**Cryptography**

Roughly speaking, cryptography is the study of creating secret code. In many situations, particularly when one group of people want to communicate sensitive information without others knowing, it makes sense to make the message they are communicating difficult for others to read, but easy for the person receiving the message to read. In fact, you participate in this type of exchange often, and may not know it. Every time you type your credit card number in online to make a purchase, that number gets encoded/encrypted so that others who are reading communication over the internet can't easily obtain your credit card number. In this case, it turns out that the vendor to whom you are sending your number, has some secret information they can use to decode/decrypt the message that everyone else doesn't have.

In these notes, we'll highlight three simple cryptography systems that were used in the past.

**Caesar Cipher/ Shift Cipher**

Julius Caesar would frequently use runners to relay messages, but he didn't always want runners to read those messages. Thus, he used a simple encryption technique: for whatever letter he wanted to write, he instead wrote the letter that occurred three letters later. For example, if he wanted to write an 'A', he wrote a 'D', and if he wanted to write a 'B', he wrote a 'E'.

Of course, when you get to the end of the alphabet, there's a bit of a problem. While 'W' encrypts to 'Z', what does 'X' encrypt to? This problem is easily solved by wrapping around the alphabet and having 'X' encrypt to 'A', 'Y' encrypt to 'B' and 'Z' encrypt to 'C'. You'll notice in this system, it's possible to "undo" the transformation by going backwards three letters, and wrapping around backwards, if necessary.

Mathematically, we would like to view the letters as follows: A = 0, B = 1, C = 2, ..., Z = 25

If these are out internal values, then Caesar's cipher can be mathematically written as follows:

f(x) = (x + 3) % 26, where % is the mod function as defined in most programming languages.

In general, instead of shifting over 3 character, we could really shift over any number of characters. In this case, the number of characters we shift is called the key. Caesar typically used a key of 3, though in much of the literature, the term Caesar cipher is used synonymously with the shift cipher.

In order to code the Caesare cipher, it's important to remember that letters in the computer are stored using their ascii values and that 'A' has a value of 65 and 'a' has a value of 97, with the rest of the upper and lower case letters following in contiguous order. Thus, we must convert the ascii value to the corresponding value in between 0 and 25 before we execute the encryption, and then convert our result back to the corresponding ascii value. This amounts to subtracting and then adding the appropriate offset, which is 'A' for upper case letters and 'a' for lower case letters.

**Affine Cipher**

The secret key for the affine cipher are two integers a and b such that gcd(a,26) = 1, (where gcd stands for greatest common divisor), and both a and b are in between 0 and 25, inclusive. Given this key, we encrypt as follows:

f(x) = (ax + b) % 26

For example, if a = 3, b = 7 were the secret key, then the plaintext L (which is 11) would encrypt to O as follows:

f(11) = (3(11) + 7) % 26

 = 40 % 26

 = 14 (O)

The reason a is restricted is that if gcd(a, 26) > 1, then a valid substitution would not be created. We won't go over the proof of this, but we'll go over a quick example.

If a = 2, and b = 5, then consider encrypting both B(1) and O(14):

f(1) = 2(1) + 5 (mod 26) = 7 (H)

f(14) = 2(14) + 5 (mod 26) = 33 (mod 26) = 7 (H).

But, it's impossible for both B and O to encrypt to H, because then, if we see an H in the ciphertext, we won't know what the corresponding plaintext was.

There are a total of 12x26 = 312 possible keys for the affine cipher. (There are 12 values in between 0 and 25 that have a gcd of 1 with 26.)

**Substitution Cipher**

Both the shift and the affine cipher are specific versions of the substitution cipher. In the substitution cipher, each letter is substituted with another one. As you might imagine, there are many possible keys for the substitution cipher (26! in fact...) so memorizing one key is much more difficult that doing so for the shift or affine cipher. Before computers, people would probably carry around a table explicitly listing each substitution in order to encrypt and decrypt using this cipher.

The key flaw with this method is that it preserves frequency information about the letters in the alphabet. Unfortunately, in English text, not all letters appear the same time. 'E' appears about 12.7% of the time while 'Z' appears about .1 % of the time. If you are always doing one for one letter substitutions, then the ciphertext (secret message), will still contain all the accurate frequencies of the letters. Using this information, one can discard most of the 26! possible keys and make intelligent guesses until the cipher is cracked. This process is certainly difficult, but if a message was important enough, spending this time was well worth it. In giving these types of ciphers to students, they are typically able to solve them doing most of the work by hand in several (under 10) hours.

**Vigenere Cipher**

Although the form of this cipher that is currently in most books isn’t exactly what Vigenere proposed, it shows the essential idea that Vigenere had to thwart frequency analysis.

The idea is as follows:

Pick a keyword, such as “COMPUTER”. Now, to encrypt a message, do the following:

Plaintext: MEETMEATTHESTORE

Keyword : COMPUTERCOMPUTER

Ciphertext: OSQIGXEKVVQHNHVV

Essentially, to encrypt, you line up the plaintext with the keyword written down repeatedly (until you get to the end of the message), and then you add the numeric values of both letters (0 to 25) and take the result mod 26 to obtain the corresponding ciphertext letter. In code, we’d have something like this, if the message were stored all in lowercase:

for (i=0; i<strlen(msg); i++)

 cipher[i] = (char)((msg[i]-‘a’+key[i%strlen(key)]-‘a’)%26+’a’);

The code above assumes that all variables are declared, and msg, cipher and key are all character array in C.

Decrypting is performed by simply subtracting out the values of the keyword from the ciphertext in the appropriate manner.

It is clear that this cipher disrupts frequency analysis because the same letter in the ciphertext (such as the two Vs at the end) can be obtained from two different letters in the plaintext. Similarly, two of the same plaintext letters can map to different ciphertext letters.