CMSC 28400 Introduction to Cryptography **Autumn 2021**

Notes #1: Classical Ciphers and Cryptanalysis

Instructor: David Cash

1.1 Syntax of a Cipher

Most cryptographic problems involve studying the construction of an algorithm or algorithms with security goals. For this part of the class, we're studying *ciphers*. A first step is define precisely what type of object we're talking about, which we informally refer to as the "syntax" of the object. Once equipped with a definition, we can then proceed to think about specific instances of that object being secure or not. This process is overkill at this point, but later we'll find that the "devil is in the details" and our precision will become necessary.

The following definition is pretty formal; We'll just state it and then unwrap its intention afterwards.

Definition 1.1. *Let $\mathcal{K}, \mathcal{M}, \mathcal{C}$ be non-empty sets. A function*

$$E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$$

is called a cipher with key-space $\mathcal{K}$, message-space $\mathcal{M}$, and ciphertext-space $\mathcal{C}$ *if for every $K \in \mathcal{K}$, the function $E(K, \cdot)$ is one-to-one.*

For such a cipher, we define

$$E^{-1} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$

by letting $E^{-1}(K, \cdot)$ be the inverse of $E(K, \cdot)$ for each $K \in \mathcal{K}$.[1]

We'll call the members of $\mathcal{K}$ *keys*, members of $\mathcal{M}$ *messages*, and members of $\mathcal{C}$ *ciphertexts*. We require that they be non-empty only to avoid trivialities (a function on an empty set isn't very interesting).

We're formalizing the process of scrambling a message $M \in \mathcal{M}$ with a key $K \in \mathcal{K}$ as function E. The idea is that two people agree on a random $K \in \mathcal{K}$ beforehand, and then later hide messages by computing $C = E(K, M)$ and sending C.

In order for E to be useful for sending hidden messages, we need that someone (who has the same key) can unscramble the message later. We capture that by requiring that the function $E(K, \cdot)$ is always one-to-one, meaning that no two distinct messages $M, M' \in \mathcal{M}$ will ever satisfy $E(K, M) = E(K, M')$. If this were the case, then whoever receives $C = E(K, M)$ couldn't tell if the sender intended to send M or M'.

The sets $\mathcal{K}, \mathcal{M}$, and $\mathcal{C}$ above are entirely abstract. The key-space $\mathcal{K}$ will always be finite, but $\mathcal{M}$ and $\mathcal{C}$ will sometimes be infinite. A typical example for $\mathcal{M}$ is $\{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}^+$, the set of non-empty strings consisting of one or more English letters (here the letters are just formal symbols, not variables). We might throw in some punctuation symbols, or digits, and so on. Later when

[1] To be quite picky, there may be some $C \in \mathcal{C}$ that would never be output by $E(K, \cdot)$; For these elements we place no restriction on $E^{-1}(K, C)$.

we consider modern ciphers, we'll take $\mathcal{M} = \{0,1\}^{128}$, i.e. bitstrings of length 128. The set of ciphertexts $\mathcal{C}$ might coincide with $\mathcal{M}$ or differ; For instance, we might prefer to use Greek characters or digits for the output.

1.1.1 A First Example: The Shift Cipher

We start at the beginning, with the *shift cipher*. Here are below let us write $\Sigma = \{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}$. For $M \in \Sigma^+$ we'll use the notation M_i to refer to the i-th letter of M.

Definition 1.2 (Shift cipher)**.** *Let $\mathcal{M} = \mathcal{C} = \Sigma^+$ and $\mathcal{K} = \{0, 1, \ldots, 25\}$. For an integer K and a letter $x \in \Sigma$, define $K + x$ to be the letter reached by shifting x forward by K spots, wrapping around the end of the alphabet if necessary; if K is negative then we shift backward by $|K|$ spots, wrapping as needed.*[2]

The shift cipher *is the function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ defined by having $E(K, M)$ output*

$$C = (K + M_1, K + M_2, \ldots, K + M_\ell).$$

That is, $E(K, M)$ shifts each letter of M forward by K positions.

