CS 198 Codebreaking
Fall 2022 Ryan Cottone, Imran Khaliq-Baporia **Note 1**

Substitution Ciphers

The most basic type of cipher is the *substitution cipher*, where the user translates their message (plaintext) from one alphabet to another alphabet. To review:

Definition 1: Cipher

A cipher is a function that encrypts some message (**plaintext**) into an unreadable **ciphertext** using a **key**. This ciphertext can be decrpyted back into plaintext later by someone who knows the key.

For example:

Plaintext:	aaaa
Key:	1
Ciphertext:	bbbb

Which can be translated to numbers as such (starting at $a = 0$ ...):

Plaintext:	0000
Key:	1
Ciphertext:	1111

As you can see, this cipher shifts all letters over by some number. This cipher is called a **Caesar cipher**, one of the earliest known cryptosystems. We can formally model this as a function E_n:

$$A := \{a, b, \ldots z\}$$
$$E_n : A \to A$$
$$(\forall x \in A)(D_n(E_n(x)) = x)$$

In effect, we define a function E_n that operates on alphabet A, and whose inverse exists for all elements in A. This function is the *encryption function*, and its inverse the *decrypting function* (D_n). For addition-based substitution ciphers like this, the decryption is simply subtracting the key.

Notably, this inverse must exist mod 26, as our alphabet has 26 letters. If we were to start at "z" (26) and add 1, we would "wrap around" to "a" (0). This relationship is modeled as follows:

$$E_n(x) = x + n \pmod{26}$$
$$D_n(x) = x - n \pmod{26}$$

Since this cipher is additive, we do not have to worry about the existence of a multiplicative inverse (however, we will in later topics using modular arithmetic).

Exercise 1.1: Decrypt NV ILHYZ using a shift of 7.

Polyalphabetic Ciphers

So far, we have seen a monoalphabetic cipher (Caesar), which employs a single alphabet. However, there exists ciphers that utilize multiple substitution alphabets.

Definition 2: Repeating key

A **repeating key** is a way of extending a key that is shorter than the plaintext to be the same length as the plaintext. For example, the plaintext "hello" and key "bc" would end up with a repeated key of "bcbcb".

The most famous one, named the *Vigenère cipher*, is a straightforward extension of the Caesar cipher using a repeating key.

Plaintext:	aaaa
Key:	ab
Ciphertext:	bbbb

Using our numbering system and extending the key to fit the full ciphertext:

Plaintext:	0000
Key:	0101
Ciphertext:	0101

Despite the plaintext being all the same letter, the ciphertext is different! This provides a minute amount more security than a Caesar cipher, but as you will see later, both are easily broken with some statistical analysis.

Formally:

$$\begin{aligned} n &= \text{length of the key} \\ E_i(x) &:= x_i + k_i \pmod{26} \\ k_i &= (i \pmod{n})\text{-th digit of the key} \\ x_i &= i\text{-th digit of the plaintext} \end{aligned}$$

