Chapter 4 The One-Time Pad

The one-time pad, which is an unbreakable cryptosystem, was described by Frank Miller in 1882 as a means of encrypting telegrams. It was rediscovered by Gilbert Vernam and Joseph Mauborgne around 1918. In terms of security, it is the best possible system, but implementation makes it unsuitable for most applications.

In this chapter, we introduce the one-time pad and show why a given key should not be used more than once. We then introduce the important concepts of perfect secrecy and ciphertext indistinguishability, topics that have become prominent in cryptography in recent years.

4.1 Binary Numbers and ASCII

In many situations involving computers, it is more natural to represent data as strings of 0s and 1s, rather than as letters and numbers.

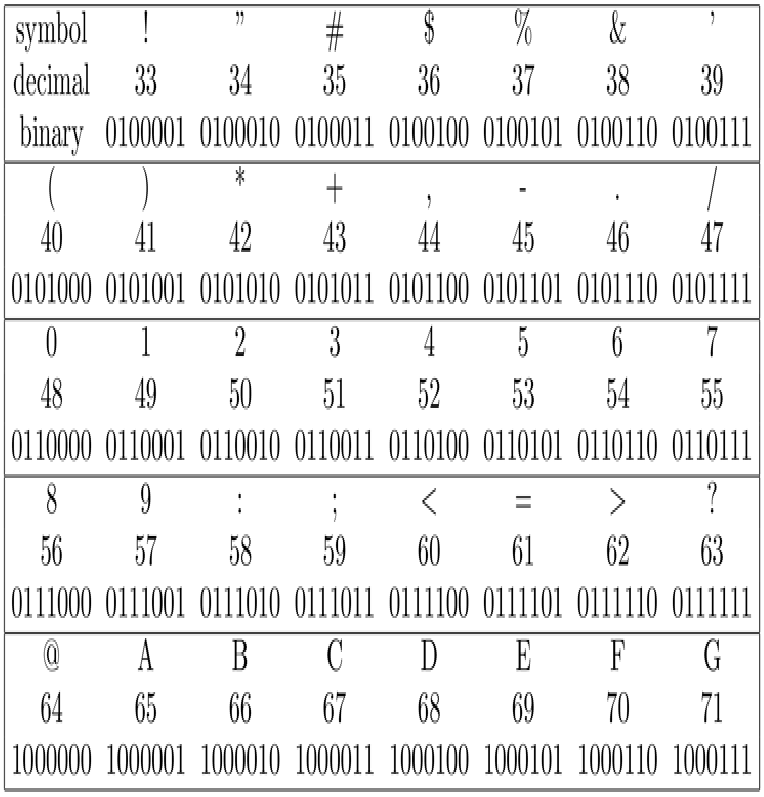
Numbers can be converted to binary (or base 2), if desired, which we'll quickly review. Our standard way of writing numbers is in base 10. For example, 123 means $1 \times 10^2 + 2 \times 10^1 + 3$. Binary uses 2 in place of 10 and needs only the digits 0 and 1. For example, 110101 in binary represents $2^5 + 2^4 + 2^2 + 1$ (which equals 53 in base 10).

Each 0 or 1 is called a **bit**. A representation that takes eight bits is called an eight-bit number, or a **byte**. The largest number that 8 bits can represent is 255, and the largest number that 16 bits can represent is 65535.

Often, we want to deal with more than just numbers. In this case, words, symbols, letters, and numbers are given binary representations. There are many possible ways of doing this. One of the standard ways is called ASCII, which stands for American Standard Code for Information Interchange. Each character is represented using seven bits, allowing for 128 possible characters and symbols to be represented. Eight-bit blocks are common for computers to use, and for this reason, each character is often represented using eight bits. The eighth bit can be used for checking parity to see if an error occurred in transmission, or is often used to extend the list of characters to include symbols such as ü and è .

Table 4.1 gives the ASCII equivalents for some standard symbols.

