CSCI3381-Cryptography

Lecture 2: Classical Cryptosystems

September 3, 2014

This describes some cryptographic systems in use before the advent of computers. All of these methods are quite insecure, from the modern standpoint, but they illustrate some important principles.

1 Caesar Cipher (Shift Cipher)

This method was attributed to Julius Caesar (1st century BC) by Suetonius in *Lives of the Twelve Caesars* (2nd century AD). As crude as it is, the basic idea of a shift cipher reappears in the Vigenère Cipher and the one-time pad that we discuss further on, and (with a 26-letter alphabet replaced by a 2-letter alphabet) in modern stream ciphers.

Each letter of the alphabet is identified with a number in the set $\{0, 1, \ldots, 25\}$. So A is 0, B is 1, *etc.* The key is also an integer $k \in \{0, 1, \ldots, 25\}$. The encryption algorithm proceeds letter by letter, replacing each letter j of the plaintext by $j + k \mod 26$. The decryption algorithm is the same, using $-k \mod 26$ in place of k.
Example. The plaintext "ATTACK AT DAWN" is identified with the sequence of numbers

$$0, 19, 19, 0, 2, 10, 0, 19, 3, 0, 22, 13$$

(In this and the next two examples, we eliminate the spacing between words.) The encryption with key $k = 9$ is then

$$9, 2, 2, 9, 11, 19, 9, 2, 12, 9, 5, 22$$

which is transmitted as

$$JCCJLTJCMJFW.$$

The recipient, to decrypt, turns this back into the corresponding numerical sequence and then adds $-9 \bmod 26 = 17$ to decrypt.

In the mathematical language we use to describe cryptosystems, the encryption of individual characters is given by

$$\mathcal{K} = \mathcal{M} = \mathcal{M}' = \{0, \ldots, 25\},$$

$$E(k, m) = (m + k) \bmod 26,$$

and the encryption of a sequence

$$m_1 m_2 \cdots m_r$$

of characters by

$$E(k, m_1 \cdots m_r) = E(k, m_1) \cdots E(k, m_r).$$

Decryption is the same as encryption, with the encryption key replaced by its additive inverse:

$$D_k = E_{-k \bmod 26}.$$

Security. If an attacker suspects that this method was used, the cipher is broken (without a computer!) by a brute-force ciphertext-only attack, since there are only 26 keys, and in all likelihood only of the 26 decryptions will make sense. (What about a known-plaintext attack?)

2 Monoalphabetic Substitution Ciphers

Here the key is a random substitution of one letter by another; for instance 'A' by 'F', 'B' by 'T', *etc.* This is a permutation of the 26 letters: if two letters are different then their encodings in the key also have to be different.

Example. The plaintext is

LAUNCH THE ATTACK AT DAWN ON MONDAY UNDER CLEAR SKIES. HOLD YOUR POSITION IN LIGHT RAIN. RETREAT IN HEAVY RAIN.

Suppose the key is the permutation

RWTPEUSQNLOFIKCVGBHZDXAYMJ of ABCDEFGHIJKLMNOPQRSTUVWXYZ; that is, 'A' will be encrypted by 'R', 'B' by 'W', *etc.*

The ciphertext is then

FRDKTQZQERZZRTORZPRAKCKICKPRMDK
PEBTFERBHONEHQCFPMCDBVCHNZNCKN
KFNSQZBRNKBEZBERZNKQERXMBRNK

The recipient decrypts using the identical algorithm, with the key replaced by the inverse permutation

WROUELQSMZNJYIKDHAGCFPBVXT

Formal description. $\mathcal{K}$ is the set of permutations of $\{0, 1, \ldots, 25\}$, that is, the set of one-to-one functions

$$\pi : \{0, 1, \ldots, 25\} \to \{0, 1, \ldots, 25\}.$$

$\mathcal{M} = \mathcal{M}' = \{0, 1, \ldots, 25\}$. Encryption of individual characters is given by

$$E(\pi, m) = \pi(m),$$

and encryption of sequences of characters is obtained, as above, by concatenating the encryptions of individual characters. Decryption is the same as encryption, with the key π replaced by the inverse permutation π^{-1}.

Observe that the Caesar cipher is just monoalphabetic substitution with a very restricted set of keys.

