Algorithms in the Real World (15-853), Spring 2018
Assignment #3 Due: 6 March 2018

Complete all problems.
You are not permitted to look at solutions of previous year assignments. You can work together in groups, but all solutions must be written up individually. If you get information from sources other than the course notes and slides, please cite the information, even if from Wikipedia or a textbook.
Please provide a simple illustration of your solution, more than just an answer.

Problem 1: Number Theory basics (10pt)

A. For what values of n is $\phi(n)$ odd?

B. Show that if $d|m$ (i.e. d divides m), then $\phi(d)|\phi(m)$.

Problem 2: McEliece Cryptosystem (20pts)
The McEliece Cryptosystem is one of the earliest public-key systems (published the same year as the RSA cryptosystem). It is based on linear error-correcting codes. Like the Merkle-Hellman Cryptosystem it uses an easy special case of an NP-complete problem which is disguised as a general case.
Recall that a linear code can be defined by a $k \times n$ binary generator matrix G, where k is the size of the plaintext, and n is the size of the codewords. A plaintext x is converted into a codeword y using $y = xG$. The rows of G form the basis of the code G, and the minimum Hamming distance between any two codewords generated by G is called the *distance* of G. A code $k \times n$ code with distance d is denoted as a $[n, k, d]$ code.
The NP-complete problem used by McEliece is the following. Given an n-bit word w find the nearest codeword (Hamming distance) that can be generated by a general $k \times n$ code G. This is exactly what error correcting codes are supposed to find. Fortunately, for error correcting codes, solving this problem for certain codes is easy. The trick of McEliece is therefore to use one of these easy codes and transform it to look like the general case.
McEliece suggested using Goppa codes as the "easy" codes. Goppa codes have the form $[n, k, d] = [2^m, 2^m - mt, 2t + 1]$ (McEliece suggested taking $m = 10$ and $t = 50$ giving $[1024, 524, 101]$). Given a word w of length n, and a Goppa code G, there is a polynomial (in t and n) algorithm for finding the nearest codeword in G assuming t or fewer errors.
Encryption in McEliece's system is based on an $k \times n$ Goppa code G, a $n \times n$ permutation matrix P, and an $k \times k$ invertible binary-matrix S. These three matrices are all part of the private key (G, S, P). The public key is the matrix $G' = SGP$. This is itself an $k \times n$ linear code, but it is not a Goppa code. A k-bit word x is encrypted using $w = xG' + e$ where e is a random n-bit vector of weight t (i.e. t 1s in it). The hope is that given that G' is not a Goppa code, it is hard to get x back from w. No one has yet broken this method (i.e. found a polynomial-time solution).

A. Describe how to decode in polynomial time given the private key.

B. Given unbounded memory and unbounded preprocessing time for a given public-key, explain how you can set things up so that you can quickly decode messages w. As a function of n and t, how much memory would this take (try to minimize this, but don't get carried away by trying to compress the data)?

C. Given a $[1024, 524, 101]$ Goppa code, assume you want to correct up to 10 bit-errors as well as use it for encryption. How do you change the encryption algorithm? How does this affect security?

D. What are some advantages or disadvantages of the McEliece cryptosystem as compared to RSA? Don't worry about time, but you can assume that using a $[1024, 524, 101]$ Goppa code gives about the same security as a 766-bit RSA modulus. (I'm thinking of one advantage and three disadvantages, but your's don't have to match mine.)

Problem 3: RSA (10pt each)
The following two questions exhibit what are called *protocol failures*. They shows how one can break a cryptosystem if the cryptosystem is used carelessly.
You can solve either one of the two questions or both.

A. Joe Hacker decides that he wants to have two public-private key pairs to be used with RSA—he feels that two is more prestigious than one. In his infinite wisdom, he decides to use a common value for $n = pq$ and selects two separate encryption exponents e_1 and e_2, giving two distinct decryption keys d_1 and d_2. He makes e_1, e_2 and n public. You can assume that e_1 and e_2 are relatively prime.

Assume Alice and Bob send the same secret plaintext message m to Joe one encrypted with e_1 and the other with e_2. Suppose that Eve is eavesdropping on the conversation and gets the two encrypted messages. Show how she can use these to reconstruct the original message m. Hint, for any positive integers a and b, the extended Euclid's algorithm finds integers r and s such that $ra + sb = \gcd(a, b)$.

B. This problem shows why it is unsafe to use a very small public key in RSA. Suppose Alice, Bob and Carol have the following RSA public keys—$(3, N_A)$, $(3, N_B)$, and $(3, N_C)$ respectively. Joe sends the message m to each one of them, encrypted using their respective public keys. Suppose that Eve is eavesdropping on the conversation and gets the three encrypted messages. Show how she can use these to reconstruct the original message m.

Problem 4: El-Gamal (15pt)
We give an example of the ELGamal Cryptosystem implemented in GF(3^3) using x^3+2x^2+1 as the irreducible polynomial. We can associate the 26 letters of the alphabet with the 26 nonzero field elements, and thus encrypt ordinary text. The associations are given on below. Suppose Bob uses the polynomial x as the generator (g in the notes) and uses 11 as the random power (i.e., his private key: x in the notes, and a in the slides). Show how Bob will decrypt the following string of ciphertext:

```
(K,H)(P,X)(N,K)(H,R)(T,F)(V,Y)(E,H)(F,A)(T,W)(J,D)(U,J)
```

We would advise writing a small program for this. (You may use any method you like to perform the decryption.) Although it might be quicker to do by hand, it would be quite tedious. Also writing a program will help you understand how to implement the Galois Field operations.

A	1	J	$1+x^2$	S	$1+2x^2$
B	2	K	$2+x^2$	T	$2+2x^2$
C	x	L	$x+x^2$	U	$x+2x^2$
D	$1+x$	M	$1+x+x^2$	V	$1+x+2x^2$
E	$2+x$	N	$2+x+x^2$	W	$2+x+2x^2$
F	$2x$	O	$2x+x^2$	X	$2x+2x^2$
G	$1+2x$	P	$1+2x+x^2$	Y	$1+2x+2x^2$
H	$2+2x$	Q	$2+2x+x^2$	Z	$2+2x+2x^2$
I	x^2	R	$2x^2$		

Problem 5: Square roots (10pt)
When p and q are distinct odd primes, $n = pq$, points in Z_n^* have either zero or four square roots. A quarter of the points have four square roots; the rest have no square root. The four square roots will look like a; $n-a$, b, and $n-b$. Suppose Harry designs an efficient algorithm S that, on input of a square x, finds some square root of x. You don't know which. Use S to make an efficient (probabilistic) algorithm F that factors n. (Hint: try to find two square roots of a number, a and b, which are not of the form $a = \pm b \pmod{n}$. Show how, if you do this, you can factor n.)