1 Perfect secrecy of the one-time pad

In this section, we make more a more precise analysis of the security of the one-time pad.

First, we need to define *conditional probability*. Let's consider an example. We know that if it rains Saturday, then there is a reasonable chance that it will rain on Sunday. To make this more precise, we want to compute the probability that it rains on Sunday, given that it rains on Saturday. So we restrict our attention to only those situations where it rains on Saturday and count how often this happens over several years. Then we count how often it rains on both Saturday and Sunday. The ratio gives an estimate of the desired probability. If we call A the event that it rains on Saturday and B the event that it rains on Sunday, then the intersection $A \cap B$ is when it rains on both days. The *conditional probability of A given B* is defined to be

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)},$$

where $P(A)$ denotes the probability of the event A. This formula can be used to define the conditional probability of one event given another for any two events A and B that have probabilities (we implicitly assume throughout this discussion that any probability that occurs in a denominator has nonzero probability).

Events A and B are *independent* if $P(A \cap B) = P(A)\,P(B)$. For example, if Alice flips a fair coin, let A be the event that the coin ends up Heads. If Bob rolls a fair six-sided die, let B be the event that he rolls a 3. Then $P(A) = 1/2$ and $P(B) = 1/6$. Since all 12 combinations of $\{Head, Tail\}$ and $\{1, 2, 3, 4, 5, 6\}$ are equally likely, $P(A \cap B) = 1/12$, which equals $P(A)\,P(B)$. Therefore, A and B are independent.

If A and B are independent, then

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A)\,P(B)}{P(A)} = P(B),$$

which means that knowing that A happens does not change the probability that B happens. By reversing the steps in the above equation, we see that

$$A \text{ and } B \text{ are independent} \iff P(B \mid A) = P(B).$$

An example of events that are not independent is the original example, where A is the event that it rains on Saturday and B is the event that it rains on Sunday, since $P(B \mid A) > P(B)$.[1]

How does this relate to cryptography? In a cryptosystem, there is a set of possible keys. Let's say we have N keys. If we have a perfect random number generator to choose the keys, then the probability that the key is k is $P(K = k) = 1/N$. In this case we say that the key is chosen uniformly

[1] Unfortunately, a widely-used high school algebra text published around 2005 gave exactly one example of independent events: these A and B.

randomly. In any case, we assume that each key has a certain probability of being chosen. We also have various possible plaintexts m and each one has a certain probability $P(M = m)$ of being used. These plaintexts usually do not all have the same probability. For example, the message *attack at noon* might be more probable than *two plus two equals seven.* Finally, each possible ciphertext c has a probability $P(C = c)$.

We say that a cryptosystem is *perfectly secret* if

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

for all possible plaintexts m and all possible ciphertexts c. In other words, knowledge of the ciphertext never changes the probability that a given plaintext occurs. This means that eavesdropping gives no advantage to Eve if she wants to guess the message.

We can now formalize what we claimed about the one-time pad. We work with plaintexts and keys of a fixed (non-variable) length.

Theorem 1. *If the key is chosen uniformly randomly from all keys of a given length, then the one-time pad is perfectly secret.*

Proof. We need to show that $P(M = m \mid C = c) = P(M = m)$ for each pair m, c.

Let's say that there are N keys, each of which has probability $1/N$. We start by showing that each possible ciphertext c also has probability $1/N$. If we have c, we know that it came from some key. The different choices of keys are disjoint events, so we can add the probabilities for the various choices of the key:

$$P(C = c) = \sum_k P(C = c \cap K = k).$$

We might want to say that the ciphertext and the key are independent, but this is not the case. For example, if one message m is very likely, then knowing that $C = c$ makes it likely that $K = c \oplus m$. However, the key and the ciphertext are chosen independently, so we change the above expression in order to use this fact:

$$\begin{aligned} P(C = c \cap K = k) &= P(M = c \oplus k \cap K = k) \\ &= P(M = c \oplus k)\, P(K = k) \\ &= P(M = c \oplus k)\, (1/N). \end{aligned}$$

We know that as k runs through all possible keys, $c \oplus k$ runs through all possible messages. This means that

$$\sum_k P(M = c \oplus k) = P(M = \text{ some possible message}) = 1.$$

Therefore, we can combine the above equations to obtain

$$P(C = c) = \sum_k P(C = c \cap K = k) = (1/N) \sum_k P(M = c \oplus k) = 1/N.$$

