<u>Shift Cipher</u>

The first cryptographic scheme most people either learn or discover on their own is the shift cipher. Rumor has it that Julius Caesar would send messages to his commanders via runners in an encrypted format so that the runners wouldn't get any critical information. Since his runners weren't terribly bright, all he did to encrypt his messages is count three letters forward from the letter he wanted to express. For example, "CAT" would encrypt to "FDW", since F comes three letters after C, D comes three letters after A and W comes three letters after T. The message recipient could easily "subtract 3 letters" to recover the original message, since he "knew the trick", so to speak. Finally, if we get to the end of the alphabet, we simply wrap around. Thus, X encrypts to A, Y encrypts to B and Z encrypts to C.

Mathematically, we need to express each letter as an integer in between 0 and 25 and utilize the mod operator to formally describe the Caesar cipher, which has a shift of three. The encryption and decryption functions are as follows:

$E(x) = (x + 3) \bmod 26$
$D(y) = (y - 3) \bmod 26$

In these formulas, x represents the plaintext (normal) letter and y represents the ciphertext (encrypted) letter.

The mod mathematically deals with the wrap-around at the end of the alphabet. This explains why we use the numbers 0 through 25 instead of 1 through 26. The former choice makes this wrap-around issue much easier to express mathematically.

The Caesar Cipher is a specific instance of the Shift Cipher. In the shift cipher, instead of always shifting over 3 letters, one can shift over any number of letters they choose. This number is known as the key for the shift cipher. Thus, the shift cipher has a total of 26 possible keys, of which one (0) would be completely useless.

In general, the functions for encrypting and decrypting using the shift cipher are as follows:

$E_k(x) = (x + k) \bmod 26$
$D_k(y) = (y - k) \bmod 26$

where k, represents the numerical value of the key.

Note: Some books/sources refer to the Caesar cipher as being identical to the shift cipher, so that the term Caesar cipher does not ALWAYS infer a shift key of three.

Cryptanalysis of the Shift Cipher

The easiest way to break the shift cipher is simply to try all 26 shifts until you see a message that makes sense. There are so few keys to try that this can be done by hand relatively quickly. People with good instincts can probably eliminate several choices very quickly. Alternatively, it should be pretty easy to write a computer program that tries all 26 shifts and prints out all 26 possible messages. Here's some Java code to accomplish that task that assumes that cipher stores only lowercase letters:

```
void printAllPossibilities(char cipher[]) {

    int i, j;
    for (i=0; i<26; i++) {

        for (j=0; j<strlen(cipher); j++)
            printf("%c", (cipher[j]-'a'+i)%26 + 'a');
        printf("\n");
    }
}
```

Note About How Characters are Stored in the Computer
In this code, the key difficulty (sorry for the pun!) is that we must convert from characters to integers and back. Given the character cipher[j], we must convert that to its numerical value (in between 0 and 25) by subtracting the ascii value of the character 'a'. Then we need to add to this the shift we are trying, in this case, i. After we use mod to map the value back into the range 0 – 25, we need to then add back the ascii value of 'a' to obtain the new character to print.

Python Note
Unlike C and Java where we are allowed to add and subtract numbers from characters, Python doesn't allow this. Python has two functions to allow conversion from a character to its Ascii value and vice versa:

```
// Returns the Ascii value of character c.
ord(c)

// Returns the character with Ascii value num.
chr(num)
```

So, to convert the code above to Python we do:

```
for i in range(26):
    for j in range(len(cipher)):
        print(chr((ord(cipher[j])-ord('a')+i)%26 + ord('a')),end=" ")
    print()
```

<u>Affine Cipher</u>

A natural progression from the shift cipher, which essentially looks like the equation of a line with slope 1, is to add a variable for the “slope”. Thus, instead of just having a single key for a shift, we have two keys, and b, and encryption is as follows:

$E_{a,b}(x) = (ax + b) \bmod 26$

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

$$\begin{aligned} C &= 3(11) + 7 \pmod{26} \\ &= 40 \pmod{26} \\ &= 14\ (O) \end{aligned}$$

One natural question is what the possible values for a and b could be. With the shift cipher, it was reasonably clear that any integers from 0 – 25 could be a possible key. Let’s consider one potential set of keys. If a = 2, and b = 5, then consider encrypting both B(1) and O(14):

$c = 2(1) + 5 \pmod{26} = 7$ (H)
$c = 2(14) + 5 \pmod{26} = 33 \pmod{26} = 7$ (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.

