CHAPTER 9

Information-Theoretic Security

Chapter Goals

- To introduce the concept of perfect secrecy.
- To discuss the security of the one-time pad.
- To introduce the concept of entropy.
- To explain the notions of key equivocation, spurious keys and unicity distance.

9.1. Introduction

Information theory is one of the foundations of computer science. In this chapter we will examine its relationship to cryptography. But we shall not assume any prior familiarity with information theory.

We first need to overview the difference between information-theoretic security and computational security. Informally, a cryptographic system is called *computationally secure* if the best *possible* algorithm for breaking it requires N operations, where N is such a large number that it is infeasible to carry out this many operations. With current computing power we assume that 2^{128} operations is an infeasible number of operations to carry out. Hence, a value of N larger than 2^{128} would imply that the system is computationally secure. Note that no actual system can be proved secure under this definition, since we never know whether there is a better algorithm than the one known. Hence, in practice we say a system is computationally secure if the best *known* algorithm for breaking it requires an unreasonably large amount of computational resources.

Another practical approach, related to computational security, is to reduce breaking the system to solving some well-studied hard problem. For example, we can try to show that a given system is secure if a given integer N cannot be factored. Systems of this form are often called provably secure. However, we only have a proof relative to some hard problem, and hence this does not provide a complete guarantee of security.

Essentially, a computationally secure scheme, or one which is provably secure, is only secure when we consider an adversary whose computational resources are bounded. Even if the adversary has large, but limited, resources, she still will not break the system. When considering schemes which are computationally secure we need to be very clear about certain issues:

- We need to be careful about the key sizes etc. If the key size is small then our adversary may have enough computational resources to break the system.
- We need to keep abreast of current algorithmic developments and developments in computer hardware.
- At some point in the future we should expect our system to become broken, either through an improvement in computing power or an algorithmic breakthrough.

It turns out that most schemes in use today are computationally secure, and so every chapter in this book (except this one) will mainly focus on computationally secure systems.

N.P. Smart, *Cryptography Made Simple*, Information Security and Cryptography, DOI 10.1007/978-3-319-21936-3_9

On the other hand, a system is said to be *unconditionally secure* when we place no limit on the computational power of the adversary. In other words a system is unconditionally secure if it cannot be broken even with infinite computing power. Hence, no matter what algorithmic improvements are made or what improvements in computing technology occur, an unconditionally secure scheme will never be broken. Other names for unconditional security you find in the literature are perfect security or information-theoretic security.

You have already seen that the following systems are not computationally secure, since we already know how to break them with very limited computing resources:

- Shift cipher,
- Substitution cipher,
- Vigenère cipher,
- Enigma machine.

Of the systems we shall meet later, the following are computationally secure but are not unconditionally secure:

- DES and AES,
- RSA,
- ElGamal encryption.

However, the one-time pad which we shall meet in this chapter is unconditionally secure, but only if it is used correctly.

9.2. Probability and Ciphers

Before we can formally introduce the concept of unconditional security we first need to understand in more detail the role of probability in simple symmetric ciphers such as those discussed in Chapter 7. We utilize the following notation for various spaces:

- Let $\mathbb{P}$ denote the set of possible plaintexts.
- Let $\mathbb{K}$ denote the set of possible keys.
- Let $\mathbb{C}$ denote the set of ciphertexts.

To each of these sets we can assign a probability distribution, where we denote the probabilities by $p(P = m)$, $p(K = k)$, $p(C = c)$. We denote the encryption function by e_k, and the decryption function by d_k. For example, if our message space is $\mathbb{P} = \{a, b, c, d\}$ and the message a occurs with probability 1/4 then we write

$$p(P = a) = \frac{1}{4}.$$

We make the reasonable assumption that P and K are independent, i.e. the user will not decide to encrypt certain messages under one key and other messages under another. The set of ciphertexts under a specific key k is defined by

$$\mathbb{C}(k) = \{e_k(x) : x \in \mathbb{P}\}.$$

We then have that $p(C = c)$ is defined by

$$p(C = c) = \sum_{k : c \in \mathbb{C}(k)} p(K = k) \cdot p(P = d_k(c)). \tag{9}$$

As an example, which we shall use throughout this section, assume that we have only four messages $\mathbb{P} = \{a, b, c, d\}$ which occur with probability

- $p(P = a) = 1/4$,
- $p(P = b) = 3/10$,
- $p(P = c) = 3/20$,
- $p(P = d) = 3/10$.

Also suppose we have three possible keys given by $\mathbb{K} = \{k_1, k_2, k_3\}$, which occur with probability

- $p(K = k_1) = 1/4$,
- $p(K = k_2) = 1/2$,
- $p(K = k_3) = 1/4$.

Now, suppose we have $\mathbb{C} = \{1, 2, 3, 4\}$, with the encryption function given by the following table.

	a	b	c	d
k_1	3	4	2	1
k_2	3	1	4	2
k_3	4	3	1	2

We can then compute, using formula (9),

$$\begin{aligned}
p(C = 1) &= p(K = k_1) \cdot p(P = d) + p(K = k_2) \cdot p(P = b) + p(K = k_3) \cdot p(P = c) \\
&= 0.2625, \\
p(C = 2) &= p(K = k_1) \cdot p(P = c) + p(K = k_2) \cdot p(P = d) + p(K = k_3) \cdot p(P = d) \\
&= 0.2625, \\
p(C = 3) &= p(K = k_1) \cdot p(P = a) + p(K = k_2) \cdot p(P = a) + p(K = k_3) \cdot p(P = b) \\
&= 0.2625, \\
p(C = 4) &= p(K = k_1) \cdot p(P = b) + p(K = k_2) \cdot p(P = c) + p(K = k_3) \cdot p(P = a) \\
&= 0.2125.
\end{aligned}$$

Hence, the ciphertexts produced are distributed almost uniformly. For $c \in \mathbb{C}$ and $m \in \mathbb{P}$ we can compute the conditional probability $p(C = c \mid P = m)$. This is the probability that c is the ciphertext given that m is the plaintext

$$p(C = c \mid P = m) \;=\; \sum_{k : m = d_k(c)} p(K = k).$$

This sum of probabilities is the sum over all keys k for which the decryption function on input of c will output m. For our prior example we can compute these probabilities as

$$\begin{aligned}
&p(C = 1 \mid P = a) = 0, && p(C = 2 \mid P = a) = 0, \\
&p(C = 3 \mid P = a) = 0.75, && p(C = 4 \mid P = a) = 0.25, \\
\\
&p(C = 1 \mid P = b) = 0.5, && p(C = 2 \mid P = b) = 0, \\
&p(C = 3 \mid P = b) = 0.25, && p(C = 4 \mid P = b) = 0.25, \\
\\
&p(C = 1 \mid P = c) = 0.25, && p(C = 2 \mid P = c) = 0.25, \\
&p(C = 3 \mid P = c) = 0, && p(C = 4 \mid P = c) = 0.5, \\
\\
&p(C = 1 \mid P = d) = 0.25, && p(C = 2 \mid P = d) = 0.75, \\
&p(C = 3 \mid P = d) = 0, && p(C = 4 \mid P = d) = 0.
\end{aligned}$$

However, when we try to break a cipher we want the conditional probability the other way around, i.e. we want to know the probability of a given message occurring given only the ciphertext. We can compute the probability of m being the plaintext given that c is the ciphertext via

$$p(P = m \mid C = c) = \frac{p(P = m) \cdot p(C = c \mid P = m)}{p(C = c)}.$$

This conditional probability can be computed by anyone who knows the encryption function and the probability distributions of K and P. Using these probabilities one may be able to deduce some information about the plaintext once the ciphertext is known.

Returning to our previous example we compute

$$\begin{aligned}
p(P=a \mid C=1) &= 0, & p(P=b \mid C=1) &= 0.571,\\
p(P=c \mid C=1) &= 0.143, & p(P=d \mid C=1) &= 0.286,\\
\\
p(P=a \mid C=2) &= 0, & p(P=b \mid C=2) &= 0,\\
p(P=c \mid C=2) &= 0.143, & p(P=d \mid C=2) &= 0.857,\\
\\
p(P=a \mid C=3) &= 0.714, & p(P=b \mid C=3) &= 0.286,\\
p(P=c \mid C=3) &= 0, & p(P=d \mid C=3) &= 0,\\
\\
p(P=a \mid C=4) &= 0.294, & p(P=b \mid C=4) &= 0.352,\\
p(P=c \mid C=4) &= 0.352, & p(P=d \mid C=4) &= 0.
\end{aligned}$$

Hence

- If we see the ciphertext 1 then we know the message is not equal to a. We also can guess that it is more likely to be b rather than c or d.
- If we see the ciphertext 2 then we know the message is not equal to a or b, and it is quite likely that the message is equal to d.
- If we see the ciphertext 3 then we know the message is not equal to c or d and there is a good chance that it is equal to a.
- If we see the ciphertext 4 then we know the message is not equal to d, but cannot really guess with confidence whether the message is a, b or c.

So in our previous example the ciphertext does reveal a lot of information about the plaintext. But this is exactly what we wish to avoid: We want the ciphertext to give no information about the plaintext. A system with this property, that the ciphertext reveals nothing about the plaintext, is said to be *perfectly secure*.

Definition 9.1 (Perfect Secrecy). *A cryptosystem has perfect secrecy if*

$$p(P=m \mid C=c) = p(P=m)$$

for all plaintexts m and all ciphertexts c.

This means the probability that the plaintext is m, given that we know the ciphertext is c, is the same as the probability that it is m without seeing c. In other words knowing c reveals no information about m. Another way of describing perfect secrecy is the following.

Lemma 9.2. *A cryptosystem has perfect secrecy if $p(C=c \mid P=m) = p(C=c)$ for all m and c.*

PROOF. This follows trivially from the definition

$$p(P=m \mid C=c) = \frac{p(P=m)p(C=c \mid P=m)}{p(C=c)}$$

and the fact that perfect secrecy means $p(P=m \mid C=c) = p(P=m)$. □

The first result about perfect security is as follows.

Lemma 9.3. *Assume the cryptosystem is perfectly secure, then*

$$\#\mathbb{K} \geq \#\mathbb{C} \geq \#\mathbb{P},$$

where

- *$\#\mathbb{K}$ denotes the size of the set of possible keys,*
- *$\#\mathbb{C}$ denotes the size of the set of possible ciphertexts,*
- *$\#\mathbb{P}$ denotes the size of the set of possible plaintexts.*

Proof. First note that in any encryption scheme, we must have

$$\#\mathbb{C} \geq \#\mathbb{P}$$

since encryption must be an injective map; we have to be able to decrypt after all.

We assume that every ciphertext can occur, i.e. $p(C = c) > 0$ for all $c \in \mathbb{C}$, since if this does not hold then we can alter our definition of $\mathbb{C}$. Then for any message m and any ciphertext c we have

$$p(C = c \mid P = m) = p(C = c) > 0.$$

For each m, this means that for all c there must be a key k such that

$$e_k(m) = c.$$

Hence, $\#\mathbb{K} \geq \#\mathbb{C}$ as required. □

We now come to the main theorem on perfectly secure ciphers, due to Shannon. Shannon's Theorem tells us exactly which encryption schemes are perfectly secure and which are not.

Theorem 9.4 (Shannon). *Let $(\mathbb{P}, \mathbb{C}, \mathbb{K}, e_k(\cdot), d_k(\cdot))$ denote a cryptosystem with $\#\mathbb{P} = \#\mathbb{C} = \#\mathbb{K}$. Then the cryptosystem provides perfect secrecy if and only if*

- *Every key is used with equal probability $1/\#\mathbb{K}$,*
- *For each $m \in \mathbb{P}$ and $c \in \mathbb{C}$ there is a unique key k such that $e_k(m) = c$.*

Proof. Note the statement is *if and only if*; hence we need to prove it in both directions. We first prove the *only if* part.

Suppose the system gives perfect secrecy. Then we have already seen, in the proof of Lemma 9.3, that for all $m \in \mathbb{P}$ and $c \in \mathbb{C}$ there is a key k such that $e_k(m) = c$. Now, since we have assumed $\#\mathbb{C} = \#\mathbb{K}$ we have

$$\#\{e_k(m) : k \in \mathbb{K}\} = \#\mathbb{K}$$

i.e. there do not exist two keys k_1 and k_2 such that

$$e_{k_1}(m) = e_{k_2}(m) = c.$$

So for all $m \in \mathbb{P}$ and $c \in \mathbb{C}$ there is exactly one $k \in \mathbb{K}$ such that $e_k(m) = c$. We need to show that every key is used with equal probability, i.e. $p(K = k) = 1/\#\mathbb{K}$ for all $k \in \mathbb{K}$.

Let $n = \#\mathbb{K}$ and $\mathbb{P} = \{m_i : 1 \leq i \leq n\}$, fix $c \in \mathbb{C}$ and label the keys $k_1, \ldots, k_n$ such that $e_{k_i}(m_i) = c$ for $1 \leq i \leq n$. We then have, noting that due to perfect secrecy $p(P = m_i \mid C = c) = p(P = m_i)$,

$$\begin{aligned} p(P = m_i) &= p(P = m_i \mid C = c) \\ &= \frac{p(C = c \mid P = m_i) \cdot p(P = m_i)}{p(C = c)} \\ &= \frac{p(K = k_i) \cdot p(P = m_i)}{p(C = c)}. \end{aligned}$$

Hence we obtain, for all $1 \leq i \leq n$, that $p(C = c) = p(K = k_i)$. This says that the keys are used with equal probability and hence $p(K = k) = 1/\#\mathbb{K}$ for all $k \in \mathbb{K}$.

Now we need to prove the result in the other direction. Namely, if

- $\#\mathbb{K} = \#\mathbb{C} = \#\mathbb{P}$,
- Every key is used with equal probability $1/\#\mathbb{K}$,

- For each $m \in \mathbb{P}$ and $c \in \mathbb{C}$ there is a unique key k such that $e_k(m) = c$,

then we need to show the system is perfectly secure, i.e. for all m and c that

$$p(P = m \mid C = c) = p(P = m).$$

Since each key is used with equal probability, we have

$$\begin{aligned} p(C = c) &= \sum_k p(K = k) \cdot p(P = d_k(c)) \\ &= \frac{1}{\#\mathbb{K}} \sum_k p(P = d_k(c)). \end{aligned}$$

Also, since for each m and c there is a unique key k with $e_k(m) = c$, we must have

$$\sum_k p(P = d_k(c)) = \sum_m p(P = m) = 1.$$

Hence, $p(C = c) = 1/\#\mathbb{K}$. In addition, if $c = e_k(m)$ then $p(C = c \mid P = m) = p(K = k) = 1/\#\mathbb{K}$. So using Bayes' Theorem we have

$$\begin{aligned} p(P = m \mid C = c) &= \frac{p(P = m) \cdot p(C = c \mid P = m)}{p(C = c)} \\ &= \frac{p(P = m) \cdot \frac{1}{\#\mathbb{K}}}{\frac{1}{\#\mathbb{K}}} \\ &= p(P = m). \end{aligned}$$

□

We end this section by discussing a couple of systems which have perfect secrecy.

9.2.1. Modified Shift Cipher: Recall that the shift cipher is one in which we "add" a given letter (the key) to each letter of the plaintext to obtain the ciphertext. We now modify this cipher by using a different key for each plaintext letter. For example, to encrypt the message HELLO we choose five random keys, say FUIAT. We then add the key to the plaintext, modulo 26, to obtain the ciphertext MYTLH. Notice how the plaintext letter L encrypts to different letters in the ciphertext.

When we use the shift cipher with a different random key for each letter, we obtain a perfectly secure system. To see why this is so, consider the situation of encrypting a message of length n. Then the total number of keys, ciphertexts and plaintexts are all equal, namely:

$$\#\mathbb{K} = \#\mathbb{C} = \#\mathbb{P} = 26^n.$$

In addition each key will occur with equal probability:

$$p(K = k) = \frac{1}{26^n},$$

and for each m and c there is a unique k such that $e_k(m) = c$. Hence, by Shannon's Theorem this modified shift cipher is perfectly secure.

9.2.2. Vernam Cipher: The above modified shift cipher basically uses addition modulo 26. One problem with this is that in a computer, or any electrical device, mod 26 arithmetic is hard, but binary arithmetic is easy. We are particularly interested in the addition operation, which is denoted by $\oplus$ and is equal to the logical exclusive-or operation.

$\oplus$	0	1
0	0	1
1	1	0

In 1917 Gilbert Vernam patented a cipher which used these principles, called the *Vernam cipher* or *one-time pad*. To send a binary string we need a key, which is a binary string as long as the message. To encrypt a message we exclusive-or each bit of the plaintext with each bit of the key to produce the ciphertext.

Each key is only allowed to be used once, hence the term *one-time* pad. This means that key distribution is a problem, which we shall come back to again and again. To see why we cannot get away with using a key twice, consider the following chosen plaintext attack. We assume that Alice always uses the same key k to encrypt a message to Bob. Eve wishes to determine this key and so carries out the following attack:

- Eve generates m and asks Alice to encrypt it.
- Eve obtains $c = m \oplus k$.
- Eve now computes $k = c \oplus m$.

You may object to this attack since it requires Alice to be particularly stupid, in that she encrypts a message for Eve. But in designing our cryptosystems we should try to make systems which are secure even against stupid users.

Another problem with using the same key twice is the following. Suppose Eve can intercept two messages encrypted with the same key

$$c_1 = m_1 \oplus k,$$
$$c_2 = m_2 \oplus k.$$

Eve can now determine some partial information about the pair of messages m_1 and m_2 since she can compute

$$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2.$$

Despite the problems associated with key distribution, the one-time pad has been used in the past in military and diplomatic contexts.

9.3. Entropy

If every message we send requires a key as long as the message, and we never encrypt two messages with the same key, then encryption will not be very useful in everyday applications such as Internet transactions. This is because getting the key from one person to another will be an impractical task. After all, one cannot encrypt it since that would require another key. This problem is called the key distribution problem.

To simplify the key distribution problem we need to turn from perfectly secure encryption algorithms to ones which are, we hope, computationally secure. This is the goal of modern cryptography, where one aims to build systems such that

- one key can be used many times,
- a small key can encrypt a long message.

Such systems will not be unconditionally secure, by Shannon's Theorem, and so must be at best only computationally secure.

We now need to develop the information theory needed to deal with these computationally secure systems. Again the main results are due to Shannon in the late 1940s. In particular, we shall use Shannon's idea of using *entropy* as a way of measuring information.

The word entropy is another name for uncertainty, and the basic tenet of information theory is that uncertainty and information are essentially the same thing. This takes some getting used to, but consider that if you are uncertain what something means then revealing the meaning gives you information. As a cryptographic application, suppose you want to determine the information in a ciphertext, in other words you want to know what the ciphertext's true meaning is. The entropy in the ciphertext is the amount of uncertainty you have about the underlying plaintext. If X is a random variable, the amount of entropy (in bits) associated with X is denoted by $H(X)$. We shall define this quantity formally in a second. First, let us look at a simple example to help clarify ideas.

Suppose X is the answer to some question, i.e. *Yes* or *No*. If you know I will always say *Yes*, then my answer gives you no information. So the information contained in X should be zero, i.e. $H(X) = 0$. There is no uncertainty about what I will say, hence no information is given by me saying it, hence there is no entropy. On the other hand, if you have no idea what I will say and I reply *Yes* with equal probability to replying *No* then I am revealing one bit of information. Hence, we should have $H(X) = 1$.

Note that the entropy does not depend on the length of the actual message; in the above case we have a message of length at most three letters but the amount of information is at most one bit. We can now define formally the notion of entropy.

Definition 9.5 (Entropy). *Let X be a random variable which takes a finite set of values x_i, with $1 \le i \le n$, and has probability distribution $p_i = p(X = x_i)$, where we use the convention that if $p_i = 0$ then $p_i \log_2 p_i = 0$. The entropy of X is defined to be*

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

Let us return to our *Yes* or *No* question above and show that this definition of entropy coincides with our intuition. Recall that X is the answer to some question with responses *Yes* or *No*. If you know I will always say *Yes* then $p_1 = 1$ and $p_2 = 0$. We compute $H(X) = -1 \cdot \log_2 1 - 0 \cdot \log_2 0 = 0$. Hence, my answer reveals no information to you. If you have no idea what I will say and I reply *Yes* with equal probability to replying *No* then $p_1 = p_2 = 1/2$. We now compute

$$H(X) = -\frac{\log_2 \frac{1}{2}}{2} - \frac{\log_2 \frac{1}{2}}{2} = 1.$$

Hence, my answer reveals one bit of information to you.

9.3.1. Properties of Entropy:

A number of elementary properties of entropy follow immediately from the definition.

- We always have $H(X) \ge 0$.
- The only way to obtain $H(X) = 0$ is if for some i we have $p_i = 1$ and $p_j = 0$ when $i \ne j$.
- If $p_i = 1/n$ for all i then $H(X) = \log_2 n$.

Another way of looking at entropy is that it measures how much one can compress information. If I send a single ASCII character to signal *Yes* or *No*, for example I could simply send *Y* or *N*, I am actually sending eight bits of data, but I am only sending one bit of information. If I wanted to I could compress the data down to 1/8th of its original size. Hence, naively if a message of length n can be compressed to a proportion ϵ of its original size then it contains $\epsilon \cdot n$ bits of information in it.

We now derive an upper bound for the entropy of a random variable, to go with our lower bound of $H(X) \ge 0$. To do this we will need the following special case of Jensen's inequality.

Theorem 9.6 (Jensen's Inequality). *Suppose*

$$\sum_{i=1}^{n} a_i = 1$$

with $a_i > 0$ for $1 \le i \le n$. Then, for $x_i > 0$,

$$\sum_{i=1}^{n} a_i \cdot \log_2 x_i \le \log_2 \left(\sum_{i=1}^{n} a_i \cdot x_i \right).$$

Equality occurs if and only if $x_1 = x_2 = \ldots = x_n$.

Using this we can now prove the following theorem.

Theorem 9.7. *If X is a random variable which takes n possible values then*

$$0 \leq H(X) \leq \log_2 n.$$

The lower bound is obtained if one value occurs with probability one, and the upper bound is obtained if all values are equally likely.

Proof. We have already discussed the facts about the lower bound so we will concentrate on the statements about the upper bound. The hypothesis is that X is a random variable with probability distribution $p_1, \ldots, p_n$, with $p_i > 0$ for all i. One can then deduce the following sequence of inequalities:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} p_i \cdot \log_2 p_i \\ &= \sum_{i=1}^{n} p_i \cdot \log_2 \frac{1}{p_i} \\ &\leq \log_2 \left(\sum_{i=1}^{n} \left(p_i \cdot \frac{1}{p_i} \right) \right) && \text{by Jensen's inequality} \\ &= \log_2 n. \end{aligned}$$

To obtain equality, we require equality when we apply Jensen's inequality. But this will only occur when $p_i = 1/n$ for all i, in other words, when all values of X are equally likely. □

9.3.2. Joint and Conditional Entropy: The basics of the theory of entropy closely match those of the theory of probability. For example, if X and Y are random variables then we define the joint probability distribution as

$$r_{i,j} = p(X = x_i \text{ and } Y = y_j)$$

for $1 \leq i \leq n$ and $1 \leq j \leq m$. The joint entropy is then obviously defined as

$$H(X, Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} r_{i,j} \cdot \log_2 r_{i,j}.$$

You should think of the joint entropy $H(X, Y)$ as the total amount of information contained in one observation of $(x, y) \in X \times Y$. We then obtain the inequality

$$H(X, Y) \leq H(X) + H(Y)$$

with equality if and only if X and Y are independent. We leave the proof of this as an exercise.

Just as with probability theory, where one has the linked concepts of joint probability and conditional probability, so the concept of joint entropy is linked to the concept of conditional entropy. This is important to understand, since conditional entropy is the main tool we shall use in understanding non-perfect ciphers in the rest of this chapter. Let X and Y be two random variables. Recall we defined the conditional probability distribution as

$$p(X = x \mid Y = y) = \text{ Probability that } X = x \text{ given } Y = y.$$

The entropy of X given an observation of $Y = y$ is then defined in the obvious way by

$$H(X \mid Y = y) = -\sum_{x} p(X = x \mid Y = y) \cdot \log_2 p(X = x \mid Y = y).$$

Given this, we define the conditional entropy of X given Y as

$$\begin{aligned} H(X \mid Y) &= \sum_y p(Y = y) \cdot H(X \mid Y = y) \\ &= -\sum_x \sum_y p(Y = y) \cdot p(X = x \mid Y = y) \cdot \log_2 p(X = x \mid Y = y). \end{aligned}$$

This is the amount of uncertainty about X that is left after revealing a value of Y. The conditional and joint entropy are linked by the following formula

$$H(X, Y) = H(Y) + H(X \mid Y)$$

and we have the following upper bound

$$H(X \mid Y) \leq H(X)$$

with equality if and only if X and Y are independent. Again, we leave the proof of these statements as an exercise.

9.3.3. Application to Ciphers: Now turning to cryptography again, we have some trivial statements relating the entropy of P, K and C.

- $H(P \mid K, C) = 0$:
  If you know the ciphertext and the key then you know the plaintext. This must hold since otherwise decryption will not work correctly.
- $H(C \mid P, K) = 0$:
  If you know the plaintext and the key then you know the ciphertext. This holds for all ciphers we have seen so far, and holds for all the block ciphers we shall see in later chapters. However, for modern encryption schemes we do not have this last property when they are used correctly, as many ciphertexts can correspond to the same plaintext.

In addition we have the following identities

$$\begin{aligned} H(K, P, C) &= H(P, K) + H(C \mid P, K) && \text{as } H(X, Y) = H(Y) + H(X \mid Y) \\ &= H(P, K) && \text{as } H(C \mid P, K) = 0 \\ &= H(K) + H(P) && \text{as } K \text{ and } P \text{ are independent} \end{aligned}$$

and

$$\begin{aligned} H(K, P, C) &= H(K, C) + H(P \mid K, C) && \text{as } H(X, Y) = H(Y) + H(X \mid Y) \\ &= H(K, C) && \text{as } H(P \mid K, C) = 0. \end{aligned}$$

Hence, we obtain

$$H(K, C) = H(K) + H(P).$$

This last equality is important since it is related to the conditional entropy $H(K \mid C)$, which is called the *key equivocation*. The key equivocation is the amount of uncertainty left about the key after one ciphertext is revealed. Recall that our goal is to determine the key given the ciphertext. Putting two of our prior equalities together we find

$$H(K \mid C) = H(K, C) - H(C) = H(K) + H(P) - H(C). \tag{10}$$

In other words, the uncertainty about the key left after we reveal a ciphertext is equal to the uncertainty in the plaintext and the key minus the uncertainty in the ciphertext.

Let us return to our baby cryptosystem considered in the previous section. Recall we had the probability spaces

$$\mathbb{P} = \{a, b, c, d\}, \ \ \mathbb{K} = \{k_1, k_2, k_3\} \text{ and } \mathbb{C} = \{1, 2, 3, 4\},$$

with the associated probabilities:

- $p(P = a) = 0.25$, $p(P = b) = p(P = d) = 0.3$ and $p(P = c) = 0.15$,

- $p(K = k_1) = p(K = k_3) = 0.25$ and $p(K = k_2) = 0.5$,
- $p(C = 1) = p(C = 2) = p(C = 3) = 0.2625$ and $p(C = 4) = 0.2125$.

We can then calculate the relevant entropies as:

$$\begin{aligned} H(P) &\approx 1.9527, \\ H(K) &\approx 1.5, \\ H(C) &\approx 1.9944. \end{aligned}$$

Hence

$$H(K \mid C) \approx 1.9527 + 1.5 - 1.9944 \approx 1.4583.$$

So around one and a half bits of information about the key are left to be found, on average, after a single ciphertext is observed. This explains why the system leaks information, and shows that it cannot be secure. After all there are only 1.5 bits of uncertainty about the key to start with; one ciphertext leaves us with 1.4593 bits of uncertainty. Hence, $1.5 - 1.4593 = 0.042$ bits of information about the key are revealed by a single ciphertext, or equivalently three percent of the key.

9.4. Spurious Keys and Unicity Distance

In our baby example above, information about the key is leaked by an individual ciphertext, since knowing the ciphertext rules out a certain subset of the keys. Of the remaining possible keys, only one is correct. The remaining possible, but incorrect, keys are called the *spurious keys*.

Consider the (unmodified) shift cipher, i.e. where the same key is used for each letter. Suppose the ciphertext is WNAJW, and suppose we know that the plaintext is an English word. The only "meaningful" plaintexts are RIVER and ARENA, which correspond to the two possible keys F and W. One of these keys is the correct one and one is spurious.

We can now explain why it was easy to break the substitution cipher in terms of a concept called the *unicity distance* of the cipher. We shall explain this relationship in more detail, but we first need to understand the underlying plaintext in more detail. The plaintext in many computer communications can be considered as a random bit string. But often this is not so. Sometimes one is encrypting an image or sometimes one is encrypting plain English text. In our discussion we shall consider the case when the underlying plaintext is taken from English, as in the substitution cipher. Such a language is called a *natural language* to distinguish it from the bit streams used by computers to communicate.

We first wish to define the entropy (or information) per letter H_L of a natural language such as English. Note that a random string of alphabetic characters would have entropy

$$\log_2 26 \approx 4.70.$$

So we have $H_L \leq 4.70$. If we let P denote the random variable of letters in the English language then we have

$$p(P = a) = 0.082, \ \ldots, \ p(P = e) = 0.127, \ \ldots, \ p(P = z) = 0.001.$$

We can then compute

$$H_L \leq H(P) \approx 4.14.$$

Hence, instead of 4.7 bits of information per letter, if we only examine the letter frequencies we conclude that English conveys around 4.14 bits of information per letter.

But this is a gross overestimate, since letters are not independent. For example Q is almost always followed by U and the bigram TH is likely to be very common. One would suspect that a better statistic for the amount of entropy per letter could be obtained by looking at the distribution of bigrams. Hence, we let P^2 denote the random variable of bigrams. If we let $p(P = i, P' = j)$

denote the random variable which is assigned the probability that the bigram "ij" appears, then we define

$$H(P^2) = -\sum_{i,j} p(P=i, P'=j) \cdot \log p(P=i, P'=j).$$

A number of people have computed values of $H(P^2)$ and it is commonly accepted to be given by

$$H(P^2) \approx 7.12.$$

We want the entropy per letter so we compute

$$H_L \leq H(P^2)/2 \approx 3.56.$$

But again this is an overestimate, since we have not taken into account that the most common trigram is *THE*. Hence, we can also look at P^3 and compute $H(P^3)/3$. This will also be an overestimate, and so on,... This leads us to the following definition.

Definition 9.8. *The entropy of the natural language L is defined to be*

$$H_L = \lim_{n \longrightarrow \infty} \frac{H(P^n)}{n}.$$

The exact value of H_L is hard to compute exactly but we can approximate it. In fact one has, by experiment, that for English

$$1.0 \leq H_L \leq 1.5.$$

So each letter in English

- requires $5 = \lceil \log_2(26) \rceil$ bits of data to represent it,
- only gives at most 1.5 bits of information.

This shows that English contains a high degree of redundancy, in that there is far less information conveyed in an English sentence than the amount of data needed to represent the sentence. One can see this from the following, which you can still hopefully read (just) even though I have deleted two out of every four letters,

On** up** a t**e t**re **s a **rl **ll** Sn** Wh**e.

The *redundancy* of a language is defined by

$$R_L = 1 - \frac{H_L}{\log_2 \#\mathbb{P}},$$

and it expresses the percentage of text in the language which can be removed (in principle) without affecting the overall meaning. If we take $H_L \approx 1.25$ then the redundancy of English is

$$R_L \approx 1 - \frac{1.25}{\log_2 26} = 0.75.$$

So this means that we should be able to compress an English text file of around 10 MB down to 2.5 MB.

9.4.1. Redundancy and Ciphertexts: We now return to a general cipher and suppose $c \in \mathbb{C}^n$, i.e. c is a ciphertext consisting of n characters. We define $\mathbb{K}(c)$ to be the set of keys which produce a "meaningful" decryption of c. Then, clearly $\#\mathbb{K}(c) - 1$ is the number of spurious keys given c.

The average number of spurious keys is defined to be $\overline{s}_n$, where

$$\begin{aligned}\overline{s}_n &= \sum_{c\in\mathbb{C}^n} p(C=c)\cdot(\#\mathbb{K}(c)-1)\\ &= \sum_{c\in\mathbb{C}^n} p(C=c)\cdot\#\mathbb{K}(c) - \sum_{c\in\mathbb{C}^n} p(C=c)\\ &= \left(\sum_{c\in\mathbb{C}^n} \#\mathbb{K}(c)\cdot p(C=c)\right) - 1.\end{aligned}$$

Now if n is sufficiently large and $\#\mathbb{P}=\#\mathbb{C}$ we obtain

$$\begin{aligned}\log_2(\overline{s}_n+1) &= \log_2\left(\sum_{c\in\mathbb{C}^n} \#\mathbb{K}(c)\cdot p(C=c)\right)\\ &\geq \sum_{c\in\mathbb{C}^n} p(C=c)\cdot\log_2\#\mathbb{K}(c) && \text{by Jensen's inequality}\\ &\geq \sum_{c\in\mathbb{C}^n} p(C=c)\cdot H(K\mid c)\\ &= H(K\mid C^n) && \text{by definition}\\ &= H(K)+H(P^n)-H(C^n) && \text{equation (10)}\\ &\approx H(K)+n\cdot H_L - H(C^n) && \text{if } n \text{ is very large}\\ &= H(K)-H(C^n)\\ &\qquad + n\cdot(1-R_L)\cdot\log_2\#\mathbb{P} && \text{by definition of } R_L\\ &\geq H(K)-n\cdot\log_2\#\mathbb{C}\\ &\qquad + n\cdot(1-R_L)\cdot\log_2\#\mathbb{P} && \text{as } H(C^n)\leq n\cdot\log_2\#\mathbb{C}\\ &= H(K)-n\cdot R_L\cdot\log_2\#\mathbb{P} && \text{as } \#\mathbb{P}=\#\mathbb{C}.\end{aligned}$$