The definition of conditional probability and the independence of K and M yield

$$\begin{aligned} P(M = m \mid C = c)P(C = c) &= P(C = c \cap M = m) \\ &= P(K = c \oplus m \cap M = m) \\ &= P(K = c \oplus m)\, P(M = m). \end{aligned}$$

Since $P(C = c) = 1/N = P(K = c \oplus m)$, we can multiply by N to obtain

$$P(M = m \mid C = c) = P(M = m),$$

which says that the one-time pad is perfectly secret. □

One of the difficulties with using the one-time pad is that the number of possible keys is as least as large the number of possible messages. Unfortunately, this is required for perfect secrecy:

Proposition 1. *If a cryptosystem is perfectly secret, then the number of possible keys is greater than or equal to the number of possible plaintexts.*

Proof. Let M be the number of possible plaintexts and let N be the number of possible keys. Suppose $M > N$. Let c be a ciphertext. For each key k, decrypt c using the key k. This gives N possible plaintexts, and these are the only plaintexts that can encrypt to c. Since $M > N$, there is some plaintext m that is not a decryption of c. Therefore,

$$P(M = m \mid C = c) = 0 \neq P(M = m).$$

This contradicts the assumption that the system is perfectly secret. Therefore, $M \leq N$. □

2 Indistinguishability and Security

Because the one-time pad is too unwieldy for many applications, pseudorandom generators are often used to generate substitutes for one-time pads. In Sections 2.10 and 2.11, we discuss some possibilities. For the present, we analyze how much such a choice can affect the secrecy of the system.

A pseudorandom key generator produces N possible keys, with each possible key k having probability $P(K = k)$. Usually, the generator takes an input, called a *seed*, and applies some algorithm to produce a key that "looks random." The seed is transmitted to the decrypter of the ciphertext, who uses the seed to produce the key and then decrypt. The seed is significantly shorter than the length of the key, which makes transmission much more efficient, but which means that there are fewer keys than with the one-time pad. Therefore, perfect secrecy is not possible. While the key might have, for example, 1 million bits, the seed could have only 100 bits.

If, for example, the seed had only 20 bits, it would be possible to use all of the seeds to generate a list of all possible keys. Then, given a ciphertext and a

plaintext, it would be easy to see if there is a key that encrypts the plaintext to the ciphertext. But with a seed of 100 bits, it is infeasible to list all seeds and find the corresponding keys. Moreover, with a good pseudorandom key generator it should be difficult to see whether a given key is one that could be produced from some seed. Therefore, a brute force attack is unlikely to be successful.

To evaluate a pseudorandom key generator, Alice (the adversary) and Bob play the following game:

Game 1: *Bob flips a fair coin. If it's Heads, he chooses a number r uniformly randomly from the keyspace. If it's Tails, he chooses a pseudorandom key r. Bob sends r to Alice. Alice guesses whether r was chosen randomly or pseudorandomly.*

Of course, Alice could always guess that it's random, for example, or she could flip her own coin and use that for her guess, In these cases, her probability of guessing correctly is 1/2. But suppose she knows something about the pseudorandom generator (maybe she has analyzed its inner workings, for example). Then she might be able to recognize sometimes that r looks like something the pseudorandom generator could produce (of course,the random generator could also produce it, but with lower probability since it has many more possible outputs). This could occasionally give Alice a slight edge in guessing. So Alice's overall probability of winning could increase slightly. We write

$$P(A \text{ is correct}) = \frac{1}{2} + \epsilon.$$

A good pseudorandom generator should have ϵ very small, no matter what strategy Alice uses.

But will a good pseudorandom key generator work in a one-time pad? Let's say that a one-time pad is implemented with some generator of keys, possibly a truly random generator or maybe a pseudorandom one. Alice (again the adversary) and Bob play the following game:

Game 2: *Alice chooses two messages m_0 and m_1 and gives them to Bob. Bob randomly chooses $b = 0$ or 1. He encrypts m_b to get a ciphertext c, which he gives to Alice. Alice then guesses whether m_0 or m_1 was encrypted.*

By randomly guessing, Alice can be correct with probability 1/2. But suppose Alice knows something about how the keys are chosen. For example, Alice might know that more than half the time Bob's key generator produces binary strings with slightly more 1's than 0's. If Alice computes $m_0 \oplus c$ and $m_1 \oplus c$, and one of these has more 1's then 0's, she can guess that the corresponding message is the one that was used. Of course, this will not always be the correct choice, but the strategy will slightly improve her chance of being correct.

