Public Key Cryptography and the RSA System

Suppose your best friend moves to California and you want to communicate via email. If you want to be sure that no one can intercept and read your messages, you'll want to encipher them in some way. If you decide to use a Substitution Cipher, you must first decide on the specific key (i.e. what substitutions you'll make) and somehow you need to send this key to your friend. Of course, you can't encipher the key because then how will your friend decipher it? But then if someone intercepts your key, that person can then read all your future email correspondence whether or not you encipher it!

For a more realistic scenario, consider purchasing things on the internet. When you type your credit card number into a purchase form, it gets sent through cyberspace to the company from whom you're buying the merchandise. This message could be intercepted, which means you certainly want it encrypted so that no one except the intended recipient can read it. On the other hand, you don't want to have to meet in advance with a representative from the company to agree upon an encryption method to use. Further, according to a study done by *PC Data*, Amazon.com had 2.3 million customers in January, 2001. If that company were to use a system like a Substitution Cipher, they'd have to store that many individual keys—one for each customer. There must be a better way!

The main problem is that Substitution Ciphers are *symmetric* (in addition to being easy to crack). In general, this means that anyone who knows how to encipher a message using the system also knows how to decipher a message which was enciphered with the system. This makes it a *private key* system, and the effectiveness of the cipher depends upon the secrecy of the key. Wouldn't it be nice if there were a way to send messages using a system where even if someone else finds out how to *send* a message using your cipher system, they can't *read* your messages?

This is the idea behind **Public Key Cryptography**. A public key cryptographic system actually has two keys—one public and one private. In order to encipher a message, all one needs to know is the public key. But to decipher the message, you need a private key too. Think of it as a box with a combination lock for which you are the only one who knows the combination. Then anyone who wants to communicate with you can put a message in the box and lock it, but you're the only one who can unlock the box to read the message. To communicate with your friend in California, you would each design your own set of keys. You would tell each other your public keys and keep your private keys to yourself. Then to send a message to your friend, you would encipher it using *her* public key. Notice that there's no reason for you to know her private key. This means that you never need to exchange your private keys, and so no one can ever intercept them! Similarly, Amazon.com has one public key that is used by every customer. No one except Amazon knows Amazon's private key, which means no one except Amazon can read the customer credit card numbers. (The encryption is done automatically by your web browser when you buy things online.)
One system for public key cryptography is the RSA system, which was developed by Ronald Rivest, Adi Shamir, and Leonard Adleman in 1977 (hence the name RSA). To understand it, you only need to know the elementary number theory which you have already learned this week. The public key is two integers n and e, and the private key is a third integer d. In practice, these integers need to be very large; typically n will have at least 300 digits.

Of course, n, e, and d need to be chosen in a special way to make the system work. In particular, n will be a product of two distinct primes, and d and e will be integers less than n chosen so that $de \equiv 1 \pmod{\phi(n)}$. The secrecy of the private key relies on the fact that given e and n, it is computationally impossible to compute d because doing so would be the same as factoring n, which is computationally impossible for n very large.

In the RSA system, each user sets up his or her own public and private keys. The public keys are "very" public — they are published in some location so that everyone who wants to can find them. The private keys need to be kept "very" private. If anyone (besides the one user who owns them) has access to them, then that person can read any secret messages sent to the rightful owner of the key. Here is a description of how a user, Alice, would go about setting up her keys. We'll give an example at each step.

Pick primes: The first step for Alice is to pick two prime numbers p and q. In practice, these primes need to be very large — about 150 digits each.

- We'll take $p = 281$ and $q = 167$.

Calculate n and $\phi(n)$: Next, Alice simply sets $n = pq$. Since Alice knows the factorization of n, she can compute $\phi(n) = (p-1)(q-1)$. Notice that in practice, n will have about 300 digits. This means that computing $\phi(n)$ would be very difficult without knowing the factorization of n, and factoring n would also be very difficult.

- In our example, we have $n = (281)(167) = 46,927$ and $\phi(n) = (280)(166) = 46,480$.

Choose e — the encoding exponent: The next step is to pick a value of e at random, making sure that $\gcd(e, \phi(n)) = 1$. Alice does this by first selecting a value for e and then performing the Euclidean Algorithm to calculate $\gcd(e, \phi(n))$. If this gcd is 1, great. Otherwise, Alice simply chooses a new value of e.

