Introduction to Cryptography
Courant, Fall 2016

Homework 4

Instructor: Oded Regev
Student: YOUR NAME HERE

Homework is due by **7am of Oct 17**. Send by email to both "regev" (under the cs.nyu.edu domain) and "avt237@nyu.edu" with subject line "CSCI-GA 3210 Homework 4" and name the attachment "YOUR NAME HERE HW4.tex/pdf". There is no need to print it. 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. (3 points) *(Rabin's permutation)* Assume $p, q \equiv 3 \pmod 4$. Does Rabin's function remain one way when its domain is restricted to $\mathbb{QR}_N^*$ (and so becomes a one way *permutation*)?

   Solution: Yes, it does. Suppose that restricting Rabin's function to be a permutation made it so that it is no longer one way. Let A be the inversion algorithm for the one-way permutation that succeeds with probability P (non-negligible). Let $y = A(x^2)$. clearly if y was a successfl inversion in the restricted domain, then $y \in \sqrt{x^2}$ in the unrestricted comain, and is therefore also a success. If the inversion was a failure in the restricted domain, it is possible that the inversion yielded a different root of x than the one that is a quadratic residue, and thus it could still be a success in the unrestricted case. Since the probability of success in the unrestricted case is at least as great as the probability in the restricted case, the easiness of inverting Rabin's Permutation clearly implies the easiness of inverting Rabin's Function. Since Rabin's function is one-way, Rabin's permutation must also be one-way.

2. *(PRGs.)*[1] Prove or disprove (giving the simplest counterexample you can find) the following statements. In constructing a counterexample, you may assume the existence of another OWF / PRG.

   (a) (4 points) Let G be a PRG with output length $\ell(n) > n$. The function $G'(s) = G(s) \oplus (s|0^{\ell(|s|)-|s|})$ is a PRG, where $|$ denotes concatenation. I need a hint! (ID 99102)

   Solution: No, not necessarily. Consider the following counterexample: suppose we have a function G that is based on Rabin's permutation, such that $(G(x) = (f(x), \oplus_{i=1} n x_i)$. We claim that this is a pseudorandom generator. Clearly the outputs a string of length n+1¿n, so we just need to show that there is no PPT distinguisher D that suceeds with non-negligible probability. I claim that the existence of a such a distinguisher implies the existence of a PPT inversion algorithm A that succeeds in inverting Rabin's permutation with noticable probability. We can construct A as follows: whenever $D = 1$,

   (b) (5 points) A PRG G with output length $\ell(n) = 2n$ is itself a one-way function. I need a hint! (ID 15489)

[1] A question from Peikert's class

Solution: Let $G : 0,1^n \rightarrow 0,1^{2n}$ be our PRG. Now suppose for contradiction that G is not a one-way function. Then there exists some PPT inversion algorithm A such that $\Pr_{x\in\{0,1\}^n}[A(1^n, G(x)) \in G^{-1}(G(x))] = \frac{1}{poly(n)}$. I claim that this inversion algorithm allows us to construct a PPT distinguisher algorithm, D. To construct such an algorithm, we could, given some y sample from either G or the uniform distribution, calculate $x` = A(y)$, and then test to see whether $G(x`) = y$, with D outputting 1 if there is a success, and 0 otherwise. Note that the Im(g) constitutes a fraction of $2^{|s|-l(|s|)}$ of the possible strings in our range, so if the string given to the distinguisher is from the uniform distribution, this is the probability that it is in Im(G). Consider what happens if we are given a sample $s \notin Im(G)$. Either the inversion algorithm is not defined for that input, outputs an incorrect answer, or the null set – but in any case, we can be sure that $G(x`) \neq y$. Thus
$P[D(G(s)) = 1] = \frac{1}{poly(n)}$, while
$P[D(u) = 1] = P[D(u) = 1|u \in Im(G)] * P[u \in Im(G)] + P[D(u) = 1|u \notin Im(G)] * P[u \notin Im(G)]$
$= \frac{1}{poly(n)} * 2^{|s|-l(|s|)} + 0.$
Thus $|P[D(G(s)) = 1] - P[D(u) = 1]| = \frac{1}{poly(n)}(1 - 2^{|s|-l(|s|)})$, which is not a negligible function. This contradicts the fact that G is a PRG, and therefore implies that G is a one-way function.