Suppose Bob is using a true random generator for the one-time pad and Alice and Bob play Game 2. Since the one-time pad with a random key generator is

perfectly secret,

$$P(M = m_0 \mid C = c) = P(M = m_0) = 1/2$$
$$P(M = m_1 \mid C = c) = P(M = m_1) = 1/2.$$

Therefore, the two possibilities have the same probability, which means that Alice must do the equivalent of guessing the outcome of a fair coin. That is, the probability that Alice guesses correctly is 1/2.

Now suppose Bob is using a pseudorandom key generator for his one-time pad, and Alice has a friend Charles who can win Game 2 (with Bob's pseudorandom key generation used in the encryption) with probability $\frac{1}{2} + \epsilon$. We claim that Alice can use Charles as a subroutine and thereby win Game 1 with probability $\frac{1}{2} + \frac{1}{2}\epsilon$. They do the following:

Bob flips a fair coin and generates a number r, as in Game 1. He gives r to Alice. Alice wants to guess whether r is random or pseudorandom. She calls up Charles, who chooses messages m_0 and m_1 and gives them to Alice. Alice chooses $b = 0$ or 1 randomly, and encrypts m_b using the key r to obtain the ciphertext $c = m_b \oplus r$, which she sends to Charles. Charles makes his guess for b and succeeds with probability $\frac{1}{2} + \epsilon$.

Alice now uses Charles's guess to finish playing Game 1. If Charles guessed b correctly, she guesses that r was pseudorandom, and if he guessed incorrectly, she guesses that r was random.

There are two ways that Alice wins Game 1. One is when Charles is correct and r is pseudorandom, and the other is when Charles is incorrect and r is random.

The probability that Charles is correct given that r is pseudorandom is $\frac{1}{2}+\epsilon$, by assumption. This means that

$$\begin{aligned} &P(\text{Charles is correct} \cap r \text{ is pseudorandom}) \\ &\quad = P(\text{Charles is correct} \mid r \text{ is pseudorandom})\, P(r \text{ is pseudorandom}) \\ &\quad = (\frac{1}{2} + \epsilon)(1/2). \end{aligned}$$

If r is random, then Alice encrypted m_b with a true one-time pad, so Charles succeeds half the time and fails half the time. Therefore,

$$\begin{aligned} &P(\text{Charles is incorrect} \cap r \text{ is random}) \\ &\quad = P(\text{Charles is incorrect} \mid r \text{ is random})\, P(r \text{ is random}) \\ &\quad = (\frac{1}{2})(1/2). \end{aligned}$$

Putting the previous two calculations together, we see that the probability that Alice wins Game 1 is

$$(\frac{1}{2} + \epsilon)(1/2) + (\frac{1}{2})(1/2) = \frac{1}{2} + \frac{1}{2}\epsilon,$$

as we claimed.

The preceding shows that if we design a good random key generator (that is, $\frac{1}{2}\epsilon$ is very small), then an adversary has only a very slight advantage in using a ciphertext to distinguish between two plaintexts.

Game 2 can be played for an arbitrary cryptosystem. If the probability that Alice wins is 1/2, then the cryptosystem has the property of indistinguishability. This is a very desirable property of a cryptosystem, and a system should not be regarded as secure if it does not have this property, or something very close to it (that is, the probability that Alice wins is $\frac{1}{2} + \epsilon$ for a very small ϵ). This means that in public key systems (see Chapter 6), where anyone can encrypt a message, there must be some randomness added to the encryption protocol. If this is done, two encryptions of a message yield distinct ciphertexts (but the decryptions still produce the original message). Then, in Game 2, an adversary cannot simply encrypt m_0 and m_1 and compare with the ciphertext c in order to win the game.

In the preceding, we showed that the security of a pseudorandom implementation of the one-time pad is reduced to the quality of the pseudorandom generator. This is a taste of the modern area of cryptography that gives what are known as proofs of security by showing that the security of a given cryptographic construction can be reduced to the strength of a reasonably primitive concept such as the quality of a pseudorandom number generator. Unfortunately, there are not good ways to prove that a given pseudorandom number generator is good (this would require solving some major problems in complexity theory), but knowledge of where the security of a system lies is significant progress. For a good introduction to cryptography via the language of computational security and proofs of security, see [Katz and Lindell].

