4 Information Theory

Definition 4.1. A (symmetric) encryption system is composed of:

- A set of possible messages $\mathcal{M}$
- A set of possible keys $\mathcal{K}$
- A set of possible ciphertexts $\mathcal{C}$
- An *encryption function* $e : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- A *decryption function* $d : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

such that the decryption function is a partial inverse to the encryption function: that is

$$d(k, e(k, m)) = m \qquad\qquad e(k, d(k, c)) = c.$$

We often write $e_k(m) = e(k, m)$ and $d_k(c) = d(k, c)$. Thus for each $k \in \mathcal{K}$, $d_k = e_k^{-1}$. This implies that each e_k is one-to-one.

Of course, some encryption systems are terrible. To be good, we'd like our cryptosystem to have the following properties:

1. Given any $k \in \mathcal{K}, m \in \mathcal{M}$, it's easy to compute $e(k, m)$.
2. Given any $k \in \mathcal{K}, c \in \mathcal{C}$, it's easy to compute $d(k, c)$.
3. Given a set of ciphertexts $c_i \in \mathcal{C}$, it's difficult to compute $d_k(c_i)$ without knowing k.
4. Given a collection of pairs (m_i, c_i), it's difficult to decrypt a ciphertext whose plaintext is not already known. ("known-plaintext attack").

The first two principles make a cryptosystem practically usable; the third and fourth make it secure. The fourth property is by far the most difficult to achieve. You'll notice that all of the cryptosystems we've studied so far satisfy the first two properties, and several of them do at least okay on the third, none of them achieve the fourth at all.

(Recall that we didn't do an unknown-plaintext attack on the Hill cipher, but we did implement a known-plaintext attack).

These principles are particularly important because they provide security even if your adversary knows the cryptosystem you're using, but not the key. This insight is often called *Kerckhoffs's Principle*, after the nineteenth-century Dutch military cryptographer Auguste

Kerckhoffs, who wrote that a cryptosystem "should not require secrecy, and it should not be a problem if it falls into enemy hands." This was later reformulated by Claude Shannon as *Shannon's maxim*: "The enemy knows the system."

This principle is practically important because, while it's difficult to come up with an entirely new cryptosystem; it's relatively easy to generate a new key. We can change keys on a regular basis, and abandon the use of old keys that have been compromised; it's much more difficult to do the same thing with an entire cryptosystem. Similarly, keys are much smaller than entire systems, so it's easier to communicate them and keep them secret.

So what are the requirements for a cryptosystem to be secure in this manner? When is the key enough to provide security?

4.1 Perfect Secrecy

Definition 4.2. A cryptosystem has *perfect secrecy* if knowing the ciphertext conveys no information about the plaintext, even given knowledge of the cryptosystem.

Mathematically, if $\mathcal{M}$ is the set of possible messages, and $\mathcal{C}$ is the set of possible ciphertexts, a system has perfect secrecy if $P(m|c) = P(m)$ for all $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

Recall that Bayes's theorem says $P(a|b)P(b) = P(b|a)P(a)$. Thus a message has perfect secrecy if and only if $P(c|m) = P(c)$ for all $m \in \mathcal{M}, c \in \mathcal{C}$. Thus we can also say a cryptosystem has perfect secrecy if knowing the *message* gives no information about the *ciphertext*.

Example 4.3. Suppose we have a cryptosystem with two keys k_1, k_2; three messages m_1, m_2, m_3; and three ciphertexts c_1, c_2, c_3. Assume that $P(m_1) = P(m_2) = 1/4$ and $P(m_3) = 1/2$. Further, suppose we have an encryption function given by the following table:

	m_1	m_2	m_3
k_1	c_2	c_1	c_3
k_2	c_1	c_3	c_2

Let's assume the keys are used with equal probability. Then we can compute the probability that the ciphertext is c_2:

$$P(k_1)P(m_1) + P(k_2)P(m_3) = \frac{1}{2}\cdot\frac{1}{4} + \frac{1}{2}\cdot\frac{1}{2} = \frac{3}{8}.$$

However, this system does not have perfect secrecy, since $P(c_1|m_3) = 0$, or alternatively, since $P(m_2|c_2) = 0$.

Because the encryption function is one-to-one, we know that $\#\mathcal{C} \geq \#\mathcal{M}$ for any encryption system. (Every system we've studied has had $\#\mathcal{C} = \#\mathcal{M}$, since the plaintext is a string of letters, and the ciphertext is an equal-length string of letters. But it's easy enough to *not* do this).

A system with perfect system has some other constraints.

Proposition 4.4. *If a cryptosystem has perfect secrecy, then $\#\mathcal{K} \geq \#\mathcal{M}$.*

Proof. Fix some specific ciphertext $c \in \mathcal{C}$ with $P(c) > 0$. Recall that perfect secrecy means that $P(c|m) = P(c)$ for every $m \in \mathcal{M}$, so this means that $P(c|m) > 0$ for every $m \in \mathcal{M}$. (In other words, if the ciphertext gives no information about the message, this means that any possible ciphertext has to be possibly linked to any possible message).

