Substitution Ciphers, continued

3. Polyalphabetic: Use multiple maps from the plaintext alphabet to the ciphertext alphabet.

Non-periodic case:

Running key substitution ciphers use a known text (in a standard book, say). Encipher as for Caesar or Vigenère, except that the key is not periodic or constant. Since the key is as long as the message, this cipher may seem to be unbreakable, like the one-time pad below, but it is not if the key is redundant, as in English text. Roughly speaking, this is so because a large proportion of letters in both key and plaintext will be high frequency letters (`ETAOIN SHRDLU`).

Rotor machines produce running key substitution ciphers with large period d. $d = 26^t$ with t rotors. The Enigma was a rotor machine ($t = 4$) used by the Germans in WW II and broken by Alan Turing (1912–1954) using group theory.

The `UNIX crypt(1)` command is a rotor machine with $t = 1$ and 256 positions.

A *one-time pad* uses a truly random sequence as long as the plaintext as its key.

The *Vernam cipher* is a one-time pad which XOR's the plaintext and key. The key must be sent in advance. Do not use a random number generator that comes with a computer; they are not cryptographically secure. Do not reuse a one-time pad: If the same key is used to encipher M_1 and M_2, then $C_1 \oplus C_2 = M_1 \oplus M_2$, which is easy to break. ($\oplus$ means exclusive-or XOR.) (Compare with the running key ciphers above.)

The one-time pad (Vernam is a good example) has *perfect secrecy*. This means that the conditional probability distribution of the plaintext after the ciphertext is known is the same as the probability distribution of the plaintext before the ciphertext is known. This is a fancy way of saying that the ciphertext tells us nothing about the plaintext (except its length).

The way to prove this statement is to show that given any ciphertext and any plaintext of the same length there is a key (as long as the plain and cipher text) which enciphers the given plaintext into the given ciphertext.

In the case of the Vernam cipher use the formula

$$C = M \oplus K \qquad K = C \oplus M$$

to construct the key from the plain and cipher texts.

4. Polygram: Make arbitrary substitution for groups of characters. Encipher blocks of $d > 1$ letters together as a unit.

An example of a polygram substitution cipher is the *Hill cipher* which codes blocks of n letters into column vectors of dimension n. It enciphers a block of n letters by multiplying it by an $n \times n$ matrix to get a vector of n ciphertext letters. The matrix must be invertible modulo the alphabet size to permit deciphering.

For example, suppose $n = 2$ and we encode the alphabet as A$=0$, B$=1$, etc. Then the plaintext AT would be encoded as $\begin{bmatrix} 0 \\ 19 \end{bmatrix}$ and the plaintext NO would be encoded as $\begin{bmatrix} 13 \\ 14 \end{bmatrix}$. Suppose the key matrix is $K = \begin{bmatrix} 3 & 18 \\ 21 & 11 \end{bmatrix}$.

Then AT would be enciphered as

$$K\begin{bmatrix} 0 \\ 19 \end{bmatrix} = \begin{bmatrix} 3 & 18 \\ 21 & 11 \end{bmatrix}\begin{bmatrix} 0 \\ 19 \end{bmatrix} =$$

$$= \begin{bmatrix} 342 \\ 209 \end{bmatrix} \equiv \begin{bmatrix} 4 \\ 1 \end{bmatrix} \pmod{26},$$

which may be converted back into the letters EB. Similarly, NO would be enciphered as $\begin{bmatrix} 5 \\ 11 \end{bmatrix}$ or FL in letters.

Someone who knew the key matrix K could decrypt ciphertext by multiplying the vectors by K^{-1} (mod 26). The matrix may be inverted by the usual techniques of linear algebra, keeping in mind that any division by d must be done by multiplying by the multiplicative inverse of d modulo 26. We will illustrate a different method by inverting K modulo 26 by Cramer's rule. The determinant of K is $3 \cdot 11 - 18 \cdot 21 \equiv 19 \pmod{26}$. The extended Euclidean algorithm finds that the inverse of 19 mod 26 is 11 since $19 \times 11 - 26 \times 8 = 1$. Then by Cramer's rule,

$$K^{-1} \equiv 11 \begin{bmatrix} 11 & -18 \\ -21 & 3 \end{bmatrix} \equiv \begin{bmatrix} 17 & 10 \\ 3 & 7 \end{bmatrix} \pmod{26}.$$

The ciphertext FL would be deciphered as

$$K^{-1} \begin{bmatrix} 5 \\ 11 \end{bmatrix} = \begin{bmatrix} 17 & 10 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} 5 \\ 11 \end{bmatrix} \equiv \begin{bmatrix} 13 \\ 14 \end{bmatrix} \pmod{26},$$

or NO in letters.

One can break the Hill cipher easily with a known-plaintext attack. If one knows n plaintext-ciphertext blocks, then one can determine K through linear algebra. Suppose we didn't know K, but we did know that $K\begin{bmatrix} 0 \\ 19 \end{bmatrix} \equiv \begin{bmatrix} 4 \\ 1 \end{bmatrix} \pmod{26}$ and $K\begin{bmatrix} 13 \\ 14 \end{bmatrix} \equiv \begin{bmatrix} 5 \\ 11 \end{bmatrix} \pmod{26}$. These matrix equations are equivalent to the single equation

$$K\begin{bmatrix} 0 & 13 \\ 19 & 14 \end{bmatrix} \equiv \begin{bmatrix} 4 & 5 \\ 1 & 11 \end{bmatrix} \pmod{26},$$

which is easy to solve for K using linear algebra.

A ciphertext-only attack is harder. Cryptanalysis based on letter frequency does not work because a Hill cipher encrypts blocks of n letters together. If n is small, one could use the frequency of n-letter blocks to guess n of the blocks and then proceed as in the known-plaintext attack above.

A recent paper makes a great advance in this cryptananlysis.

C. Product Ciphers.

A *product cipher* is a composition of several substitution and transposition ciphers.

$$C = E_K(M) = S_t \circ T_{t-1} \circ \cdots \circ S_2 \circ T_1(M).$$

These two types of ciphers modify the plaintext by **confusion** and **diffusion**, respectively.

Example. DES enciphers 64-bit blocks with a 56-bit key. It has 16 “rounds” or steps. There are $2^{56} \approx 7 \cdot 10^{16}$ DES keys. A million chips, each testing one key per microsecond, can break DES in about one day. Such a machine has been built for less than thirty thousand dollars. The key size of 56 bits is too small. Double encipherment

$$C = DES_{K_1} DES_{K_2}(M)$$

has been proposed. This does not work because of the time-memory trade-off attack.