- We'll take $e = 39,423$. You can check that $\gcd(39,423,\ 46,480) = 1$ using the Euclidean Algorithm.

Find d — the decoding exponent: Now, Alice needs to find a value of d so that $de \equiv 1 \pmod{\phi(n)}$. She can do this by using the Extended Euclidean Algorithm to find x and y so that $ex + \phi(n)y = 1$, and then setting $d = x$. One finds that $d = 26,767$.

Now Alice does the following:

Publish the public keys: $n = 46,927$ and $e = 39,423$

Keep secret the private key: $d = 26,767$

Let m stand for the plaintext message (converted to a numeric value), s stand for the secret message (converted to a numeric value), f stand for the encoding function and f^{-1} the corresponding decoding function.

Encode: To encode the message m for Alice, Bob simply computes

$$s = f(m) = m^e \pmod{n}.$$

In fact, anyone in the world can do this, since everyone knows Alice's public keys.

Decode: To decode an encoded message s, Alice needs to compute

$$m = f^{-1}(s) = s^d \pmod{n}.$$

Only she can do this, because only she knows the value of d. Incidentally, Alice should also destroy her records of p, q, and $\phi(n)$. These bits of information are no longer needed by her, and if anyone else finds them, they will be able to figure out her (private) decoding exponent d.

We will prove later that the RSA system works, but for now, let's look at an encoding/decoding example. Bob wants to send Alice a message in which the characters can be any letter of the alphabet (upper case) or the space. Specifically, Bob would like to tell Alice "NO WAY" to her invitation to go sky-diving. Bob needs to follow these steps:

Look up Alice's public keys: Bob finds $n = 46,927$, $e = 39,423$.

Break up his message into blocks: Bob wants to break up "NO WAY" into blocks that are as large as possible so that frequency analysis cannot be used to decipher his message. Since the encryption function encrypts numbers, and each block will be assigned a number, Bob needs to know how many different "words" (using the characters A-Z and the space) could be formed using a block of a certain length. There are a total of 27 possible different characters in a message from Bob, and so there are $27 \cdot 27 = 729$ 2-letter "words", $27 \cdot 27 \cdot 27 = 19,683$ 3-letter "words", $27 \cdot 27 \cdot 27 \cdot 27 = 531,441$ 4-letter "words", etc. Since $27^3 = 19,683$ is the largest power of 27 less than the public modulus $n = 46,927$ while $27^4 = 531,441$ is greater, the plaintext should be broken into "trigraphs" (blocks of size 3) for enciphering. If Bob used blocks of size larger than 3, he would run into the problem of having 2 different plaintext blocks correspond to the same enciphered block since the encoding function works modulo n. He breaks up "NO WAY" into the two blocks "NO " and "WAY".

Convert the alphabetic blocks to numbers: Letting "A" be "0", "B" be "1", . . . , "Z" be "25", and the space be "26". He thinks of each block as a number in base 27 and converts it to base 10:

"NO ": $13 \cdot 27^2 + 14 \cdot 27^1 + 26 \cdot 27^0 = 9,881$

"WAY": $22 \cdot 27^2 + 0 \cdot 27^1 + 24 \cdot 27^0 = 16,062$

Encode the numbers: Now, Bob simply encodes each block separately using fast exponentiation to do the computations. To save himself even more time, he uses a fast exponentiation program that can be found at

http://www.nebrwesleyan.edu/people/kpfabe/FastExp.html.

To encode "NO ", he computes $9,881^{39,423} \pmod{46,927} = 9,388$.

To encode "WAY", he computes $16,062^{39,423} \pmod{46,927} = 21,358$.

Convert these encrypted numbers to strings: Since the encryption step can produce numbers between 0 and 46,926, each plaintext block of 3 characters will have to correspond to a ciphertext block of 4 characters to accomodate enciphered numbers in excess of 19,683 and less than 46,927. This means that Bob should convert the enciphered numbers to base 27, being careful to always include 4 digits.

Convert $9,388$:

$$\begin{aligned} 9,388 \div 27 &= 347 \text{ R } 19 \\ 347 \div 27 &= 12 \text{ R } 23 \\ 12 \div 27 &= 0 \text{ R } 12 \end{aligned}$$

So $9,388 = 0 \cdot 27^3 + 12 \cdot 27^2 + 23 \cdot 27^1 + 19 \cdot 27^0$, which corresponds to the string "AMXT".