Table 4.1 ASCII Equivalents of Selected Symbols



symbol	!	”	#	$	%	&	’
decimal	33	34	35	36	37	38	39
binary	0100001	0100010	0100011	0100100	0100101	0100110	0100111
(	)	*	+	,	-	.	/
40	41	42	43	44	45	46	47
0101000	0101001	0101010	0101011	0101100	0101101	0101110	0101111
0	1	2	3	4	5	6	7
48	49	50	51	52	53	54	55
0110000	0110001	0110010	0110011	0110100	0110101	0110110	0110111
8	9	:	;	<	=	>	?
56	57	58	59	60	61	62	63
0111000	0111001	0111010	0111011	0111100	0111101	0111110	0111111
@	A	B	C	D	E	F	G
64	65	66	67	68	69	70	71
1000000	1000001	1000010	1000011	1000100	1000101	1000110	1000111

Table 4.1 Full Alternative Text

4.2 One-Time Pads

Start by representing the message as a sequence of 0s and 1s. This can be accomplished by writing all numbers in binary, for example, or by using ASCII, as discussed in the previous section. But the message could also be a digitalized video or audio signal.

The key is a random sequence of 0s and 1s of the same length as the message. Once a key is used, it is discarded and never used again. The encryption consists of adding the key to the message mod 2, bit by bit. This process is often called **exclusive or**, and is denoted by XOR or $\oplus$. In other words, we use the rules $0 \oplus 0 = 0, 0 \oplus 1 = 1$, $1 \oplus 1 = 0$. For example, if the message is 00101001 and the key is 10101100, we obtain the ciphertext as follows:

$$\begin{array}{rl} (\text{plaintext}) & 00101001 \\ (\text{key}) \oplus & 10101100 \\ (\text{ciphertext}) & 10000101 \end{array}$$

Decryption uses the same key. Simply add the key onto the ciphertext: $10000101 \oplus 10101100 = 00101001$.

A variation is to leave the plaintext as a sequence of letters. The key is then a random sequence of shifts, each one between 0 and 25. Decryption uses the same key, but subtracts instead of adding the shifts.

This encryption method is completely unbreakable for a ciphertext-only attack. For example, suppose the ciphertext is *FIOWPSLQNTISJQL*. The plaintext could be *wewillwinthewar* or it could be *theduckwantsout*. Each one is possible, along with all other messages of the same length. Therefore the ciphertext gives no information about the plaintext (except for its length).

This will be made more precise in Section 4.4 and when we discuss Shannon's theory of entropy in Chapter 20.

If we have a piece of the plaintext, we can find the corresponding piece of the key, but it will tell us nothing about the remainder of the key. In most cases a chosen plaintext or chosen ciphertext attack is not possible. But such an attack would only reveal the part of the key used during the attack, which would not be useful unless this part of the key were to be reused.

How do we implement this system, and where can it be used? The key can be generated in advance. Of course, there is the problem of generating a truly random sequence of 0s and 1s. One way would be to have some people sitting in a room flipping coins, but this would be too slow for most purposes. It is often suggested that we could take a Geiger counter and count how many clicks it makes in a small time period, recording a 0 if this number is even and 1 if it is odd, but care must be taken to avoid biases (see Exercise 12 in Chapter 5). There are other ways that are faster but not quite as random that can be used in practice (see Chapter 5); but it is easy to see that quickly generating a good key is difficult. Once the key is generated, it can be sent by a trusted courier to the recipient. The message can then be sent when needed. It is reported that the "hot line" between Washington, D.C., and Moscow used one-time pads for secure communications between the leaders of the United States and the U.S.S.R. during the Cold War.

A disadvantage of the one-time pad is that it requires a very long key, which is expensive to produce and expensive to transmit. Once the key is used up, it is dangerous to reuse it for a second message; any knowledge of the first message gives knowledge of the second, for example. Therefore, in most situations, various methods are used in which a small input can generate a reasonably random sequence of 0s and 1s, hence an “approximation” to a one-time pad. The amount of information carried by the courier is then several orders of magnitude smaller than the messages that will be sent. Two such methods, which are fast but not highly secure, are described in Chapter 5.

