CS 127/CSCI E-127: Introduction to Cryptography

Problem Set 4

Assigned: Oct. 4, 2013 Due: Oct. 11, 2013 (5:00 PM)

Justify all of your answers. See the syllabus for collaboration and lateness policies. Submit solutions by email to `mbun@seas` (and please put the string "CS127PS4" somewhere in your subject line).

Problem 1. (Computational Number Theory) For the programming parts of this problem, describe in prose what the main ideas are and justify your major design choices. Attach the source code in an appendix to your submission, but **please keep your entire submission contained in one PDF file**.

a) Let $N = 453504209$. Write a program to extract a square root of 105592908 in $\mathbb{Z}_N^*$ using brute-force search. How long does it take to find the square root? (You might need to run your program multiple times and take an average.)

b) If the modulus is a prime p congruent to 3 modulo 4 (i.e. $[p \bmod 4] = 3$), extracting square roots modulo p is very easy. Specifically, prove that for every $y \in \mathrm{QR}_p$, $[\pm y^{(p+1)/4} \bmod p]$ are the square roots of y in $\mathbb{Z}_p^*$. (Hint: use the fact that $\mathbb{Z}_p^*$ has a generator.)

c) Suppose that you know the factorization of $N = p \cdot q$ ($453504209 = 20743 \cdot 21863$), and that p and q are both congruent to 3 modulo 4. Implement a more efficient algorithm to find the square roots of 105592908 in $\mathbb{Z}_N^*$ given this factorization. How long does it take to find the square roots this time? (Hint: Use the Chinese Remainder Theorem. This part may require substantially more programming than the first part.)

d) Compare the time it takes to find square roots of random numbers in $\mathbb{Z}_N^*$ using the approaches in parts a) and c) as the bit-lengths of p and q increase (but only let the brute-force search run for as long as your patience allows). Draw a graph to illustrate your findings. Here is a table of primes congruent to 3 modulo 4 to help you:

bit-length	p	q
12	3011	3967
13	4547	6803
14	14683	13963
15	26479	18367
16	63311	49451
17	81611	77611
18	224831	256643
19	278651	475807
20	769487	915451

(Be careful to use at least 64-bit integers when working with the larger primes.)

Problem 2. (More candidate one-way functions) Which of the following functions are likely to be one-way functions? Justify your answers by either giving a polynomial-time adversary that inverts the function with nonnegligible probability or by showing that the function's one-wayness follows from the one-wayness of one of the candidates given in class.

a) $f(x) = x^2$, where x is an integer.

b) $f(x, y) = x \cdot y - 2^{\lceil \|x\|/2 \rceil} \cdot y$, where x and y are integers such that $\|x\| = \|y\|$.

c) $f(x_1, \ldots, x_n, S) = \left(x_1, \ldots, x_n, \left[\sum_{i \in S} x_i \bmod n^2\right]\right)$, where each $x_i \in \{1, \ldots, n^2\}$ and $S \subseteq \{1, \ldots, n\}$.

d) Extra credit: $f(x_1, \ldots, x_n, S) = \left(x_1, \ldots, x_n, \sum_{i \in S} x_i\right)$, where each $x_i \in \{1, \ldots, n^2\}$ and $S \subseteq \{1, \ldots, n\}$.

Problem 3. (Modifying one-way functions) Suppose we are given a one-way function f. For which of the constructions is g always a one-way function (for every choice of f)? Prove your answers by either a reducibility argument or by constructing a one-way function f (e.g. by modifying a candidate or arbitrary one-way function) for which g fails to be one-way.

a) $g(x) = f(0^{|x|}x)$

b) $g(x, y) = f(x \oplus y)$, where $|x| = |y|$