Convert $21,358$:

$$\begin{aligned} 21,358 \div 27 &= 791 \text{ R } 1 \\ 791 \div 27 &= 29 \text{ R } 8 \\ 29 \div 27 &= 1 \text{ R } 2 \\ 1 \div 27 &= 0 \text{ R } 1 \end{aligned}$$

So $21,358 = 1 \cdot 27^3 + 2 \cdot 27^2 + 8 \cdot 27^1 + 1 \cdot 27^0$, which corresponds to the string "BCIB".

Send the encrypted message: He sends "AMXTBCIB" to Alice.

Now let's look at the decoding step. Alice receives the message "AMXTBCIB" from Bob. In order to read the message, she performs the following steps:

Break the message up into blocks of size 4: Alice breaks "AMXTBCIB" up into "AMXT" and "BCIB".

Convert the alphabetic blocks to numbers: Thinking of the blocks as being base 27 representations, she converts them to base 10:

"AMXT": $0 \cdot 27^3 + 12 \cdot 27^2 + 23 \cdot 27^1 + 19 \cdot 27^0 = 9,388$

"BCIB": $1 \cdot 27^3 + 2 \cdot 27^2 + 8 \cdot 27^1 + 1 \cdot 27^0 = 21,358$

Decode the encrypted numbers: Alice must decode each block separately, by raising it to the dth power and reducing modulo n. She'll use fast exponentiation to do the computations.

To decode "AMXT", she computes $9,388^{26,767} \pmod{46,927} = 9,881$.

To decode "BCIB", she computes $21,358^{26,767} \pmod{46,927} = 16,062$.

Convert the decoded numbers to strings: We first find the base 27 representations of our decoded numbers:

$9,881 = 13 \cdot 27^2 + 14 \cdot 27^1 + 26 \cdot 27^0$ corresponds to "NO ".

$16,062 = 22 \cdot 27^2 + 0 \cdot 27^1 + 24 \cdot 27^0$ corresponds to "WAY".

The message that Bob sent Alice was "NO WAY". Too bad for Alice; she'll have to skydive alone.

Now let's see why the RSA system works — i.e., why, for any message m, we have

$$s^d \equiv m \pmod{n}.$$

Remember that d and e were chosen so that $de \equiv 1 \pmod{\phi(n)}$. This means that $de = \phi(n)k + 1$ for some integer k. Also, $s \equiv m^e \pmod{n}$. Therefore,

$$\begin{aligned} f^{-1}(s) = f^{-1}(f(m)) &= f^{-1}(m^e) \\ &\equiv (m^e)^d \pmod{n} \\ &\equiv m^{de} \pmod{n} \\ &\equiv m^{\phi(n)k+1} \pmod{n} \\ &\equiv m^{\phi(n)k} \times m^1 \pmod{n} \\ &\equiv (m^{\phi(n)})^k \times m \pmod{n} \\ &\equiv (1)^k \times m \pmod{n} \text{ (by Euler's Theorem)}^{**} \\ &\equiv m \pmod{n} \end{aligned}$$

The ** after the note about Euler's Theorem is there because Euler's Theorem really only applies if $\gcd(m, n) = 1$. However, it is in fact true that

$$m^{\phi(n)k+1} \equiv m \pmod{n}$$

for all positive integers m less than n, when n is a product of two distinct primes, as it is for us. You might want to try to prove this. You'll need to use the fact that if $n = pq$, then $\phi(n) = (p-1)(q-1)$.

Exercise: The class will divide into two groups. Each group should elect a coordinator. There are two parts to this exercise.

1. Each group will be given a message to encode using RSA, with n, d and e as in the example above. The cleartext should be broken up into blocks of 3 characters. If the length of the message is not a multiple of 3, pad the message with spaces at the end so that the length is a multiple of 3. The coordinator will give each person blocks to encipher. Once the group members have enciphered their blocks, the coordinator will compose the encrypted message and give it to the other group.

2. Each group will decode the other group's message using RSA, with n, d and e as in the example above. The ciphertext should be broken up into blocks of 4 characters. The coordinator will give each person blocks to decipher. Once the group members have deciphered their blocks, the coordinator will compose the decrypted message and the group will follow the instructions of the message.

This handout was modified from Dr. Judy Walker's notes from her Math 398 course taught in the Spring of 2002 at UNL. Some of the discussion was also taken from the book "In Code" by Sarah Flannery.