A variation of the one-time pad has been developed by Maurer, Rabin, Ding, and others. Suppose it is possible to have a satellite produce and broadcast several random sequences of bits at a rate fast enough that no computer can store more than a very small fraction of the outputs. Alice wants to send a message to Bob. They use a public key method such as RSA (see Chapter 9) to agree on a method of sampling bits from the random bit streams. Alice and Bob then use these bits to generate a key for a one-time pad. By the time Eve has decrypted the public key transmission, the random bits collected by Alice and Bob have disappeared, so Eve cannot decrypt the message. In fact, since the encryption used a one-time pad, she can never decrypt it, so Alice and Bob have achieved everlasting security for their message. Note that bounded storage is an integral assumption for this procedure. The production and the accurate sampling of the bit streams are also important implementation issues.

4.3 Multiple Use of a One-Time Pad

Alice sends messages to Bob, Carla, and Dante. She encrypts each message with a one-time pad, but she's lazy and uses the same key for each message. In this section, we'll show how Eve can decrypt all three messages.

Suppose the messages are M_1, M_2, M_3 and the key is K. The ciphertexts C_1, C_2, C_3 are computed as $M_i \oplus K = C_i$. Eve computes

$$C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K) = M_1 \oplus M_2.$$

Similarly, she obtains $M_1 \oplus M_3 = C_1 \oplus C_3$ and $M_2 \oplus M_3 = C_2 \oplus C_3$. The key K has disappeared, and Eve's task is to deduce M_1, M_2, M_3 from knowledge of $M_1 \oplus M_2$, $M_1 \oplus M_3$, $M_2 \oplus M_3$. The following example shows some basic ideas of the method.

Let's assume for simplicity that the messages are written in capital letters with spaces but with no other punctuation. The letters are converted to ASCII using

$$A = 1000001,\ B = 1000010,\ \cdots,\ Z = 1011010,\ \text{space } = 0100000$$

(the letters A to Z are the numbers 65 through 90 written in binary, and *space* is 32 in binary; see Table 4.1).

The XORs of the messages are

$M_1 \oplus M_2 =$
0000000 0001100 1100100 0001101 1100001 0011110 1111001 1100001
1100100 0011100 0001001 0001111 0010011 0010111 0000111

$M_1 \oplus M_3 =$
0000001 1100101 1100001 0000101 1100001 0011011 1100101 0010011
1100101 1110011 0010001 0001100 0010110 0000100 0001101

$M_2 \oplus M_3 =$
0000001 1101001 0000101 0001000 0000000 0000101 0011100 1110010
0000001 1101111 0011000 0000011 0000101 0010011 0001010.

Note that the first block of $M_1 \oplus M_2$ is 0000000. This means that the first letter of M_1 is the same as the first letter of M_2. But the observation that makes the biggest difference for us is that "*space*" has a 0 as its leading bit, while all the letters have 1 as the leading bit. Therefore, if the leading bit of an XOR block is 1, then it arises from a letter XORed with "*space*."

For example, the third block of $M_1 \oplus M_2$ is 1100100 and the third block of $M_1 \oplus M_3$ is 1100001. These can happen only from $\backslash\backslash space \oplus 1000100''$ and $\backslash\backslash space \oplus 1000001''$, respectively. It follows easily that M_1 has "*space*" as its third entry, M_2 has $1000100 = H$ as its third entry, and M_3 has A as its third entry. Similarly, we obtain the 2nd, 7th, 8th, and 9th entries of each message.

The 5th entries cause a little more trouble: $M_1 \oplus M_2$ and $M_1 \oplus M_3$ tell us that there is "*space*" XORed with A, and $M_2 \oplus M_3$ tells us that the 5th entries of M_2 and M_3 are equal, but we could have "*space A A*" or "*A space space*." We need more information from surrounding letters to determine which it is.