Security and cryptanalysis The number of keys is the number of permutations of a 26-element set, namely

$$26! \approx 4.03 \times 10^{26}.$$

A brute-force attack is not feasible on a key space this large. But this kind of puzzle appears in the newspapers, and people solve them by hand. (To be fair, the newspaper versions include word boundaries.) How?

There are two serious weaknesses: First, the frequency distribution of letters in English is very uneven, and this same uneven distribution is present in the ciphertext. (Figure 1.)

Thus in a long ciphertext, one might well guess that the most frequently occurring character is the encryption of E.

Second, each character of ciphertext depends on only one character of plaintext and one character of the key. Thus when one part of the key is guessed correctly, it is possible to build on this progress to get other parts of the key. In better systems, changing any piece of the key or the plaintext should completely change the ciphertext.

Our short English plaintext does not exactly reflect these distributions—for instance A (12%), I (9%), N (11%), T (9%) all occur more frequently than E (8%).

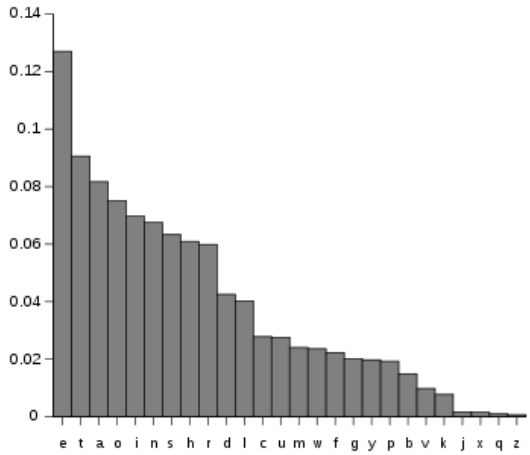

Figure 1: *Relative letter frequencies in English. The six most frequently occurring letters ETAOIN account for about 44% of all letters in a large corpus.*

But the frequency profile is close enough to make some plausible guesses, and build on these to a complete decipherment. Using information about the relative occurrences of 2-, 3- and 4-letter sequences speeds the process. (See the textbook's description, the posted extract from Poe's story 'The Gold Bug', and the first posted project.)

3 Vigenère Cipher (polyalphabetic substitution cipher)

This dates to the 16th century. (The actual inventor was named Bellaso; Vigenère invented a related system, but the misattribution has stuck.) It was widely thought to be essentially unbreakable by commentators writing as late as the early 20th century, but successful attacks were discovered by researchers in the 19th century.

Instead of using a single shift value, as in the Caesar cipher, the Vigenère cipher uses different shift values for different letters of the plaintext. The key is a string typically between 5 and 10 letters long. The shift values are given by the usual integer encodings (0 for A, 1 for B, etc.) of the key characters.

Example. Here the key is

PIGSTY

and the plaintext is

LAUNCH THE ATTACK AT DAWN ON MONDAY UNDER CLEAR SKIES. HOLD YOUR POSITION IN LIGHT RAIN. RETREAT IN HEAVY RAIN. SEND ADVANCE SCOUTING PARTIES TO VERIFY ESCAPE ROUTES.

The table below shows the encryption of the first 15 characters. This is just addition of each column mod 26, using the integer encodings of the characters.

L	A	U	N	C	H	T	H	E	A	T	T	A	C	K
P	I	G	S	T	Y	P	I	G	S	T	Y	P	I	G
A	I	A	F	V	F	I	P	K	S	M	R	P	K	Q

Decryption is done in exactly the same way, subtracting the shift value instead of adding it.

Formal Description Here

$$\mathcal{K} = \mathcal{M} = \mathcal{M}' = \{0, 1, \ldots, 25\}^k,$$

i.e., sequences of k characters, where k is the length of the key. Encryption is given by

$$E(s_1 s_2 \cdots s_k, p_1 p_2 \cdots p_k) = c_1 \cdots c_k$$

with

$$c_i = (p_i + s_i) \bmod 26.$$

Decryption is the same, with $s_1, \ldots, s_k$ replaced by $-s_1 \bmod 26, \ldots, -s_k \bmod 26$.

Security and cryptanalysis. Using several different shift values smooths out the uneven distribution that we get from using a single shift or a single substitution alphabet, so the frequency distribution of letters in the ciphertext is much more uniform. Another advantage over the monoalphabetic substitution was that the short key could be easily memorized.

