then each m is encrypted individually as follows:

1. $\mathsf{Enc}(\mathsf{PK}, m) = (c_1, c_2) = (g^r \mod p,\ y^r * m \mod p)$, where r is a random number in $\mathbb{Z}_{p-1}$ picked by the encryption algorithm. (Each ciphertext is a pair of $(c_1, c_2) \in (\mathbb{Z}^*_p, \mathbb{Z}^*_p)$.)
2. $m = \mathsf{Dec}(\mathsf{SK}, (c_1, c_2)) = c_2/(c_1)^x \mod p$.

Assume that someone creates an (efficient) algorithm $\mathcal{A}$ which decrypts ElGamal ciphertexts knowing just the public key, but only if c_1 starts with at least 5 leading zeroes, i.e. $c_1 = 00000....$ What's the advantage of $\mathcal{A}$ in breaking the one-wayness of ElGamal? Is it negligible?

Problem 7: Clarifying Advantage (8 pts)

The purpose of this problem is to clarify the concept of *advantage*. Consider the following two experiments $\mathsf{EXP}(0)$ and $\mathsf{EXP}(1)$:

- In $\mathsf{EXP}(0)$ the challenger flips a fair coin (probability 1/2 for HEADS and 1/2 for TAILS) and sends the result to the adversary $\mathcal{A}$.
- In $\mathsf{EXP}(1)$ the challenger always sends TAILS to the adversary $\mathcal{A}$.

[1]Recall that $\mathbb{Z}^*_p = \{1, \ldots, p-1\}$, $\mathbb{Z}_{p-1} = \{0, \ldots, p-2\}$, and that if g is a generator then for every element $y \in \mathbb{Z}^*_p$ there is a unique element $x \in \mathbb{Z}_{p-1}$ s.t. $y = g^x \mod p$. Therefore, in particular, if you pick x at random in $\mathbb{Z}_{p-1}$, element $y = g^x \mod p$ is itself random, i.e. uniformly distributed, in $\mathbb{Z}^*_p$.

The adversary's goal is to distinguish these two experiments: at the end of each experiment the adversary outputs a bit 0 or 1 for its guess for which experiment it is in. For $b = 0, 1$ let W_b be the event that in experiment b the adversary output 1. The adversary tries to maximize its distinguishing advantage, namely the quantity

$$\mathsf{Adv} = |\Pr[W_0] - \Pr[W_1]| \in \{0, 1\}$$

The advantage Adv captures the adversary's ability to distinguish the two experiments. If the advantage is 0 then the adversary behaves exactly the same in both experiments and therefore does not distinguish between them. If the advantage is 1 then the adversary can tell perfectly what experiment it is in. If the advantage is negligible for all efficient adversaries (as defined in the class) then we say that the two experiments are indistinguishable.

1. Calculate the advantage of each of the following adversaries:
   - $\mathcal{A}_1$: Always output 1.
   - $\mathcal{A}_2$: Ignore the result reported by the challenger, and randomly output 0 or 1 with even probability.
   - $\mathcal{A}_3$: Output 1 if HEADS was received from the challenger, else output 0.
   - $\mathcal{A}_4$: Output 0 if HEADS was received from the challenger, else output 1.
   - $\mathcal{A}_5$: If HEADS was received, output 1. If TAILS was received, randomly output 0 or 1 with even probability.
2. What is the maximum advantage possible in distinguishing these two experiments? Explain why.

Problem 8: Random Oracle (9 pts)

Let b denote a given "message block size" (e.g. $b = 512$ bits). For this problem, assume all messages are exactly k blocks long, for some moderate k (e.g. $k = 1000$). Each message has length bk bits. Let n denote a given desired hash output size, in bits (e.g. $n = 160$).

Let $Maps(t, u)$ denote the set of all possible functions with domain $\{0,1\}^t$ and range $\{0,1\}^u$ A randomly chosen function from $Maps(t, u)$ may be viewed as a "random oracle" (from t-bit strings to u-bit strings). Ideally, a hash function should be indistinguishable from a random oracle with the same domain and range. However, in practice this may not be the case, due to the manner in which the hash function is constructed.

1. Suppose f is a random oracle drawn from $Maps(bk, n)$. Suppose you draw values $x_1, x_2, \ldots$ uniformly at random from $\{0,1\}^{bk}$, and for each such x_i you compute $f(x_i)$. How many such x's do you expect to have to try before you find a "collision" (a pair of distinct x_i, x_j values such that $f(x_i) = f(x_j)$)? (No need for proof here. Also, your answer does not need to be exact, just a reasonable approximation.)
2. Now suppose that hash function h mapping $\{0,1\}^{bk}$ to $\{0,1\}^n$ is constructed in a serial fashion from the random oracle g drawn from $Maps(b + n, n)$, as follows. To compute $h(M)$ where $M = m_1||m_2||\ldots||m_k$ (and each m_i is b-bits long):
   - Let $v_0 = 0^n$.
   - Let $v_i = g(v_{i-1}||m_i)$ for $i = 1, 2, \ldots, k$. The function g takes $n + b$ bits and hashes them down to n bits.

- Let $h(M) = v_k$.

Argue that an adversary can distinguish such a hash function h from a random oracle such as the one in Part (1) having the same domain and range, by looking for collisions in a certain way. Assume that the adversary can perform an arbitrary number of evaluations of h but cannot evaluate g.