Substitution Ciphers

- A substitution cipher replaces one character in the plaintext with another in the ciphertext, but leaves the message in the same order, or stated another way, it is an unscrambled sequence.

The Caesar Cipher

- We will look at a **simple substitution cipher** or **monoalphabetic cipher**. The **Caesar Cipher** (also described earlier) is an example of such a simple substitution cipher.

- Using this cipher, each plaintext character is replaced by the character three positions to the right modulo 26.

- Thus, the plaintext "**a wilderness of mirrors**" becomes "**d zloghuqhvv ri pluuruv**" (offset = 3).

- The example code provided shows the implementation of this cipher. Such a simple cipher has very little use in securing information (unless used to fool grade school students); it is far too easy to crack.

- It is easy to crack because:

    - There are only 26 possible offsets, all of which can be tested within a very short period of time.

    - The cipher preserves all of the spaces between the words, which makes it even more easy for the code breaker.

- Such a simple cipher has very little use in securing information (unless used to fool or impress grade school children); it is far too easy to crack.

- The example code provided illustrates just how easy it is to break a simple substitution cipher. The program will try all 26 possible offsets to see which one produces readable plaintext.

<u>Randomized Substitution</u>

- The Caesar cipher presented above can be improved by using a scrambled alphabet instead of a simple offset. In addition we can also encode the spaces or eliminate them entirely.

- Now, for example, we can map the random string that contains every letter of the alphabet, including whitespace:

  "**abcdefghijklmnopqrstuvwxyz<space>**";

- Into the following randomized string:

  "**qazwsxedcrfvtgbyhnujm<space>ikolp**"

- The randomized alphabet in effect becomes the encryption key.

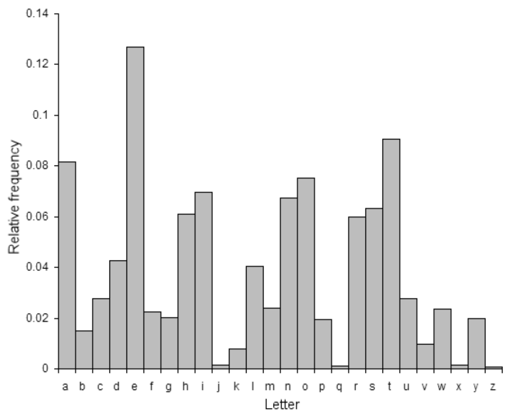
- The example code provided shows the implementation of a random substitution cipher. The program has been modified to eliminate whitespace in the plaintext.

- The main improvement here is the fact that there are now 26! (or 4.0329×10^{26}) possible ways to arrange the alphabet. Thus, attempting to crack this cipher by trying each possible combination will take substantial computing power and time.

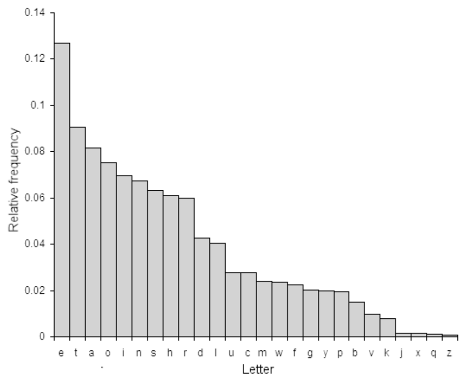
- The easiest way to break this cipher is to exploit the fact that there is a lot of redundancy and other statistical properties of English language that would make it easy to guess at possible keys.

- The first step is to calculate the frequency distribution of the letters in the cipher text. This consists of counting how many times each letter appears.

- Proper English text (current Internet/Social media idioms and vernacular excepted) has a very distinct distribution that can be used in this task. The following chart shows the relative distribution of each character:



- The following chart shows the letter frequencies ordered from most frequent to least frequent:



- As we can see, the letter 'e' is the most common, and appears almost 13% of the time, whereas 'z' appears far less than 1 percent of time.

- Now, substitution ciphers do not change these letter frequencies, they simply scramble the plaintext and change the order of characters.

- All a cryptanalyst need do is to find the key that was used to encrypt the message, which means finding the mapping for each character.

- For reasonably large pieces of text (several hundred characters), it is possible to just replace the most common ciphertext character with 'e', the second most common ciphertext character with 't' and so on.

- This process usually produces result a reasonably good approximation of the original plaintext, but only for pieces of text with statistical properties close to that for English.

- Note that for a reasonably good approximation, a large amount of ciphertext is required. Shorter blocks of ciphertext will require a lot more effort and sophistication to crack.

- Obviously if the original plaintext and/or punctuation from the original plaintext is maintained in the ciphertext, it becomes a lot easier to crack. This is due to the fact that we can apply some of the rules regarding common English language words as follows:

One-Letter Words	a, I.
Frequent Two-Letter Words	of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am
Frequent Three-Letter Words	the, and, for, are, but, not, you, all, any, can, had, her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way, who, boy, did, its, let, put, say, she, too, use
Frequent Four-Letter Words	that, with, have, this, will, your, from, they, know, want, been, good, much, some, time

** the information in the above table was borrowed from Simon Singh's website and book.

- The example code provided shows the implementation of a program that random substitution cipher, given different sets of keys. This is simply a slight variation on the random substitution program.

- The trick of course is to feed it the correct key. Different keys can be generated using the techniques presented above.

- Your assignment will be requiring you to implement a program to generate possible keys using frequency tables.