Example 1.1. *For $K = 5$ and $M =$* `CRYPTO`*, $E(K, M) =$* `HWDUYT`*.*

Exercise 1.1. *Implement the shift cipher in Python3, and use your implementation to decrypt the ciphertext*

`J PXXM MRBPDRBN BQXDUM LXWLNJU XWNB QNRPQC`

(ignoring spaces), which was produced using $K = 9$.

In order for this to be a cipher, we need to check that it is one-to-one. We can actually do this by pointing out that the inverse of $E(K, \cdot)$ is the function $E(-K, \cdot)$, i.e. the function that shifts left by K positions, wrapping around the front of the alphabet as necessary.

1.2 Security of Ciphers: Exhaustive Key Search Attacks

Is the shift cipher secure? What does it mean for a cipher to *be secure*? This question will consume a large part of efforts this quarter. For now we think informally: Suppose an adversary has captured a ciphertext C; Now what? Countless adversaries have found themselves in this position. Typically, they may or may not know which cipher E was used, or they may know something about the type of message the sender might send (for example, if message is likely a military order of a personal note). Adversaries today have computers, and perhaps even sophisticated mathematicians and computer scientists to help.

How we should think adversary capabilities in general is something we'll return to later. For now, let's consider the following adversarial strategy, which we'll call *exhaustive key search*:

1. Assume a ciphertext C was produced using the shift cipher.
2. For each $K' \in \mathcal{K}$, run $M = E(K', M)$; If M "looks like a typical message" then halt and output M. Otherwise, continue.

[2] So for example $3 + \mathtt{A} = \mathtt{D}$, $4 + \mathtt{X} = \mathtt{B}$ and $-5 + \mathtt{G} = \mathtt{B}$.

This strategy just tries all of the keys and hopes to recognize when it hits on the correct one. If the parties are sending English text, then in all likelihood only one key will result in a message that looks like English text. Moreover, there are only $|\mathcal{K}| = 26$ keys, so this will run very very quickly, even by hand. It's enough to convince me that the shift cipher is broken.

Exercise 1.2. *Use your shift cipher implementation to mount an exhaustive key search attack against the ciphertext*

`RZEFZXDL, TEBOB CRK DLBP QL ABZLAB.`

The punctuation was not encrypted; it's there just for decoration.

1.3 Substitution Ciphers

A fatal flaw of the shift cipher is that there are only 26 keys. Our next cipher was historically designed to address this issue. We recall that a *permutation* is a function with the same domain and range that is one-to-one and onto.

Definition 1.3. *Let* $\mathcal{M} = \mathcal{C} = \Sigma^+$. *Let*

$$\mathcal{K} = \{\pi : \Sigma \to \Sigma \ : \ \pi \text{ a permutation}\}$$

be the set of all permutations on Σ.

The substitution cipher *is the function* $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ *defined by having* $E(\pi, M)$ *output*

$$C = (\pi(M_1), \pi(M_2), \ldots, \pi(M_\ell)).$$

That is, $E(\pi, M)$ *applies its key* π *to each letter of* M.

Note that we've started writing π instead of K for a key, but this only a notational change.

Representing an element $\pi \in \mathcal{K}$ on a computer is easy; We can write out a string like

`QBFMUOJLTZYDSXEVCGKWANRIPH`

to represent the permutation that takes $\pi(\mathtt{A}) = \mathtt{Q}$, $\pi(\mathtt{B}) = \mathtt{B}$, etc.

The set $\mathcal{K}$ is now huge, consisting of $26 > 2^{88}$ elements (we'll come back to thinking about huge numbers later; for now you think of this number as something like the number of hydrogen atoms in the sun, i.e. mindbogglingly large but finite). An exhaustive key search attack is thus impractical.

