YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CPSC 467a: Cryptography and Computer Security *Handout #14*
Professor M. J. Fischer *November 5, 2008*

Problem Set 5

Due before midnight on Friday, November 14, 2008.

Goldwasser-Micali Probabilistic Encryption

Write three computer programs to implement the Goldwasser-Micali Probabilistic cryptosystem as described in section 67 of lecture notes 15. The three programs are:

Key generation Takes three command line arguments: a security parameter s, which is a decimal integer in the range $[1 \ldots 2048]$, the name of a public key output file, and the name of a private key output file. The program generates random distinct odd primes p, q, each $\lceil s/2 \rceil$-bits long, computes their product n, and finds a number $y \in Q_n^{00}$. It writes the public key (n, y) to the public key output file as a pair of whitespace-separated decimal integers. It writes the private key p to the private key output file as a single decimal integer.

Encryption Takes three command line arguments: the name of a public key file, the name of a message file, and the name of a ciphertext output file. A message consists of a sequence of "0" and "1" ASCII characters. Whitespace and any other characters in the message file are ignored. Encrypts the message bit by bit using the public key and writes the resulting numbers to the ciphertext output file as a sequence of decimal integers.

Decryption Takes three command line arguments: the name of a private key file, the name of a ciphertext file, and the name of a plaintext output file. Decrypts the ciphertext using the private key and writes the decrypted bits to the plaintext output files as a sequence of ASCII "0" and "1" characters.

Your program should be written in C, C++, or Java and should use one of the big number libraries discussed in Section 39 of lecture notes 8 (if using C or C++) or using the appropriate Java class libraries (if using Java). You may use any of the provided functions in solving this problem. In particular, you do not need to implement your own primality testing function, modular exponentiation function, or random number generator if the versions provided by the package are adequate for this problem.