To proceed, we use the fact that some letters are more common than others, and therefore certain XORs are more likely to arise from these than from others. For example, the block 0010001 is more likely to arise from $E \oplus T = 1000101 \oplus 1010100$ than from $K \oplus Z = 1001011 \oplus 1011010$. The most common letters in English are $E, T, A, O, I, N, S, H, R, D, L$. Make a table of the XORs of each pair of these letters. If an XOR block occurs in this table, we guess that it comes from one of the pairs yielding this block. For example, `0001001` arises from $A \oplus H$ and from $E \oplus L$, so we guess that the 11th block of $M_1 \oplus M_2$ comes from one of these two combinations. This might not be a correct guess, but we expect such guesses to be right often enough to yield a lot of information.

Rather than produce the whole table, we give the blocks that we need and the combinations of these frequent letters that produce them:

0000001 :	$D \oplus E,\ H \oplus I,\ R \oplus S$
0000011 :	$L \oplus O$
0000100 :	$A \oplus E,\ H \oplus L$
0000101 :	$A \oplus D,\ I \oplus L$
0001000 :	$A \oplus I,\ D \oplus L$
0001001 :	$A \oplus H,\ E \oplus L$
0001100 :	$D \oplus H,\ E \oplus I$
0001111 :	$A \oplus N$
0010001 :	$E \oplus T$
0010011 :	$A \oplus R$
0010110 :	$D \oplus R,\ E \oplus S$
0010111 :	$D \oplus S,\ E \oplus R$
0011000 :	$L \oplus T$
0011011 :	$H \oplus S,\ I \oplus R,\ O \oplus T$
0011110 :	$L \oplus R$

Let's look at the next-to-last block of each XOR. In $M_1 \oplus M_2$, we have `0010111`, which could come from $D \oplus S$ or $E \oplus R$. In $M_1 \oplus M_3$, we have `0000100`, which could come from $A \oplus E$ or $H \oplus L$. In $M_2 \oplus M_3$, we have `0010011`, which could come from $A \oplus R$. The only combination consistent with these guesses is E, R, A for the next-to-last letters of M_1, M_2, M_3, respectively. This type of reasoning, combined with the information obtained from the occurrence of spaces, yields the following progress (- indicates a space, * indicates a letter to be determined):

$$\begin{aligned}M_1 &= *\text{E–}**\text{R–A–SE}*\text{RE}*\\ M_2 &= *\text{ID}**\text{LY–DOL}*\text{AR}*\\ M_3 &= *\text{–A}**\text{IERE–T}*\text{DA}*\end{aligned}$$

The first letter of M_3 is a one-letter word. The XOR block `0000001` gives us the possibilities D, E, H, I, R, S, so we guess M_3 starts with I. The XORs tell us that M_1 and M_2 start with H.

Now let's look at the 12th letters. The block `0001111` for $M_1 \oplus M_2$ suggests $A \oplus N$. The block for $M_1 \oplus M_3$ suggests $D \oplus H$ and $E \oplus I$. The block for $M_2 \oplus M_3$ suggests $L \oplus O$. These do not have a common solution, so we know that one of the less common letters occurs in at least one of the messages. But M_2 ends with $DOL{*}AR{*}$, and a good guess is that this should be $DOLLARS$. If so, then the XOR information tells us that M_1 ends in `SECRET`, and M_3 ends in `TODAY`, both of which are words. So this looks like a good guess. Our progress so far is the following:

$$\begin{aligned}M_1 &= \text{HE–}**\text{R–A–SECRET}\\ M_2 &= \text{HID}**\text{LY–DOLLARS}\\ M_3 &= \text{I–A}**\text{IERE–TODAY}\end{aligned}$$

