Introduction to Cryptography, Fall 2021

Homework 3

Due: 1pm, 11/10/2021 (Wednesday)

Part 1: Written Problems

1. Suppose you have an identical and independent source of bits, where bit 0 is generated with probability 0.4 and bit 1 is generated with probability 0.6.
   A. Design a conditioning algorithm to generate a bit string with independent bits, where 0 and 1 appear with probability 0.5 each.
   B. What is the expected number of input bits in order to generate an output bit?

2. Write a BBS-generator program with n=23858977l and seed=7477 to generate a string of 1,000,000 bits.
   A. Compute the ratios of bits 0, 1. Are both of them around 50%? If not, why?
   B. Compute the ratios of bit pattern '00', '01', '10', and '11'. Are all of them around 25%? If not, why? **Note**: 00011 is counted as two '00', one '01 and one '11'. The ratios are 50%, 25%, 0% and 25%, respectively.

3. Alice and Bob use the same RSA modulus n=143. Assume that Alice's key exponents e=7 and d=103 and Bob's public key exponent e=13. Assume that David encrypts a message as C=60 with Bob's public key for Bob.
   A. Factor n, compute Bob's private key and decrypt C.
   B. Show that Alice can decrypt C without factoring n=143.

4. Alice and Bob use the Diffie–Hellman key exchange technique with a common prime q = 131 and a primitive root $\alpha = 6$. If Alice chooses $X_A = 15$ and Bob chooses $X_B = 27$, what are Y_A, Y_B and the shared secret by the method?

5. Alice and Bob use the ElGamal scheme with a common prime q = 131 and a primitive root $\alpha = 6$. Let Bob's public key be $Y_B = 3$.
   A. What is the ciphertext of M=9 if Alice chooses the random integer k=4?
   B. If Alice uses the same k to encrypt two messages M_1 and M_2 as (12, 65) and (12, 64), what is the relation between M_1 and M_2?

6. Consider the elliptic curve $y^2 = x^3 + 3x + 1$ over Z_7. Assume that G = (3, 3) and Bob's private key is $n_B = 4$.
   A. Compute all the points over the curve.
   B. What is Bob's public key P_B?
   C. Alice wants to encrypt message $P_m = (2, 1)$ to Bob and chooses the random value k = 3. What is the ciphertext C_m?
   D. Decrypt the ciphertext ((5, 1), (2, 6)) using Bob's private key.

Part 2: Programming Problem

This programming problem is to practice RSA encoding and decoding using Crypto++. We only deal with one-block operation. You need to check whether the message length (in bits) is strictly shorter modulus n's length.

I. Read in the key length in decimal, a public key (e, n) in Hex and a message in ASCII and do encryption as described in the following table. The first row is for testing and the rest is the problem.

The ASCII message is treated as an integer, for example, "Hi" = "4869" (Hex) = 18537 (decimal). Since we are dealing with very long integer, use "Integer" class for integer operations.

Key length (decimal)	Public key=(e, n) (Hex)	Message (ASCII)	Ciphertext (Hex)
64	(11, c963f963d93559ff)	ElGamal	6672e7d4a8786631
128	(11, 04823f9fe38141d93f1244be161b20f)	Hello World!	?
256	(10001, 9711ea5183d50d6a91114f1d7574cd5262 1b35499b4d3563ec95406a994099c9)	RSA is public key.	?

II. Read in key length in decimal, e and n in Hex and a ciphertext in Hex. Try to find the correct private key d by searching thru possible keys as specified. Then, use the private to decrypt the ciphertext to a meaningful message.

Key length (decimal)	e, n (Hex)	Ciphertext (Hex)	Private key (Hex)	Message (ASCII)
64	11, c45350fa19fa8d93	a4a59490b843eea0	454a950c5bcbaa41	secrecy
128	1d35, c4b361851de35f080d3 ca7352cbf372d	a02d51d0e87efe1de fc19f3ee899c31d	53a0a95b089cf23adb5 cc73f07XXXXX ?	?

III. Submission: you need to upload two files: rsa.cpp and out.txt, where rsa.cpp solves the above two problems and out.txt consists of 4 lines for the answers in the above problems.

IV. If you want to generate some random RSA keys for practice, try the following program segment:

```
#include "rsa.h"
#include "osrng.h"
// random number generator
AutoSeededRandomPool rng;

InvertibleRSAFunction parameters;

// Generate RSA keys with key_length bits
int key_length = 256;
parameters.GenerateRandomWithKeySize(rng, key_length);

const Integer& n = parameters.GetModulus();
const Integer& p = parameters.GetPrime1();
const Integer& q = parameters.GetPrime2();
const Integer& d = parameters.GetPrivateExponent();
const Integer& e = parameters.GetPublicExponent();
```