**UNDERSTANDING RSA ENCRYPTION**

**(cite Eddie Woo videos)**

**Introduction**

[Purpose of encryption in general, and RSA encryption in particular. Key idea: RSA is so hard to break that it’s, for practical purposes, impossible. This is because some functions that are easy to compute have inverses that are very difficult to compute. We will explore this more in the runtime analysis.]

**Mathematical reminders**

Meaning of modulo

Meaning of prime

Fundamental Theorem of Arithmetic

**Encoding (In general, for example)**

The encrypter needs to know e and n.

1. Convert your message to an integer, using some previously agreed upon algorithm. For example, you might start your message with 1 and then map every letter to two digits, depending on its position in the alphabet. So “HELLO” would map to 1 08 05 12 12 15.
2. Create the cyphertext by substituting your numeric message for m in the formula m^e mod n = c. The values e and n are the public key published by the recipient of your message. The value of your cyphertext is c. Anyone can see these numbers, but only the owner of the private key can decrypt a message that is created with them. This is already quite a bit of computation for a person, but not difficult for a computer.
3. Send your message!

**Decoding**

The decrypter needs to know d and n. (But they also know e, because everyone does.)

1. Substitute the received cyphertext for c in the formula c^d mod n = m. Recall that m is the numeric value of the original message.

**Generating d, e, n.**

1. Choose 2 primes, p, q. In practice, these primes would be very large.
2. Let n = pq. This is the modulus for the en/decryption.
3. Calculate phi(n) = (p – 1)(q – 1), but which also equals the integers less than n that do not share any common factors with n. [why?]
4. Now we’ll find e by searching for a number subject to a few conditions:
   1. e must be between 1 and phi(n).
   2. e must not share any factors with n and phi(n). [Will there always be an e for any choice of p, q?)
5. Now the public key is complete.
6. Now we’ll find d by searching for a number that satisfies a few conditions:
   1. (de) mod (phi(n)) = 1. [This condition seems like something you could satisfy if you know n and e, which are public.] [Eddie Wu says there’s another condition that he doesn’t tell us about. It’s enough to make us not choose 5 in his example. What is it?] He chooses 11. [Are there other available choices?]
7. So (11, 14) is the private key.

D is called the secret exponent, or the decryption exponent.

E is called the public exponent, or the encryption exponent, or the exponent.

N is called the modulus.

Example:

Start with the public key (5,14), the text “B”, and the private key (11,14). Then show that it works:

Encrypt: M^e mod n = 2^11 mod 14 = 4.

Decrypt: c^d mod n = 4^11 mod 14 = 2. It works!

1. Choose two primes, p = 2, q = 7.
2. Then n = 14.
3. Then phi(n) = (2 – 1)(7 – 1) = 6. (This is also the number of ints less than 14 that share no factors with 14.)
4. Now, to find e:
   1. It must be between 1 and phi(n) = 6.
   2. It must share no factors with 14 or 6. So the only choice is 5. [What if there were more choices?]
5. Now, to find d:
   1. It must satisfy (de) mod(phi(n)) = 1. 🡪 5d mod 6 = 1. But don’t choose 5. (Is that because e = 5?)
   2. You could choose d = 11. You could also choose 17? Does this work? Yes. So there are actually lots of options for any e. In fact, 11, 17, 23, … all work. Isn’t this a problem? I guess not when numbers are very large?

Why would it help to know the two factors that form n? Somehow they have to help you find d.

De mod (phi(n)) = 1, and… that’s the only condition. If you know the factors of n, then does that help you learn phi(n)?

I still don’t understand why factoring a large prime is the key to solving this puzzle. In the RSA-129 video, R says that “you might multiply two large primes together and post that number as your public key.” But Eddie Woo says that your public key is two numbers, the exponent and the modulus. I guess the modulus is the product of two large primes… okay.