Problem Set 4

Due: Lesson 24

(50 pts)

1. [5] Is 3 a primitive root of 43? Show your work using modular exponentiation. (HINT: Don’t raise 3 to all possible positive values in the mod space – take the shortcut!)

43-1 = 42 = 2\*3\*7. This gives 21,14,6. 321 (mod 43) ≡ 42. 314 (mod 43) ≡ 36. 36 (mod 43) ≡ 41. Thus, 3 is not a primitive root of 43.

2. [5] Suppose the key generation algorithm in your implementation of RSA sometimes generates a large *p* or *q* that is composite. What effect will this have? Be specific.

This will cause the *n* to be factored much more easily when a composite is multiplied with a prime. Because of this, there will also be more chances of possible multiplicative inverses of the totient of *n* (mod *n*). If this happens, it will be easier to break the encryption and decrypt the message.

3. [5] Your friend is using an RSA like encryption scheme in which they encrypt a message *m* by computing 𝑐≡ 𝑚11(𝑚𝑜𝑑 127). What is your decryption exponent such that 𝑚≡ (𝑚𝑜𝑑 127)? (Note: Your *n*, 127, is prime). Is this as strong as RSA? Why or why not?

No, this is not as strong as RSA. This is because given that *n* is prime, then the totient of *n*, which is published as part of the public key, is *(n-1)*. Because of this, someone could use *e* and the knowledge of how to compute the totient of *n* to find d, thus giving them the ability to decrypt messages.

4. [5] Suppose two users, Alice and Bob, have the same RSA modulus *n* and suppose their encryption exponents, *eA* and *eB*, are relatively prime. Charles wants to send the same message, *m*, to both Alice and Bob so he encrypts to get 𝑐𝐴≡𝑚(𝑚𝑜𝑑 𝑛) and 𝑐𝐵≡𝑚𝑒𝐵(𝑚𝑜𝑑 𝑛). Show how Eve can determine *m* if she intercepts *cA* and *cB*. *Hint: since eA and eB are relatively prime, what special integers x and y can we always find?*

Because eA and eB are relatively prime, Eve can use the Euclidean Algorithm to find the multiplicative inverses of both. By doing this, she can raise each ciphertext to its respective multiplicative inverse and will be able to see the plaintext.

5. [5] Show all the steps of key generation, encrypting and decrypting with RSA, when the primes you choose are 17 and 29, the public exponent is 13, and the message is 7. Clearly identify the public key, private key, ciphertext, etc.

*n* = p\*q = 17\*29 = 493.

Φ(n) = (17-1)(29-1) = 448.

*e =* 13.

*d* = *e*-1mod(Φ(n)) = 69.

Public Key (*n,e*) = (493,13).

Private Key (*d, p, q, Φ(n)*) = (69, 17, 29, 448).

Message = 7

Encrypted ciphertext ≡ Me (mod *n*) ≡ 713 (mod 493) ≡ 431.

Decrypted message ≡ Cd (mod *n*) ≡ 43169 (mod 493) ≡ 7.

6. [5] For n odd, what is an efficient procedure for putting n-1 in the form 2sr where r is odd? Use this procedure to express 46 and 32 in this form, then compute 346 (mod 47) and 332 (mod 33) showing your work.

An efficient way to compute this is to divide (n-1) by two until the quotient is odd. By doing this, *s* can be found, and the odd quotient can be used as *r*.

46/2 = 23 🡪 *s* = 1, r = 23.

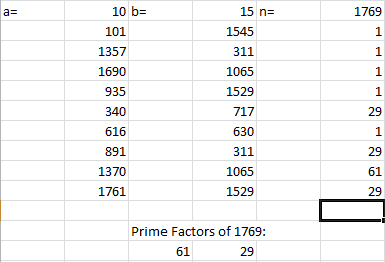
32/2 = 16 🡪 16/2 = 8 🡪8/2 = 4 🡪 4/2 = 2 🡪 2/1 🡪 1 🡪 *s* = 5, *r* = 1.

47-1 = 46 / 2 = 23 🡪 21\*23 = 46 🡪 346 (mod 47) ≡ 1.

33 – 1 = 32 🡪 32 / 2 = 16 🡪 16 / 2 = 8 🡪 8 / 2 = 4 🡪 4 / 2 = 2 🡪 2 / 2 = 1 🡪

25\*1 = 32 🡪 332 (mod 33) ≡ 1.

7. [10] Use Pollard’s Rho Algorithm to factor 1769. Show all the (a,b) pairs that led to a successful factorization, the corresponding values of gcd(|a-b|,n), and provide the final factorization. (Note: Copy/paste from Excel is acceptable to show your work)



Final factorization: 61, 29.

8. [10] (Recommend program or Excel) Use the Miller-Rabin primality test algorithm to check the following numbers *n* for primality: {17, 43}. Show all the steps. Just run the algorithm once for each n, with a witness of your choice. This means that, in the pseudocode below, t would equal 1.

For *n* = 17:

*n =* 17, *r* = 1, *s* = 4, *a =* 4.

*y =* 41 (mod 17) ≡ 4.

*y* doesn’t equal 1 and *y* does not equal (*n*-1).

*j* = 1.

*j* is less than (­*s*-1) and *y* does not equal (*n*-1).

*y* does not equal 1.

*y* does not equal (*n*-1).

(loop back around)

*y =* 42 (mod 17) ≡ 16.

*y* is equal to (n-1), thus *n* is prime.

For *n* = 43:

*n* = 43, *r* = 21, *s* = 1, *a* = 15.

*y* = 152 (mod 43) ≡ 10.

*y* doesn’t equal 1 and *y* does not equal (*n*-1).

*j* = 1.

*j* is not less than (­*s*-1). Thus, *y* is prime.