Introduction to Cryptography | **Instructor: Oded Regev**
Courant, Fall 2015 | **Student: YOUR NAME HERE**

Homework 0

Homework is due by **7am of Sep 14**. Send by email to both "regev" (under the cs.nyu.edu domain) and "mgeorgiou@nyu.edu" with subject line "CSCI-GA 3210 Homework 0" and name the attachment "YOUR NAME HERE HW0.tex/pdf". No need for a printed copy. Start early!

Instructions. Solutions must be typeset in LaTeX (a template for this homework is available on the course web page). Your work will be graded on *correctness*, *clarity*, and *conciseness*. You should only submit work that you believe to be correct; if you cannot solve a problem completely, you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) "proof summary" that describes the main idea.

You are expected to read all the hints either before or after submission, but before the next class.

You may collaborate with others on this problem set and consult external sources. However, you must ***write your own solutions***. You must also ***list your collaborators/sources*** for each problem.

1. Send email to Oded (`regev` at `cims`) with subject `CSCI-GA 3210 student` containing (1) a few sentences about yourself and your background (including your department, graduate program, how long in program), and (2) your comfort level with the following: mathematical proofs, elementary probability theory, big-O notation and analysis of algorithms, Turing machines, and P, BPP, NP, and NP-completeness. Please also mention any courses you've taken covering these topics.

2. *(Working with negligible functions.*[1]*)* Recall that a non-negative function $\nu : \mathbb{N} \to \mathbb{R}$ is *negligible* if it decreases faster than the inverse of any polynomial (otherwise, we say that ν is *non-negligible*). More precisely, $\nu(n) = o(n^{-c})$ for every fixed constant $c > 0$, or equivalently, $\lim_{n\to\infty} \nu(n) \cdot n^c = 0$.

   State whether each of the following functions is negligible or non-negligible, and give a brief justification. In the following, $\mathrm{negl}(n)$ denotes some arbitrary negligible function, and $\mathrm{poly}(n)$ denotes some arbitrary polynomial in n. (If you are not comfortable with these notion, read Section 4.2 of Lecture 2 in Peikert's notes)

   (a) (1 point) $\nu(n) = 1/2^{100 \log n}$.

   (b) (1 point) $\nu(n) = n^{-\log\log\log n}$. (Compare with the previous item for "reasonable" values of n.)

   (c) (1 point) $\nu(n) = \mathrm{poly}(n) \cdot \mathrm{negl}(n)$. (State whether ν is *always* negligible, or not necessarily.)

   (d) (1 point) $\nu(n) = (\mathrm{negl}(n))^{1/\mathrm{poly}(n)}$. (Same instructions as previous item.)

   (e) (1 point)
   $$\nu(n) = \begin{cases} 2^{-n} & \text{if } n \text{ is composite} \\ 100^{-100} & \text{if } n \text{ is prime.} \end{cases}$$

3. *(The group $\mathbb{Z}_p^*$)* Let p be an odd prime. We use $\mathbb{Z}_p^*$ to denote the multiplicative group of integers modulo p. (In mathematics the common notation is $(\mathbb{Z}/p\mathbb{Z})^*$.)

   (a) (1 point) Find an efficient algorithm that given $a \in \mathbb{Z}_p^*$ and an integer $b \geq 0$ computes $a^b \in \mathbb{Z}_p^*$. Can we simply compute a^b as integers and then reduce the result modulo p? (if not, say exactly why)

   (b) (2 points) Find an efficient algorithm to check if a given $a \in \mathbb{Z}_p^*$ is a quadratic residue.

[1] Based on a question from Peikert's class

(c) (2 points) What fraction of the elements of $\mathbb{Z}_p^*$ are generators? How does it behave asymptotically? (You can use Wikipedia for the latter; there is no need for very precise asymptotics, just the order of magnitude)

(d) (2 points) Describe an efficient algorithm to check if a given $g \in \mathbb{Z}_p^*$ is a generator. Assume that the algorithm is also given a factorization of $p-1$. (It is not known how to perform this task efficiently without this factorization.)

(e) (2 points) There is a known efficient algorithm that given a number n (in unary) outputs a uniform n-bit prime p, together with a generator g of $\mathbb{Z}_p^*$. How can that be in light of what we said earlier about the necessity of the factorization of $p-1$? Explain the apparent paradox and suggest a solution.

4. (5 points) *((In)distinguishability warm-up.)* Let D_0, D_1 be two distributions. Let $\mathcal{A}$ be an algorithm that outputs either 0 or 1. We say that $\mathcal{A}$ can distinguish the distributions D_0 and D_1 with probability ε if

$$\Pr_{x \leftarrow D_0}[A(x) = 0] - \Pr_{x \leftarrow D_1}[A(x) = 0] = \varepsilon \, .$$

Now suppose that we play the following game with $\mathcal{A}$. First, we pick at random a bit $b \leftarrow \{0,1\}$ and then we pick $x \leftarrow D_b$ and we give x to $\mathcal{A}$. Finally, $\mathcal{A}$ returns a bit $\mathcal{A}(x)$. Show that the probability that $\mathcal{A}$ returns the correct bit b is $1/2 + \varepsilon/2$ if and only if $\mathcal{A}$ can distinguish the distributions D_0 and D_1 with probability ε. Equivalently,

$$\Pr_{x \leftarrow D_0}[A(x) = 0] - \Pr_{x \leftarrow D_1}[A(x) = 0] = \varepsilon \iff \Pr_{\substack{b \leftarrow \{0,1\} \\ x \leftarrow D_b}}[A(x) = b] = \frac{1}{2} + \frac{\varepsilon}{2}$$

5. *(Error-correcting codes (optional, no credit).)* This is a bit off topic, but will give you an idea of the kind of math we use in this course. It will also give you a glimpse to an immensely important topic that also dates back to Shannon's seminal work. These ideas are used in pretty much all digital communication protocols: cell phones, Internet, satellites, etc.

(a) Assume we choose $2^{n/20}$ strings from the set $\{0,1\}^n$ uniformly at random. Show that with positive probability (in fact, high probability) the Hamming distance (i.e., number of different coordinates) between *any* two strings in the set is more than $n/4$. I need a hint! (ID 84542)

(b) Show how Alice can communicate to Bob a message of k bits by sending only $n = 20k$ bits in such a way that Bob can recover the message even if an adversary flips up to $n/8$ bits of the communication. Would simply repeating the message 20 times be good enough?