Thus there is at least one key k such that $e(k, m) = c$. Further, by injectivity, this key has to be different for each message: if $e(k, m_1) = c$ and $e(k, m_2) = c$ then $m_1 = m_2$. Thus there is at least one key for each message, and thus $\#\mathcal{K} \geq \#\mathcal{M}$. □

Example 4.5. The Caesar cipher does not have perfect secrecy for messages of more than one letter, since there are only 26 possible keys, and more than 26 possible messages.

The Vigenère and autokey ciphers do not have perfect secrecy for messages longer than the keylength.

We've shown that for a cryptosystem with perfect secrecy $\#\mathcal{C} \geq \#\mathcal{M}$ and now $\#\mathcal{K} \geq \#\mathcal{M}$. The most convenient possible world is when all three of these things are the same.

Theorem 4.6 (Shannon)**.** *Suppose a cryptosystem satisfies $\#\mathcal{K} = \#\mathcal{M} = \#\mathcal{C}$. Then the system has perfect secrecy if and only if:*

1. *Each key $k \in \mathcal{K}$ is used with equal probability; and*
2. *For each $m \in \mathcal{M}$ and $c \in \mathcal{C}$ there is exactly one $k \in KK$ with $e(k, m) = c$.*

Proof. Suppose the cryptosystem has perfect secrecy. Let $S_{m,c} = \{k \in \mathcal{K} : e_k(m) = c\}$. To prove (2) we just need to show that $S_{m,c}$ contains exactly one element for each m and c.

By injectivity, we know that $S_{m_1,c} \cap S_{m_2,c} = \varnothing$ if $m_1 \neq m_2$—otherwise there would be some key that encrypts both m_1 and m_2 to the same ciphertext c.

Further, $S_{m,c}$ is non-empty for every pair m, c. Since the cryptosystem has perfect security, knowing the ciphertext gives no information about the plaintext—so knowing the ciphertext is c can't rule out the fact that the plaintext is m. Thus every ciphertext must be reachable by every plaintext; so for each pair m, c, there is a key k such that $e_k(m) = c$.

Now fix some specific ciphertext $c \in \mathcal{C}$. We know that for each m there is at least one $k \in S_{m,c}$. But $\#\mathcal{M} = \#\mathcal{K}$ by hypothesis, so there must be exactly one k for each m. Thus $S_{m,c}$ has exactly one element for each m.

Now we just want to prove (1), that each key is used with equal probability. But for any k, c we can pick take $m = d_k(c)$ and then because our encryption system has perfect secrecy, we can compute :

$$P(m) = P(m|c) = \frac{P(m,c)}{P(c)} = \frac{P(m,k)}{P(c)} = \frac{P(m)P(k)}{P(c)}$$

and canceling gives $P(k) = P(c)$. Since this is true for every k and every c, we must have $P(k)$ and $P(c)$ both constant; and in fact, $P(k) = P(c) = \frac{1}{\#\mathcal{C}}$.

Conversely, suppose our cryptosystem satisfies these conditions. Then given any $m \in \mathcal{M}, c \in \mathcal{C}$, there is exactly one $k \in \mathcal{K}$ with $e_k(m) = c$.

Fix some ciphertext c. Then each plaintext corresponds to exactly one key; and each key has the same probability; so each plaintext occurs with exactly the same probability. But this is the definition of perfect secrecy. □

Example 4.7 (The one-time pad)**.** A one-time pad is a cryptosystem of the following form:

The message is a string of N letters. The key is a randomly generated string of N letters. The ciphertext is obtained by adding each letter in the plaintext to the corresponding letter of the key (mod 26). This is essentially a Vigenère cipher, with a key length equal to the message length.

It's clear that the one-time pad satisfied property (2) of theorem 4.6. As long as the keys are generated uniformly at random, it also satisfies property (1), and this cryptosystem has perfect secrecy.

Thus a properly-implemented one-time pad is mathematically perfectly secure. However, it is rarely used becuase it is quite cumbersome, and we have many much less cumbersome systems that are "good enough".

Further, the proper implementation can be difficult; if your process for generating the key is *not* perfectly uniformly random, then the cryptosystem is not perfectly secure, and it is possible to break it with enough information.

4.2 Entropy

Suppose a cipher doens't have perfect secrecy. How much information do we actually need to break it? Well, first we need to specify what we mean by "information".

Definition 4.8. Let X be a random variable that takes on finitely many possible values $x_1, \dots, x_n$ with probabilities $p_1, \dots, p_n$. Then the *entropy* of X is given by

$$H(X) = H(p_1, \dots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

(adopting the convention that if $p = 0$ then $p \log_2 p = 0$).

Proposition 4.9 (Shannon)**.** *1. H is continuous in each variable.*

2. *If X_n is a random variable uniformly distributed over n possibilities, then $H(X_n)$ is monotonically increasing as a function of n.*

3. *If X can be broken down into consecutive subchoices, then $H(X)$ is a weighted sum of H for the successive choices.*

Further, any function with these three properties is a constant multiple of H.