With a key length of 9, the number of keys is about 5.4×10^{12}. Even with computers, checking every key is a stretch, but can be carried out. In the 19th century environment, of course, a brute-force attack was absolutely out of the question.

There is an effective ciphertext-only attack based on a two-step analysis. The first step determines the key length, and the second determines the key. The idea is this: In the first phase, for $i = 2, 3, \ldots, 10$, compare each ciphertext letter to the letter i characters ahead, and count the number of times these two letters match. The value of i that gives the largest number of matches is probably the key length. Once the key length is established (let's suppose it is 6) we can partition

the ciphertext into 6 subtexts, each of which is the encryption of plaintext using a single Caesar shift value. We can examine all 26 possible decryptions for each of these subtexts and choose the one that most closely resembles the distribution statistics for English.

Here is a precise mathematical description of the attack, along with an explanation of why it works.

We will use the following useful fact: If $(a_1, \ldots, a_n)$ is a sequence of real numbers and $(b_1, \ldots, b_n)$ is a permutation of the same sequence, then

$$\sum_{i=1}^{n} a_i^2 \geq \sum_{i=1}^{n} a_i b_i.$$

For example, look at the sequences $(2, 1, 3, 4)$ and $(4, 3, 2, 1)$. Then

$$4^2 + 3^2 + 2^2 + 1^2 = 30 > 21 = 2 \cdot 4 + 1 \cdot 3 + 3 \cdot 2 + 4 \cdot 1.$$

Let's denote by P_a the probability that a letter chosen at random from English text is an 'a', and similarly P_b, P_c, *etc.* Then the probability that two letters chosen independently at random from English text are the same is

$$P_a^2 + P_b^2 + \cdots + P_z^2.$$

The same will be true if we choose two letters independently at random from English text that has been encrypted by a shift cipher. What if we make the first selection from text that has been encrypted by a shift cipher using one shift amount, and the second from text that has been encrypted with a different shift? In this case the probability that the two letters are the same will be

$$P_a P_{a'} + P_b P_{b'} + \cdots + P_z P_{z'}$$

where $(P_{a'}, P_{b'}, \ldots, P_{z'})$ is some permutation of the original probabilities. Because of our inequality, we should expect a higher value when the same shift value is used for both selections than when different shift values are used. Observe that this would not work at all if the letter frequencies were uniformly distributed, since we would then get roughly the same value for both sums.

To apply this idea, align the ciphertext with itself advanced i characters for $i = 2, 3, \ldots, 9$ characters (as high as the key length is likely to be) and count the number of matches. The table below shows the alignment for the first few characters of ciphertext in our example, with $i = 4, 5$.

A	I	A	F	V	F	I	P	K	S	M	R	P	K	Q
				A	I	A	F	V	F	I	P	K	S	M
					A	I	A	F	V	F	I	P	K	S

Count the number of matches for each alignment (there are three matches for $i = 5$ in this example): By the above analysis, the alignment that gives the largest number of matches should give away the length of the key, although we would need longer samples than the one shown in the diagram above.

In the second step we use the information about the key length (let us suppose it is 6) to divide the text into 6 subtexts: the first subtext consists of the 1st, 7th, 13th, etc. characters, the next consists of the 2nd, 8th, 14th, etc., characters, and so on. Each of these subtexts represents a selection of English characters encrypted with a *single shift*. It remains to find the shift associated with each subtext.

For each subtext we compute the vector

$$(f_a, f_b, \ldots, f_z),$$

giving the number of occurrences of $a, b, c, etc.$. These should be roughly proportional to the probabilities $P_a, P_b, \ldots,$ but shifted: for instance, we would have $(f_a, f_b, \ldots, f_z)$ is approximately a multiple of $(P_y, P_z, P_a, \ldots, P_w, P_x)$ if the shift value was 2. We use an estimate of the probabilities $P_a, P_b, \ldots, P_z$ and compute each of the sums

$$f_a P_a + f_b P_b + \cdots + f_z P_z,$$

$$f_a P_b + f_b P_c + \cdots + f_z P_a,$$

$$\cdots$$

$$f_a P_z + f_b P_a + \cdots + f_z P_y.$$

Again our inequality suggests that the largest of these gives the correct shift for each subtext, and thus reveals the key. An example is worked out in detail in the Python notebook posted on the course website.