Homework 1—Solutions

1. (As usual, we associate the English alphabet with the set \{0, \ldots, 25\}. ) Gen: choose the period \(t\) uniformly from \{1, \ldots, t^*\} (where \(t^*\) is some upper bound on the period); then choose \(k_1, \ldots, k_t\) uniformly and independently from \{0, \ldots, 25\}.

Enc: on input a key \(k_1 \cdot \cdots \cdot k_t\) and a message \(m_1 \cdot m_\ell\), where each \(m_i \in \{0, \ldots, 25\}\), output the ciphertext \(c_1 \cdot \cdots \cdot c_\ell\) where

\[
c_i := \left[ m_i + k_{[i \mod \ell]} \mod 26 \right].
\]

Dec: on input a key \(k_1 \cdot \cdots \cdot k_t\) and a ciphertext \(c_1 \cdot \cdots \cdot c_\ell\), where each \(c_i \in \{0, \ldots, 25\}\), output the message \(m_1 \cdot m_\ell\) where

\[
m_i := \left[ c_i - k_{[i \mod \ell]} \mod 26 \right].
\]

2. In each case, we show how the ciphertext corresponding to a specific, chosen plaintext enables the attacker to learn the key. For the shift cipher, if the encryption of the single-character plaintext \(m\) is \(c\), then the key is \(\left[ c - m \mod 26 \right]\). For the substitution cipher, the key can easily be derived from the encryption of the 25-character string abcdefghijklmnopqrstuvwxyz. For the Vigenère cipher, let \(t^*\) be an upper bound on the period; given the encryption of any \(2t^*\)-character plaintext, the key can be determined.\(^1\)

3. If \(abcd\) is encrypted with the shift cipher, the second character of the ciphertext will always be one more (modulo 26) than the first character of the ciphertext; if \(bedg\) is encrypted, the second character of the ciphertext will always be three more (modulo 26) than the first character of the ciphertext. This makes it easy to tell which of the two messages was encrypted.

4. Period 2 It is not possible to tell which password was encrypted. The reason is because in this case the shift used in the first and third positions is the same, and the shift used in the second and fourth positions is the same, but the shifts used in the first/third and second/fourth positions are independent. And the difference between the first and third characters (resp., the second and fourth characters) of the first password is the same as the difference between the first and third characters (resp., the second and fourth characters) of the second password.

\(^1\)Given the encryption of any \(t^*\)-character plaintext, the key can usually be determined, but there are a few corner cases where it cannot.
Period 3 Here it is possible to tell which password was encrypted, because the shift used in the first and fourth positions is the same. So if abcd is encrypted, the fourth character of the ciphertext will always be three more (modulo 26) than the first character of the ciphertext; if bedg is encrypted, the fourth character of the ciphertext will always be five more (modulo 26) than the first character of the ciphertext.

Period 4 It is not possible to tell which password was encrypted, because using a 4-character key to encrypt a 4-character plaintext is perfectly secret (by analogy with the one-time pad).

5. The basic idea of the attack follows along the lines of what is in the book, with some modifications made because we are working with bytes rather than the English alphabet. There are two steps: finding the period, and then determining the key. These main ideas are described below, and some code is available on Piazza as well.

Finding the period. Let $c_1, c_2, \ldots$ denote the ciphertext, where each $c_i$ is now a byte (i.e., two hex characters, or 8 bits). To find the period, we try candidate periods $\tau = 1, 2, \ldots$. For each candidate period $\tau$, we tabulate the frequencies of the ciphertext stream $c_1, c_1+\tau, c_1+2\tau, \ldots$. Note that now there are not 26 frequencies (one for each English letter), but 256 frequencies (one for each possible byte). So for each candidate period $\tau$, we get 256 values $q_0^\tau, \ldots, q_{255}^\tau$, where $q_i^\tau$ denotes the frequency with which the $i$th byte occurs in the given stream.

We can then calculate $S_\tau = \sum_{i=0}^{255} q_i^\tau$. In contrast to when we were working with the English alphabet, however, here we have no idea what the sum will be for the correct $\tau$. It doesn’t really matter: when $\tau$ is the correct period, $S_\tau$ will be maximized.

Determining the key. Once we know the period, we take a particular stream of ciphertext, call it $c'_1, c'_2, \ldots$, and determine the corresponding character $k$ of the key. To do this, we try all 256 possibilities for $k$. For each such guess, we compute the plaintext stream $c'_1 \oplus k, c'_2 \oplus k, \ldots$ that would result, and tabulate the frequencies of the plaintext characters. We could compute a set of frequencies $q_0, \ldots, q_{255}$ as before, but this wouldn’t be so helpful since we do not know the correct frequencies $p_0, \ldots, p_{255}$. (Actually, though, we do know that $p_i = 0$ for $i > 127$ or $i < 32$, and this can already be used to rule out most choices for $k$.)

Instead of computing the fraction of all characters that are equal to byte $i$ (for all $i \in \{0, \ldots, 255\}$) we can make our lives easier by computing the fraction of lowercase letters that are equal to $i$, for $i \in \{a, \ldots, z\}$. (So, for example, if we have 300 characters of which 200 are lowercase letters and 10 are ‘e,’ we will calculate the fraction of letters that are ‘e’ (i.e., 10/200) rather than the fraction of characters that are ‘e’ (i.e., 10/300).) In this way we get a set of 26 frequencies $q_0, \ldots, q_{25}$ which should correspond to the known letter frequencies $p_0, \ldots, p_{25}$ of standard English text.

Other techniques (e.g., eliminating keys that ever lead to non-printable characters, or that lead to too few lowercase letters) can be used to determine the correct characters of the key; there is some cleverness involved here as well.