(c) (4 points) (extra credit) A PRG G with output length $\ell(n) = n + 1$ is itself a one-way function. I need a hint for 1 points! (ID 19634)

Solution:

3. (a) (2 points) *(Computing square roots efficiently modulo prime)* Let $p > 2$ be a prime. Assume we are given a quadratic residue $x \in \mathbb{Z}_p^*$ and we wish to compute its (two) square roots. Show that when $p \equiv 3 \bmod 4$, this can be done efficiently by computing $\pm x^{(p+1)/4}$, a formula due to Lagrange. (The case $p \equiv 5 \bmod 8$ is a bit more difficult; the case of a general prime can also be done efficiently but is more involved; feel free to look it up and summarize it here!)

Solution: If $p \equiv 3 \bmod 4$, then we know that $\frac{p+1}{4}$ is an integer. And since x is a quadratic residue, we know (from homework 0) that $x^{\frac{p-1}{2}} = 1$. Multiplying both sides by x, we have $x^{\frac{p+1}{2}} = x$. Now we can take the sqaure root of both sides, giving $\sqrt{x} = \pm x^{(p+1)/4}$. It was crucial here that $\frac{p+1}{4}$ is an integer, so that the exponent is well-defined. Note that we can compute this exponent efficiently using the same technique outlined in homework 0. Finally, we can simply take the inverse of the result in order to computethe other square root, which can also be done very efficiently.

(b) (4 points) *(LSB is not hard.[1])* Show how given a prime $p > 2$, a generator g of $\mathbb{Z}_p^*$, and $g^x \bmod p$ for an unknown $x \in \{0, \ldots, p-2\}$, we can efficiently decide if x is odd. (This shows that "least significant bit" [x is odd] is *not* a hard-core predicate for the modular exponentiation function $f_{p,g}(x) = g^x \bmod p$.)

Solution: if x is prime, then it ca be written as a product of odd primes: $x = p_1 * ... * p_k$. Note that since $\frac{x}{2}$ is an integer if and only if $g^x = a$ is a quadratic residue. As discussed in homework 0, we can compute whether x is a quadratic residue by testing whether $x^{\frac{p-1}{2}} = 1$, which can be done efficiently using efficient modular exponentiation as described in homework 0. Therefore the problem of determining the least significant bit is equivalent to the problem of finding quadratic residuosity, which is not hard when we have a prime modulus.

(c) (2 points) Here is a sketch of an attempt to efficiently compute discrete logs (a problem believed to be hard). Complete the missing details and identify the bug.

We are given $y = g^x \bmod p$ for an unknown $x \in \{0, \ldots, p-2\}$. Write $x = \sum_{j=0}^{\lceil \log p \rceil} 2^j b_j$ in its binary expansion. Efficiently find b_0 as above. Let $y_1 = y/g^{b_0}$ and notice that it is a quadratic residue. Compute the square root of y_1, and continue recursively to recover all the bits of x.

Solution: As noted above, we can find the least significant bit by determining whether y is a quadratic residue. If we write y as $y = g^{\sum_{j=0}^{\lceil \log p \rceil} 2^j b_j}$, we can see that $y_1 = \frac{y}{g^{b_0}} = g^{\sum_{j=1}^{\lceil \log p \rceil} 2^j b_j}$ is the same as setting the LSB of y to 0. We claim that y_1 is in fact a quadratic residue. Note that if x was even, then $y_1 = \frac{y}{g^{b_0}} = y$, and we know that y is a quadratic residue from the previous problem, so y_1 is also a quadratic residue. If x is odd, then $y_1 = \frac{y}{g^{b_0}} = \frac{y}{g}$. In other words, we subtract 1 from the exponent x, so that it becomes even, and therefore y_1 is a quadratic residue. Thus y_1 is a quadratic residue. Now, taking the square root of y_1 is the same thing as dividing the exponent by two, which in binary representation is the same as shifting all of the bits of x to the right. Denote this new exponent with its bits shifted to the right as x_1. Then We can repeat this process starting with x_1 and y_1 in order to find the LSB of x_1, which is the second bit in our original exponent x, and continue with this process to recover successive bits of x. However, this process cannot continue for the full length of x.

4. (2 points) ♣ *(Using hard core predicates to construct PRGs)* We say that an efficiently computable function $h : \{0,1\}^* \to \{0,1\}$ is *hard-core* for a function f if for all non-uniform PPT algorithms $\mathcal{A}$,

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(f(x)) = h(x)] \leq \frac{1}{2} + \mathrm{negl}(n) \ .$$

Assume we're able to show that a certain h is hard-core for a one-way *permutation* f. Suggest a way to construct a PRG from f and h, and try to think what the analysis would entail. (We'll do the analysis in class and in the next homework.)

Solution: