Cryptography: An Introduction (3rd Edition)

Nigel Smart

Chapter 5 — Edited for CIS 331 (cut short)

CHAPTER 5

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 notion of key equivocation, spurious keys and unicity distance.
- To use these tools to understand why the prior historical encryption algorithms are weak.

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^{80} operations is an infeasible number of operations to carry out. Hence, a value of N larger than 2^{80} would imply that the system is computationally secure. Notice 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, hence this does not provide an absolute proof.

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 solely be interested in computationally secure systems.

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.

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

- DES and Rijndael,
- 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.

2. Probability and Ciphers

Before we can introduce formally the concept of unconditional security we first need to understand in more detail the role of probability in understanding simple ciphers. We make the following definitions:

- 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.

Each of these can be thought of as a probability distribution, where we denote the probabilities by $p(P = m), p(K = k), p(C = c)$.

So for example, if our message space is $\mathbb{P} = \{a, b, c\}$ 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

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

where the encryption function is defined by $e_k(m)$. We then have the relationship

$$p(C = c) = \sum_{k : c \in \mathbb{C}(k)} p(K = k) \cdot p(P = d_k(c)), \tag{7}$$

where the decryption function is defined by $d_k(c)$. 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 (7),

$$\begin{aligned} p(C = 1) &= p(K = k_1)p(P = d) + p(K = k_2)p(P = b) \\ &\qquad + p(K = k_3)p(P = c) = 0.2625, \end{aligned}$$

$$\begin{aligned} p(C = 2) &= p(K = k_1)p(P = c) + p(K = k_2)p(P = d) \\ &\qquad + p(K = k_3)p(P = d) = 0.2625, \end{aligned}$$

$$\begin{aligned} p(C = 3) &= p(K = k_1)p(P = a) + p(K = k_2)p(P = a) \\ &\qquad + p(K = k_3)p(P = b) = 0.2625, \end{aligned}$$

$$\begin{aligned} p(C = 4) &= p(K = k_1)p(P = b) + p(K = k_2)p(P = c) \\ &\qquad + p(K = k_3)p(P = a) = 0.2125. \end{aligned}$$

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

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

This sum 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

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

But, 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 c is the ciphertext via,

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

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 you have seen the ciphertext.

Returning to our previous example we compute

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

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. We also can be pretty certain 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 have 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 certainty as to 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 5.1 (Perfect Secrecy). *A cryptosystem has perfect secrecy if*

$$p(P=m|C=c)=p(P=m)$$

for all plaintexts m and all ciphertexts c.

This means the probability that the plaintext is m, given that you 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 via:

LEMMA 5.2. *A cryptosystem has perfect secrecy if $p(C=c|P=m)=p(C=c)$ for all m and c.*

PROOF. This trivially follows from the definition

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

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

The first result about a perfect security is

LEMMA 5.3. *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 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

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

This means for each m, 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 due to Shannon on perfectly secure ciphers. Shannon's Theorem tells us exactly which encryption schemes are perfectly secure and which are not.

Theorem 5.4 (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 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.

$$p(K = k) = 1/\#\mathbb{K} \text{ for all } k \in \mathbb{K}.$$

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

$$e_{k_i}(m_i) = c \text{ for } 1 \leq i \leq n.$$

We then have, noting that due to perfect secrecy $p(P = m_i|C = c) = p(P = m_i)$,

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

Hence we obtain, for all $1 \leq i \leq n$,

$$p(C = c) = p(K = k_i).$$

This says that the keys are used with equal probability and hence

$$p(K = k) = 1/\#\mathbb{K} \text{ for all } k \in \mathbb{K}.$$

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.

$$p(P = m|C = c) = p(P = m).$$

We have, since each key is used with equal probability,

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

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 $p(C = c|P = m) = p(K = k) = 1/\#\mathbb{K}$. So using Bayes' Theorem we have

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

□

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

2.1. Modified Shift Cipher. Recall the shift cipher is one in which we 'add' a given letter (the key) onto 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 onto 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.

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, or XOR, 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 you need a key, which is a binary string as long as the message. To encrypt a message we XOR 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 pain, 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 and 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.