It is time to revisit the 5th letters. We already know that they are "*space A A*" or "*A space space*." The first possibility requires M_1 to have two consecutive one-letter words, which seems unlikely. The second possibility means that M_3 starts with the two words $\text{I–A}*$, so we can guess this is I AM. The XOR information tells us that M_1 and M_2 have H and E, respectively. We now have all the letters:

$$\begin{aligned}M_1 &= \text{HE–HAR–A–SECRET}\\ M_2 &= \text{HIDE–LY–DOLLARS}\\ M_3 &= \text{I–AM–IERE–TODAY}\end{aligned}$$

Something seems to be wrong in the 6th column. These letters were deduced from the assumption that they all were common letters, and this must have been wrong. But at this point, we can make educated guesses. When we change I to H in M_3, and make the corresponding changes in M_1 and M_2 required by the XORs, we get the final answer:

$$\begin{aligned} M_1 &= \text{HE–HAS–A–SECRET} \\ M_2 &= \text{HIDE–MY–DOLLARS} \\ M_3 &= \text{I–AM–HERE–TODAY.} \end{aligned}$$

These techniques can also be used when there are only two messages, but progress is usually slower. More possible combinations must be tried, and more false deductions need to be corrected during the decryption. The process can be automated. See [Dawson-Nielsen].

The use of spaces and the restriction to capital letters made the decryption in the example easier. However, even if spaces are removed and more symbols are used, decryption is usually possible, though much more tedious.

During World War II, problems with production and distribution of one-time pads forced Russian embassies to reuse some keys. This was discovered, and part of the Venona Project by the U.S. Army's Signal Intelligence Service (the predecessor to NSA) was dedicated to deciphering these messages. Information obtained this way revealed several examples of Russian espionage, for example, in the Manhattan Project's development of atomic bombs, in the State Department, and in the White House.

4.4 Perfect Secrecy of the One-Time Pad

Everyone knows that the one-time pad provides perfect secrecy. But what does this mean? In this section, we make this concept precise. Also, we know that it is very difficult in practice to produce a truly random key for a one-time pad. In the next section, we show quantitatively how biases in producing the key affect the security of the encryption.

In Section 20.4, we repeat some of the arguments of the present section and phrase them in terms of entropy and information theory.

The topics of this section and the next are part of the subject known as **Provable Security**. Rather than relying on intuition that a cryptosystem is secure, the goal is to isolate exactly what fundamental problems are the basis for its security. The result of the next section shows that the security of a one-time pad is based on the quality of the random number generator. In Section 10.5, we will show that the security of the ElGamal public key cryptosystem reduces to the difficulty of the Computational Diffie-Hellman problem, one of the fundamental problems related to discrete logarithms. In Section 12.3, we will use the Random Oracle Model to relate the security of a simple cryptosystem to the noninvertibility of a one-way function. Since these fundamental problems have been well studied, it is easier to gauge the security levels of the cryptosystems.

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 **conditional probability of B given A** is

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

(4.1)

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 is nonzero).

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 $P(B \mid A) > P(B)$. (Unfortunately, a widely used high school algebra text published around 2005 gave exactly one example of independent events: A and B.)

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 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 $P(M = m)$. These probably do not all have the same probability. For example, the message *attack at noon* is usually more probable than *two plus two equals seven*. Finally, each possible ciphertext c has a probability $P(C = c)$.

We say that a cryptosystem has **perfect secrecy** if

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

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.

Theorem

If the key is chosen uniformly randomly, then the one-time pad has perfect secrecy.

Proof. We need to show that $P\Big(M = m \mid C = c\Big) = P(M = m)$ 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 $1/N$. Start with any plaintext m. If c is the ciphertext, then the key is $k = m \oplus c$. Therefore, the probability that c is the ciphertext is the probability that $m \oplus c$ is the key, namely $1/N$, since all keys have this probability. Therefore, we have proved that

$$P\Big(C = c \mid M = m\Big) = 1/N$$

for each c and m.