1.3.1 Breaking Substitution Ciphers: Frequency Analysis

Exhaustive key search is not the only attack someone may mount against a cipher. Our next attack, called *frequency analysis*, leverages the observation that each occurence of the same letter in the message to the same letter in the ciphertext. For instance, from the ciphertext `TRRT` we know that the message had the same first and last letters, and that the middle two letters were also the same.

This observation becomes very powerful when combined with the fact that letter frequencies are remarkably consistent across different, sufficiently large pieces of English text. In Figure **??**, I have plotted the percentage of all letters that were `A`, `B`, etc. (In more detail, I ignored case and all formatting, and included the title and copyright information contained in the particular version I downloaded from `gutenburg.org`.)

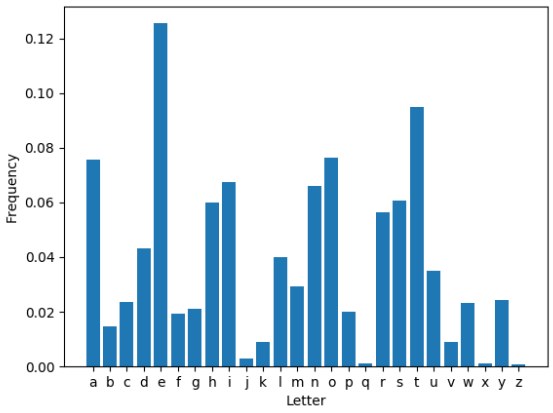

Figure 1.1: Letter frequencies from *The Secret Adversary* by Agatha Christie.

This suggests the following attack strategy against the substitution cipher: Look for the most common letter in the ciphertext, and assume that it probably represents `E`. Then look for the second most common letter, and assume it is `T` or perhaps `A`. As you fill in letters, words should start to emerge if the guesses are correct. If the guesses are incorrect, then odd-looking coincidences will typically occur, indicating that some backtracking is necessary (for example, `E` is not *always* the most common letter).

See the Lecture 2 slides for an example of this process.

1.4 Polyalphabetic Substitution: The Vigenère Cipher

The next cipher, called the *Vigenère cipher* after its popularizer, successfully defended against frequency analysis for hundreds of years. The following description is informal, and I leave it to you to write a formal definition like the above ciphers. A key for the cipher consists of some finite number of "shifts" $K_1, K_2, \ldots, K_n$, each in $\{0, 1, \ldots, 25\}$. To encrypt a string of letters, the cipher applies shift K_1 to the first letter, then K_2 to the second, and so on up to applying K_n to the n^{th} letter. For the next letter, the first shift K_1 is reusued, and so on.

This cipher was particularly popular because it was both hard to break and easy to use. Instead of numbers, one can represent the shifts using letters, and one can even pick the shifts as a word. For instance, the key $K =$ `MAROON` corresponds to shifts of $12, 0, 17, 14, 14, 13$, but is easily remembered.

This type of cipher is called *polyalphabetic* because it is using "multiple alphabets" in a systematic fashion. For even moderately large n, exhaustive key search becomes impractically slow because there are 26^n keys. Naive frequency analysis also fails, as the frequencies get averaged over the different shifts.

1.4.1 Breaking the Polyalphabetic Ciphers

The cipher is still easily broken by modern standards, however. There are two steps: Finding the key length n, and then finding the shifts. One method for finding the key length is to simply try them all, starting from $n = 1$, divide the letters into buckets modulo n, and examine the frequencies

within each bucket. If you have the correct n and enough text, then diagramming the frequencies for each bucket should look like Figure **??**, but with the columns cyclically permuted.

Another trick that works surprisingly well is called *Kasiski Examination*. here one looks for chunks of text that repeat will typically be spaced by a multiple of n apart. For example, in the ciphertext

CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG,

the occurrences of TYC are spaced 6 and 12 apart; This suggests that n divides 6, so probably $n \in \{1, 2, 3, 6\}$.