Thus, the problem is, for some values of a, two different plaintext characters map to the same ciphertext character. Specifically, in the example above, what happened was that 14 = 1 + 13, and when we broke down the expression:

$2(14) + 5 = 2(1 + 13) + 5 = 2(1) + 5 + 2(13) = (2(1) + 5) + 26$

what was happening was that the value of a (2) times 13 equals 26, and this was the getting added to the encrypted value of B, leaving the output unchanged. In fact, if a shares **<u>any common factor</u>** with 26 (or more generally the alphabet size), then we can divide 26 by that number, getting a smaller number, which, when multiplied by a would yield a multiple of 26, causing this problem.

Thus, the valid secret keys 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. There are 12 integers in the range [1, 25] that share no common factors with 26. These are 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 and 25. Thus, there are a total of 12x26 = 312 possible keys for the affine cipher.

More generally, if the affine cipher is for a language with n characters, the total number of keys is $n\phi(n)$, where $\phi(n)$ represents the number of integers in the set {1, 2, 3, …, n-1} that do NOT share a common factor with n.

Calculating the Decryption Key Given an Encryption Key

In our past example of a valid affine encryption function, we have:

$E_{3,7}(x) = (3x + 7) \bmod 26$

Now, though, the question is, how do we decrypt the affine cipher. Unlike the shift cipher where it's clear that all you have to do is subtract the key to recover the plaintext, it's less clear what operation needs to be done to recover the plaintext for the affine cipher.

This problem boils down to finding the inverse function. Let's swap x and y and the solve for y, the standard technique taught in algebra class for finding inverses of functions:

$x \equiv (3y + 7) \bmod 26$
$3y \equiv (x - 7) \bmod 26$

Since this is a mod equation, we are not allowed to divide through by 3. But…we can multiply through by 9:

$9(3y) \equiv 9(x - 7) \bmod 26$

$27y \equiv 9(x - 7) \bmod 26$

When we are dealing with a mod equation, we are allowed to substitute any number or expression with another that is equivalent to it under mod, if the expression is additive or multiplicative. (This can't be done with an exponent.) Since $27 \equiv 1 \pmod{26}$, anywhere we see 27, we can replace it with the number 1:

$1y \equiv 9(x - 7) \bmod 26$
$y \equiv 9x - 63 \bmod 26$

Since $-63 \equiv 15 \bmod 26$, which we can see by adding 3(26) to -63, we get our final decryption function:

$y \equiv 9x + 15 \bmod 26$

Thus, for the affine cipher with **encryption keys** a = 3, b = 7, the corresponding **decryption keys** are a = 9, b = 15.

The one "trick" shown here that wasn't explained was how the value of 9 was discovered. It turns out that 9 is defined as the 3^{-1} mod 26, read as "three inverse mod 26". Using the Extended Euclidean Algorithm (next lecture), one can reliably determine modular inverses. The definition of a modular inverse is as follows:

$b \equiv a^{-1} (mod\ n)$ if and only if $ab \equiv 1\ (mod\ n)$

Cryptanalysis of the Affine Cipher

Since the keyspace is fairly small, one can simply try all 312 decryption keys and examine all the possible plaintexts and find one that makes sense. One can do this by hand, or, perhaps use a computer program that has a dictionary of common words stored, and then searches for substrings of the supposed plaintext, looking for those common words. A vast majority of the incorrect decryptions won't have ANY valid words in them, so a program can automatically remove these from consideration so that fewer options have to be examined by hand.

If one happens to know two matching plaintext-ciphertext characters, then one can do actual cryptanalysis and, in most cases, solve for the key itself via algebra.

Consider the following example:

If through frequency analysis you thought that the plaintext E(4) mapped the ciphertext C(2), and that the plaintext T(19) mapped to the ciphertext F(5), then we could set up the following two equations:

2 = a(4) + b (mod 26)
5 = a(19) + b (mod 26

Subtracting the top equation from the bottom yields

3 = 15a (mod 26)
7(3) = 7(15)a (mod 26)
21 = 105a (mod 26)
21 = a (mod 26)

Notice that here, we had to know that $7 \equiv 15^{-1}(mod\ 26)$. Here is a list of the modular inverses, mod 26:

Value	1	3	5	7	11	17	25
Inverse Mod 26	1	9	21	15	19	23	25