We now combine the contributions from the various possibilities for m. Note that if we sum $P(M = m)$ over all possible m, then

$$\sum_m P(M = m) = 1$$

(4.2)

since this is the probability of the plaintext existing. Similarly, the event $C = c$ can be split into the disjoint sets $\Big(C = c \bigcap M = m\Big)$, which yields

$$\begin{aligned} P\left(C = c\right) &= \sum_m P\left(C = c \cap M = m\right) \\ &= \sum_m P\left(C = c|M = m\right)P\left(M = m\right) \quad (\text{by } (4.1)) \\ &= \sum_m \frac{1}{N} P\left(M = m\right) \\ &= \tfrac{1}{N} \quad (\text{by } (4.2)). \end{aligned}$$

Applying Equation (4.1) twice yields

$$\begin{aligned} P\Big(M = m \mid C = c\Big) P(C = c) &= P\Big(C = c \bigcap M = m\Big) \\ &= P\Big(C = c \mid M = m\Big) P(M = m). \end{aligned}$$

Since we have already proved that $P(C=c) = 1/N = P\Big(C = c \ | \ M = m\Big)$, we can multiply by N to obtain

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

which says that the one-time pad has perfect secrecy.

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

If a cryptosystem has perfect secrecy, 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 has perfect secrecy. Therefore, $M \leq N$.

Example

Suppose a parent goes to the pet store to buy a pet for a child's birthday. The store sells 30 different pets with 3-letter names (ant, bat, cat, dog, eel, elk, ...). The parent chooses a pet at random, encrypts its name with a shift cipher, and sends the ciphertext to let the other parent know what has been bought. The child intercepts the message, which is ZBL. The child hopes the present is a dog. Since DOG is not a shift of ZBL, the child realizes that the conditional probability

$$P(M = dog \mid C = ZBL) = 0$$

and is disappointed. Since $P(M = dog) = 1/30$ (because there are 30 equally likely possibilities), we have $P(M = dog \mid C = ZBL) \neq P(M = dog)$, so there is not perfect secrecy. This is because a given ciphertext has at most 26 possible corresponding plaintexts, so knowledge of the ciphertext restricts the possibilities for the decryption. Then the child realizes that YAK is the only possible shift of ZBL, so $P(M = yak \mid C = ZBL) = 1$. This does not equal $P(M = yak)$, so again we see that we don’t have perfect secrecy. But now the child is happy, being the only one in the neighborhood who will have a yak as a pet.

4.5 Indistinguishability and Security

A basic requirement for a secure cryptosystem is **ciphertext indistinguishability**. This can be described by the following game:

CI Game: *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 guess correctly about 1/2 of the time. If there is no strategy where she guesses correctly significantly more than 1/2 the time, then we say the cryptosystem has the ciphertext indistinguishability property.

For example, the shift cipher does not have this property. Suppose Alice chooses the two messages to be *CAT* and *DOG*. Bob randomly chooses one of them and sends back the ciphertext *PNG*. Alice observes that this cannot be a shift of *DOG* and thus concludes that Bob encrypted *CAT*.

When implemented in the most straightforward fashion, the RSA cryptosystem (see Chapter 9) also does not have the property. Since the encryption method is public, Alice can simply encrypt the two messages and compare with what Bob sends her. However, if Bob pads the messages with random bits before encryption, using a good pseudorandom number generator, then Alice should not be able to guess correctly significantly more than 1/2 the time because she will not know the random bits used in the padding.

The one-time pad where the key is chosen randomly has ciphertext indistinguishability. Because Bob chooses b randomly,

$$P(m_0) = P(m_1) = \frac{1}{2}.$$

From the previous section, we know that

$$P(M = m_0 \mid C = c) = P(M = m_0) = \frac{1}{2}$$

and

$$P(M = m_1 \mid C = c) = P(M = m_1) = \frac{1}{2}.$$

Therefore, when Alice receives c, the two possibilities are equally likely, so the probability she guesses correctly is 1/2.