A similar method tries moving the entire ciphertext forward by one spot, then two, then three, etc, counting the number of characters that match between the modified string and the original. Usually when the key length is reached, there will be an unusually large number of matches. For example, after moving everything forward one spot, we find that two characters match:

```
CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
 CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
```

Shifts $2, 3, 4, 5$ similarly produce a few matches, but a shift of 6 yields nine matching characters:

```
CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
      CTYCGSTTYCVOPRQBTBATYCLOURAPGBGIAPGQCEAPGG
```

One would tend to guess that 6 is the key length.

Once the key-length has been recovered, the rest of the attack is simple and mostly left to you. The crucial fact is that for each bucket there are only a small number of possible shifts, and one of them should cycle the frequencies around to closely match the frequencies typical English.

1.5 Homophonic Substitution

Another method for defeating frequency analysis is to allow multiple possible representations of the same message letter. I'll again describe an example of this cipher informally. The example cipher takes $\mathcal{M} = \Sigma^+$ as before, but it lets $\mathcal{C} = (\Sigma \cup \Pi)^+$, where $\Pi = \{0, 1, \ldots, 9\}$ are digits. A typical key will look the following table:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	A	6	V	3	7	K	E	W	1	2	8	F	Z	J	T	P	5	U	Q	M	H	N	9	4	G
C				B			Y	S					I	X			R	L	O						
				O																					

Now encryption is defined to map each letter to *one of* the letters listed below it. For A there are two choices: D and C, while C will always map to 6. When multiple options are available, encryption can pick arbitrarily. For instance, it can rotate amongst the choices, or even pick a choice at random each time.[3]

The output options may be much larger than 36. For instance, in the Great Cipher, each letter was mapped to a three digit number.

[3]Strictly speaking, if we allow E to pick options at random, then it is no longer a function; It's a sort of "randomized function", where the output $E(K, M)$ is a random variable.

1.5.1 Breaking the Homophonic Ciphers

A properly designed homomophonic cipher can make frequency analysis perform very poorly, since it evens out the distribution of most of the letters. However, with a large enough ciphertext some patterns emerge. Once the output alphabet has been determined, one useful technique is to analyze the frequency of *pairs* or *triples* (usually called bigrams and trigrams) of consecutive outputs. In a large text, commmon bigrams and trigrams will eventually emerge as (less common) bigrams and trigrams in the ciphertext. By starting with the most frequent bigrams and trigrams, one can start guessing substitutions and checking if a consistent text emerges.

1.6 The One-Time Pad Cipher

This last cipher gets a new name, but it is just the Vigenére cipher with a very large key; In fact, as large as the message that is being sent. A bit more formally, this cipher will a message space $\mathcal{M} = \Sigma^\ell$ for some fixed length ℓ. It will also take $\mathcal{K} = \mathcal{C} = \Sigma^\ell$ as well. The actual encryption is defined exactly like Vigenère (where we take the letters of the key to represent shifts).

1.6.1 Breaking the One-Time Pad Cipher

Breaking the one-time pad is quite an interesting question. We'll return to this later, but it turns out to be impossible in some circumstances! One circumstance where it *is* possible is when two messages are encrypted with the same key. The key observation is that one can get some information about the underlying messages in this case, and sometimes enough to recover the entire messages.

Suppose that $C = E(K, M)$ and $C' = E(K, M')$ with $M \neq M'$ are both encrypted with the one-time pad cipher with same key K. Use subscripts like M_i to indicate the i^{th} letter of a string, and let us treat letters interchangeably with numbers modulo 26. Then

$$C_i - C'_i = (M_i + K_i) - (M'_i + K_i) = M_i - M'_i \mod 26,$$

so one can learn $M_i - M'_i \mod 26$ from just the ciphertexts. With some experimentation, it sometimes possible to run frequency analysis on these differences, which will follow a general trend of the language. Another approach, called *crib-dragging*, is also commonly used and can be found in other course material to be posted.