Entropy measures the amount of information we get from a choice or evaluation of a random variable.

Example 4.10. Supose X is a "random" variable that returns x_1 with probability 1. Then

$$H(X) = -1 \log_2(1) = 0$$

because seeing the actual outcome gives no additional information over knowing the distribution.

Example 4.11. Supose X is a uniform distribution over a set of size n. Then

$$H(X) = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = \sum_{i=1}^{n} \frac{\log_2(n)}{n} = \log_2(n).$$

This makes sense—choosing uniformly from n things gives you about $\log_2(n)$ bits of information.

In particular, if X is a uniform distribution over the English alphabet, the entropy is $\log_2(26) \approx 4.7$.

Fact 4.12. *Let X be a random variable with n possible outcomes. Then*

1. $H(X) \leq \log_2(n)$.

2. $H(X) = \log_2(n)$ *if and only if the distribution is uniform.*

Thus entropy is maximized when the choice is maximally uncertain.

We said the entropy of a uniform distribution over the English alphabet is about 4.7 bits. But in actual English, letters aren't chosen at random! We can only really find the entropy of written English experimentally, by testing large bodies of English.

If we simply look at the probability distribution over letters from a frequency chart, we get $H \approx 4.132$ per letter. The fact that this number is less than 4.7 reflects the fact that not all letters are equally common.

However, English also doesn't consist of random sequences of letters. Some bigrams are more common that others. Taking bigram frequencies into account gives an entropy estimate of approximately 3.56 per letter. Of course then we need to consider trigrams, and quatragrams, and pentagrams, and in fact the entire infinite sequence; we experimentally estimate that English has an entropy of about 1.5 bits per letter.

Thus written English is highly redundant: about 70% redundant. We can also say that English has a *redundancy* of about 3.2 bits per letter. This doesn't mean we can randomly remove 70% of letters and still expect a readable message; it does mean that with a clever algorithm, we can *compress* a message to 30% of its original bit count.

This sounds wasteful, but is a really useful property of language: if we needed to track the difference between xkkyrosl and xkkyorsl carefully, reading would be quite difficult.

This also explains a semi-famous meme:

"Aoccdrnig to rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe."

(The claim as stated isn't quite true; it's possible to scramble words enough to make reading difficult, especially if you move the first and last letters. But this does show the redundancy in written English.)

4.3 Unicity distance

So what does entropy tell us about the practical ability to decrypt a message? How can we use this redundancy to understand the security of a cryptosystem?

We can view a cryptosystem as using the key to *remove* information from the plaintext to give us the ciphertext. But the key can only remove as much information as it contains; if the key doesn't remove all the redundancy, there's enough information in principle to recover the message.

Definition 4.13. The *unicity distance* for a given language and cipher is the length of an original ciphertext necessary to, on average, have enough information to break the cipher.

Example 4.14. CWU as a Caesar cipher can be "GAY" or "VPN". Without more context there's no way of telling. In fact the Caesar cipher has unicity distance of 2; most messages over 2 characters are breakable, so this is an exception.

Example 4.15. Suppose ABCDE is the ciphertext from a simple substitution cipher. It can be any word with no repeated letters; it could be "water" or "slope" or "maths" or a number of other things.

Example 4.16. Suppose the ciphertext is still "ABCDE", but this time we think the text was enciphered with a Vigenère cipher with a keyword of 5 letters. In this case the plaintext can be literally anything.

Recall that these are average minimums. You don't want to promise anyone you can break a simple substitution cipher with thirty characters of ciphertext.

How get these numbers?

Proposition 4.17. *The unicity distance of a language and encryption scheme is the number of bits in the key devided by the redundancy of the language.*

Example 4.18. A Caesar cipher has 25 possible keys, which is 4.64 bits. $4.64/3.2 \approx 1.45$, so you need at least 1.45 characters to decrypt a message enciphered with a Caesar cipher.

A simple substitution cipher has 26! possible keys, which is about 2^{88}. Thus there are 88 bits worth of keys. $88/3.2 \approx 27.5$ so you need at least 28 letters to decrypt a message enciphered with a simple substitution cipher.

A Vigenere cipher with an N-letter keyword has 26^N possible keys, for $\log_2(26^N) = N\log_2(26) \approx N \cdot 4.7$ bits. Thus the unicity distance is $N \cdot 4.7/3.2 \approx N \cdot 1.47$.

Example 4.19. Earlier we mentioned one-time pads. By definition, a one-time pad has a key length equal to the message length. Suppose we have an N-letter message. Then like the Vigenère cipher there are 26^N possible keys, worth $\approx 4.7N$ bits. The unicity distance is $4.7N/3.2 \approx 1.47N$ letters.

But since the message is of length $N < 1.47N$, it is below the unicity distance, and we can't decrypt it. This is exactly what you'd expect, since a one-time pad has perfect secrecy and thus can't be decrypted at all.

We can think of a one-time pad as putting n bits of information in the key, and then using that key to transmit n bits of information. The key can completely conceal all the

information in the message, since the key contains as much information as the message. But this means the key isn't any easier to communicate than the message itself is.