Because the one-time pad is too unwieldy for many applications, pseudorandom generators are often used to generate substitutes for one-time pads. In Chapter 5, we discuss some possibilities. For the present, we analyze how much such a choice can affect the security of the system.

A pseudorandom key generator produces N possible keys, with each possible key k having a probability $P(K = k)$. Usually, it 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. While the key might have, for example, 1 million bits, the seed could have only 100 bits, which makes transmission much more efficient, but this also means that there are fewer keys than with the one-time pad. Therefore, Proposition 4.4 says that perfect secrecy is not possible.

If the seed had only 20 bits, it would be possible to use all of the seeds to generate 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 2^{100} 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.

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

R Game: *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.

In an extreme case, suppose Alice knows that the pseudorandom number generator always has a 1 at the beginning of its output. The true random number generator will produce such an output with probability 1/2. If Alice sees this initial 1, she guesses that the output is from the pseudorandom generator. And if this 1 is not present, Alice knows that r is random. Therefore, Alice guesses correctly with probability 3/4. (This is Exercise 9.)

We write

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

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 test it with the CI Game. Suppose Charles is using a pseudorandom key generator for his one-time pad and he is going to play the CI game with Alice. Moreover, suppose Alice has a strategy for winning this CI Game with probability $\frac{1}{2} + \epsilon$ (if this happens, then Charles's implementation is not very good). We'll show that, under these assumptions, Alice can play the R game with Bob and win with probability $\frac{1}{2} + \frac{\epsilon}{2}$.

For example, suppose that the pseudorandom number generator is such that Alice has probability at most 0.51 of winning the R Game. Then we must have $\epsilon/2 \leq .01$, so $\epsilon \leq .02$. This means that the probability that Charles wins the CI game is at most 0.52. In this way, we conclude that if the random number generator is good, then its use in a one-time pad is good.

Here is Alice's strategy for the R game. Alice and Bob do the following:

1. Bob flips a fair coin, as in the R game, and gives r to Alice. Alice wants to guess whether r is random or pseudorandom.
2. Alice uses r to play the CI game with Charles.
    1. She calls up Charles, who chooses messages m_0 and m_1 and gives them to Alice.
    2. Alice chooses $b = 0$ or 1 randomly.

    3. She encrypts m_b using the key r to obtain the ciphertext $c = m_b \oplus r$.
    4. She sends c to Charles.
    5. Charles makes his guess for b and succeeds with probability $\frac{1}{2} + \epsilon$.

3. Alice now uses Charles's guess to finish playing the R Game.
4. If Charles guessed b correctly, her guess to Bob is that r was pseudorandom.
5. If Charles guessed incorrectly, her guess to Bob is that r was random.

There are two ways that Alice wins the R Game. 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 when r is pseudorandom is $\frac{1}{2} + \epsilon$, by assumption. This means that

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

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,

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

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

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

as we claimed.

The preceding shows that if we design a good random key generator, then an adversary can gain only a very slight advantage in using a ciphertext to distinguish between two plaintexts. 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 the 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].

4.6 Exercises

1. Alice is learning about the shift cipher. She chooses a random three-letter word (so all three-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*.

    1. Compute $P(M = cat \mid C = mxp)$. (Hint: Can *mxp* shift to *cat*?)

    2. 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 five-letter word (so all five-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. Suppose a message m is chosen randomly from the set of all five-letter English words and is encrypted using an affine cipher mod 26, where the key is chosen randomly from the 312 possible keys. The ciphertext is $HHGZC$. Compute the conditional probability $\mathrm{Prob}(m = HELLO \ \mid \ c = HHGZC)$. Use the result of this computation to determine whether of not affine ciphers have perfect secrecy.

4. Alice is learning about the Vigenère cipher. She chooses a random six-letter word (so all six-letter words in the dictionary have the same probability) and encrypts it using a Vigenère cipher with a randomly chosen key of length 3 (that is, each possible key has probability $1/26^3$). Eve intercepts the ciphertext *eblkfg*.