So, if n is sufficiently large and $\#\mathbb{P}=\#\mathbb{C}$ then

$$\overline{s}_n \geq \frac{\#\mathbb{K}}{\#\mathbb{P}^{n\cdot R_L}} - 1.$$

As an attacker we would like the number of spurious keys to become zero, and it is clear that as we take longer and longer ciphertexts then the number of spurious keys must go down.

The unicity distance n_0 of a cipher is the value of n for which the expected number of spurious keys becomes zero. In other words this is the average amount of ciphertext needed before an attacker can determine the key, assuming the attacker has infinite computing power. For a perfect cipher we have $n_0=\infty$, but for other ciphers the value of n_0 can be alarmingly small. We can obtain an estimate of n_0 by setting $\overline{s}_n=0$ in

$$\overline{s}_n \geq \frac{\#\mathbb{K}}{\#\mathbb{P}^{n\cdot R_L}} - 1$$

to obtain

$$n_0 \approx \frac{\log_2\#\mathbb{K}}{R_L\cdot\log_2\#\mathbb{P}}.$$

In the substitution cipher we have

$$\begin{aligned}\#\mathbb{P} &= 26,\\ \#\mathbb{K} &= 26! \approx 4\cdot 10^{26}\end{aligned}$$

and using our value of $R_L = 0.75$ for English we can approximate the unicity distance as

$$n_0 \approx \frac{88.4}{0.75 \times 4.7} \approx 25.$$

So we require on average only 25 ciphertext characters before we can break the substitution cipher, again assuming infinite computing power. In any case after 25 characters we expect a unique valid decryption.

Now assume we have a modern cipher which encrypts bit strings using keys of bit length l. We have

$$\#\mathbb{P} = 2,$$
$$\#\mathbb{K} = 2^l.$$

Again we assume $R_L = 0.75$, which is an underestimate since we now need to encode English into a computer communications medium such as ASCII. Then the unicity distance is

$$n_0 \approx \frac{l}{0.75} = \frac{4 \cdot l}{3}.$$

Now assume instead of transmitting the plain ASCII we compress it first. If we assume a perfect compression algorithm then the plaintext will have no redundancy and so $R_L \approx 0$. In which case the unicity distance is

$$n_0 \approx \frac{l}{0} = \infty.$$

So you may ask whether modern ciphers encrypt plaintexts with no redundancy? The answer is no; even if one compresses the data, a modern cipher often adds some redundancy to the plaintext before encryption. The reason is that we have only considered passive attacks, i.e. an attacker has been only allowed to examine ciphertexts and from these ciphertexts the attacker's goal is to determine the key. There are other types of attack called active attacks; in these an attacker is allowed to generate plaintexts or ciphertexts of her choosing and ask the key holder to encrypt or decrypt them, the two variants being called a chosen plaintext attack and a chosen ciphertext attack respectively. In public key systems that we shall see later, chosen plaintext attacks cannot be stopped since anyone is allowed to encrypt anything.

We would like to stop chosen ciphertext attacks for all types of cipher. The current wisdom for encryption algorithms is to make the cipher add some redundancy to the plaintext before it is encrypted. In this way it is hard for an attacker to produce a ciphertext which has a valid decryption. The philosophy is that it is then hard for an attacker to mount a chosen ciphertext attack, since it will be hard for an attacker to choose a valid ciphertext for a decryption query. We shall discuss this more in later chapters.

Chapter Summary

- A cryptographic system for which knowing the ciphertext reveals no more information than if you did not know the ciphertext is called a perfectly secure system.
- Perfectly secure systems exist, but they require keys as long as the message and a different key to be used with each new encryption. Hence, perfectly secure systems are not very practical.
- Information and uncertainty are essentially the same thing.
- The amount of uncertainty in a random variable is measured by its entropy.
- An attacker really wants, given the ciphertext, to determine some information about the plaintext.

- The equation $H(K \mid C) = H(K) + H(P) - H(C)$ allows us to estimate how much uncertainty remains about the key after one observes a single ciphertext.
- The natural redundancy of English means that a naive cipher does not need to produce a lot of ciphertext before the underlying plaintext can be discovered.

Further Reading

Our discussion of Shannon's theory has closely followed the treatment in the book by Stinson. Another possible source of information is the book by Welsh. A general introduction to information theory, including its application to coding theory, is in the book by van der Lubbe.

J.C.A. van der Lubbe. *Information Theory.* Cambridge University Press, 1997.

D. Stinson. *Cryptography: Theory and Practice.* Third Edition. CRC Press, 2005.

D. Welsh. *Codes and Cryptography.* Oxford University Press, 1988.