3 Exercises

1. Alice is learning about the shift cipher. She chooses a random 3-letter word (so all 3-letter words in the dictionary have the same probability) and encrypts it using a shift cipher with a randomly chosen key (that is, each possible shift has probability 1/26). Eve intercepts the ciphertext *mxp*.
   (a) Compute $P(M = cat \mid C = mxp)$.
   (b) Use your result from part (a) to show that the shift cipher does not have perfect secrecy (this is also true because there are fewer keys than ciphertexts; see the proposition at the end of the first section).

2. Alice is learning more advanced techniques for the shift cipher. She now chooses a random 5-letter word (so all 5-letter words in the dictionary have the same probability) and encrypts it using a shift cipher with a randomly chosen key (that is, each possible shift has probability 1/26). Eve intercepts the ciphertext *evire*. Show that $P(M = arena \mid C = evire) = 1/2$. (*Hint:* Look at Exercise 1 in Chapter 2.)

3. Alice and Bob play the following game (this is Game 2 of the second sec-

tion). Alice chooses two two-letter words m_0 and m_1 and gives them to Bob. Bob randomly chooses $b = 0$ or 1. He encrypts m_b using a shift cipher (with a randomly chosen shift) to get a ciphertext c, which he gives to Alice. Alice then guesses whether m_0 or m_1 was encrypted.
(a) Alice chooses $m_0 = HI$ and $m_1 = NO$. What is the probability that Alice guesses correctly?
(b) Give a choice of m_0 and m_1 that Alice can make so that she is guaranteed to be able to guess correctly.

4. Bob has a weak pseudorandom generator that produces N different M-bit keys, each with probability $1/N$. Alice and Bob play Game 1. Alice makes a list of the N possible pseudorandom keys. If the number r that Bob gives to her is on this list, she guesses that the number is pseudorandom. If it is not on the list, she guess that it is random.
(a) Show that

$$P(r \text{ is on the list} \mid r \text{ is random}) = \frac{N}{2^M}$$

and

$$P(r \text{ is on the list} \mid r \text{ is pseudorandom}) = 1.$$

(b) Show that

$$P(r \text{ is random} \cap r \text{ is on the list}) = \frac{1}{2}\left(\frac{N}{2^M}\right)$$
$$P(r \text{ is random} \cap r \text{ is not on the list}) = \frac{1}{2}\left(1 - \frac{N}{2^M}\right)$$
$$P(r \text{ is pseudorandom} \cap r \text{ is on the list}) = \frac{1}{2}$$
$$P(r \text{ is pseudorandom} \cap r \text{ is not on the list}) = 0.$$

(c) Show that Alice wins with probability

$$\frac{1}{2}\left(1 - \frac{N}{2^M}\right) + \frac{1}{2}.$$

(d) Show that if $N/2^M = 1 - \epsilon$ then Alice wins with probability $\frac{1}{2} + \frac{1}{2}\epsilon$. (This shows that if the pseudorandom generator misses only a small fraction of the possible keys, then Alice has an advantage in the game, provided that she can make a list of all possible outputs of the generator. Therefore, it is necessary to make N large enough that making such a list is infeasible.)

5. Suppose Alice knows that Bob's pseudorandom key generator has a slight bias and that with probability 51% it produces a key with more 1's than 0's. Alice and Bob play Game 2. Alice chooses messages $m_0 = 000\cdots0$ and $m_1 = 111\cdots1$ to Bob, who randomly chooses $b \in \{0, 1\}$ and encrypts

m_b with a one-time pad using his pseudorandom key generator. He gives the ciphertext $c = m_b \oplus r$ (where r is his pseudorandom key) to Alice. Alice computes $s = c \oplus m_0$. If s has more 1's than 0's, she guesses that $b = 0$. If not, she guesses that $b = 1$. For simplicity in the following, we assume that the message lengths are odd (so there cannot be the same number of 1's and 0's).

(a) Show that exactly one of $m_0 \oplus r$ and $m_1 \oplus r$ has more 1's than 0's.

(b) Show that $P(s \text{ has more 1's} \mid b = 0) = .51$

(c) Show that $P(s \text{ has fewer 1's} \mid b = 1) = .51$

(d) Show that Alice has a probability .51 of winning.