    1. Compute the conditional probability $P(M = attack \mid C = eblkfg)$.

    2. Use your result from part (a) to show that the Vigenère cipher does not have perfect secrecy.

5. Alice and Bob play the following game (this is the CI Game of Section 4.5). 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.

    1. Alice chooses $m_0 = HI$ and $m_1 = NO$. What is the probability that Alice guesses correctly?

    2. Give a choice of m_0 and m_1 that Alice can make so that she is guaranteed to be able to guess correctly.

6. Bob has a weak pseudorandom generator that produces N different M-bit keys, each with probability $1/N$. Alice and Bob play Game R. 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 chosen pseudorandomly. If it is not on the list, she guess that it is random.

    1. Show that

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

    and

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

    2. Show that

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

    3. Show that Alice wins with probability

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

    4. 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 a 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.)

7. 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 the CI Game. Alice chooses messages $m_0 = 000 \cdots 0$ and $m_1 = 111 \cdots 1$ 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).

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

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

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

    4. Show that Alice has a probability .51 of winning.

8. In the one-time pad, suppose that some plaintexts are more likely than others. Show that the key and the ciphertext are not independent. That is, show that there is a key k and a ciphertext c such that

$$P\left(C = c \bigcap K = k\right) \neq P(C = c)\, P(K = k).$$

    (Hint: The right-hand side of this equation is independent of k and c. What about the left-hand side?)

9. Alice and Bob are playing the R Game. Suppose Alice knows that Bob's pseudorandom number generator always has a 1 at the beginning of its output. If Alice sees this initial 1, she guesses that the output is from the pseudorandom generator. And if this 1 is not present, Alice knows that r is random. Show that Alice guesses correctly with probability 3/4.

10. Suppose Bob uses a pseudorandom number generator to produce a one-time pad, but the generator has a slight bias, so each bit it produces has probability 51% of being 1 and only 49% of being 0. What strategy can Alice use so that she expects to win the CI Game more than half the time?

11. At the end of the semester, the professor randomly chooses and sends one of two possible messages:

$$m_0 = YOUPASSED \quad \text{and} \quad m_1 = YOUFAILED.$$

    To add to the excitement, the professor encrypts the message using one of the following methods:

    1. Shift cipher

2. Vigenère cipher with key length 3

3. One-time pad.

You receive the ciphertext and want to decide whether the professor sent m_0 or m_1. For each method (a), (b), (c), explain how to decide which message was sent or explain why it is impossible to decide. You may assume that you know which method is being used. (For the Vigenère, do not do frequency analysis; the message is too short.)

12. On Groundhog Day, the groundhog randomly chooses and sends one of two possible messages:

$$m_0 = SIXMOREWEEKSOFWINTER$$
$$m_1 = SPRINGARRIVESINMARCH.$$

To add to the mystery, the groundhog encrypts the message using one of the following methods: shift cipher, Vigenère cipher with key length 4 (using four distinct shifts), one-time pad. For each of the following ciphertexts, determine which encryption methods could have produced that ciphertext, and for each of these possible encryption methods, decrypt the message or explain why it is impossible to decrypt.

1. *ABCDEFGHIJKLMNOPQRST*

2. *UKZOQTGYGGMUQHYKPVGT*

3. *UQUMPHDVTJYIUJQQCSFL*.

13. Alice encrypts the messages M_1 and M_2 with the same one-time pad using only capital letters and spaces, as in Section 4.3. Eve knows this, intercepts the ciphertexts C_1 and C_2, and also learns that the decryption of M_1 is *THE LETTER * ON THE MAP GIVES THE LOCATION OF THE TREASURE*. Unfortunately for Eve, she cannot read the missing letter *. However, the 12th group of seven bits in C_1 is 1001101 and the 12th group in C_2 is 0110101. Find the missing letter.