Thus,

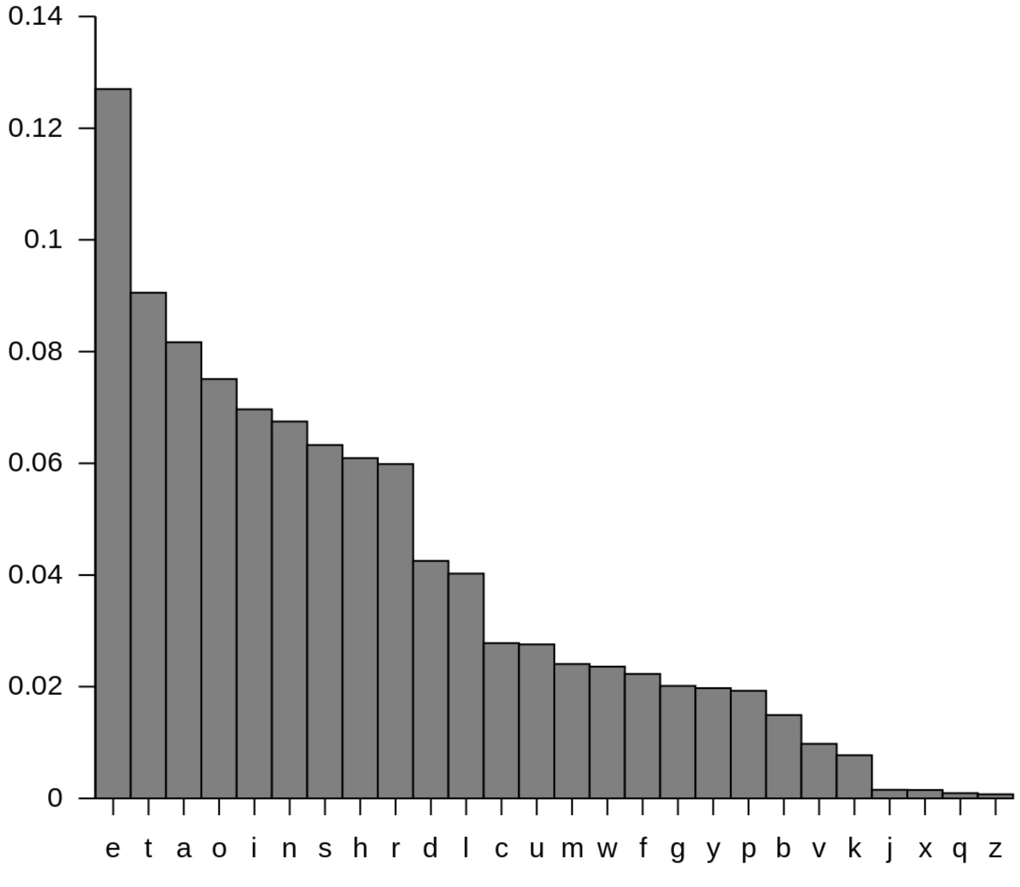
$$E(x) := E_1(x) \dots E_n(x)$$

The decryption function is defined similarly, just subtracting the key instead. These structures work for any arbitrary alphabet and encryption function (presuming that the inverse always exists). You could represent your data in binary, hexadecimal, or any base with no problems. Encrypting and decrypting is as simple as addition and subtraction modulo some number. For this reason, such ciphers dominated the early years of cryptography, as they provided easy and quick security. However, as we will see in the next page, the security is basically nonexistent.

Exercise 1.2: Decrypt "fw fmohb-sai-r oqrbyeqhqrrkg" using key "denero".

Frequency Analysis

Unfortunately for our newly constructed ciphers, there exists a very powerful tool at our disposal to detect what the shift of a monoalphabetic cipher is: *frequency analysis*. Simply put, English (and other languages) use some letters more than others. Over a large set of numbers, the distribution will converge to be something like this:



Using this, we can construct a function to "score" any given blurb of English text by comparing it to this distribution. If the frequencies of the given text are close to the frequencies of this distribution, we give it a high score, meaning it is likely to be English.

However, small texts often have very high variance, meaning they won't necessarily follow the distribution even if it is valid English. For example, "fun" is a valid word but virtually indistinguishable from gibberish using this method since it is so short. The Law of Large Numbers implies that as the length of this hypothetical English text gets longer, the distribution will converge to the one pictured above. Thus, it will work for *most* examples.

Definition 3: Law of Large Numbers

The **Law of Large Numbers**, or LLN for short, states that as the number of experiments approaches infinity, the average of the results will converge to the expected value.

Now, how to implement this in mathematically? There are various methods of frequency analysis, with more advanced ones taking into account subparts of words like "-ing", but for now we will stick to simply comparing the frequencies of each individual letter. For those who have taken some statistics classes (or Data 8), the concept of *Total Variation Distance* may be familiar. TVD is a method of calculating how "different" two distributions are by summing the absolute value of the differences between each corresponding entry.

Definition 4: Total Variation Distance

The **Total Variation Distance**, or TVD for short, is a method for measuring the "distance" between two categorical distributons. The function $f(x,y)$for finding the TVD is as follows for distributions x,y with n categories:

$$f(x,y) = \frac{1}{2}\sum_{i=0}^{n} \mid x_i - y_i \mid$$

Given base frequency values $F_0 ... F_{25}$ and ciphertext frequency values $C_0 ... C_{25}$, where F_0 represents the frequency of A, we wish to find:

$$\min_k \frac{1}{2}\sum_{i=0}^{25} \mid (C_{i+k} - F_i) \mid$$

Informally, we find the offset with the smallest TVD, which translates to the offset giving us the most English-like decrypted text. This offset is our best guess at the key.

Polyalphabetic Frequency Analysis

While this basic form of frequency analysis works well for monoalphabetic ciphers like Caesar's cipher, it does not work well with polyalphabetic ones. This is because the multiple alphabets have different frequency charts, which overlap and eventually form a uniform distribution.

However, **if** we are able to discern the key length (say, k), we can split the cipher into k different monoalphabetic ciphers to crack individually. For example, a Vigenère cipher with key length 2, ciphertext "BCBC" can be split into "BB" and "CC" individual, monoalphabetic ciphers.

Once solved via frequency analysis, we take the offset and add it back into the overall key. For this example, the offsets are 1 and 2, so the key is 12 = "BC".

The trouble lies in determining the key size. For a very long message of length n, attempting to try every different key size results in $O(n^2)$ operations. In the grand scheme of cryptography, this polynomial-time algorithm is actually very tractable. However, we can do better with statistical analysis.

Definition 5: Big-O Notation

Big-O notation is a way of analying the complexity of an algorithm. Informally put, a algorithm with input size n taking $O(n^2)$ time means it will take n^2 operations as $n \to \text{inf}$. "Tractable" means computationally feasible to run in a reasonable amount of time.

The **Kasiski test** utilizes repetitions of characters to identify the most likely key size. For example, take the following encryption:

Plaintext:	Cryptography isnt cryptocurrency
Key:	abcd
Ciphertext:	**Csastp**iuaqjb itpw **csastp**exrsgqcz

Notice the bolded sections – these letters are the same, meaning that it is very likely they were the same plaintext encoded by the *same key*. We can use this to our advantage by recording how far apart these two sections are, to try and find a key length that would "repeat" at exactly that length. In this case, measuring from the ’o’ to the next ’o’ in crypto, we see they are 16 characters apart. 4 divides 16, and thus is far more likely to be the key length than other distances. Of course, so do 2 and 8 – but as your ciphertext gets longer and longer, all these occurences of duplicate substrings will narrow down the choice quite considerably.

Open-Ended Question: Is there a way to modify this cipher to be more resistant to Kasiski tests?

Contributors:

- Ryan Cottone