1 One-Time Pad

Communication is the act of conveying information from a sender to a receiver. Secure communication refers to the problem of making the communication unavailable to anyone except the desired receiver. Secure communication is the oldest application of cryptography, and remains the centerpiece of cryptography to this day. After all, the word *cryptography* means "hidden writing" in Greek. So, secure communication is a natural place to start our study of cryptography.

History provides us with roughly 2000 years of attempts to secure sensitive communications in the presence of eavesdroppers. Despite the many brilliant minds that lived during this period of time, almost no useful concepts remain relevant to modern cryptography. In fact, it was not clear how to even *formally define the goal* of secure communication until the 1940s. The two modern definitions in use today were identified only in 1982 and 1990.

The *only* cryptographic method developed before 1900 that has stood the test of time is the **one-time pad**, which appears in some form in essentially every modern encryption scheme. In this chapter, we introduce the one-time pad and discuss its important characteristics. Along the way, we will start to get acclimated to cryptographic security definitions.

1.1 Syntax & Correctness for Encryption

The cryptographic approach to secure communication is a tool known as **encryption.** Before discussing the specifics of one-time pad, we will first define what pieces comprise an *encryption scheme* in general.

Definition 1.1 (Encryption syntax)

*A **symmetric-key encryption (SKE) scheme** consists of the following algorithms:*

- KeyGen*: a randomized algorithm that outputs a **key** $k \in \mathcal{K}$.*
- Enc*: a (possibly randomized) algorithm that takes a key $k \in \mathcal{K}$ and **plaintext** $m \in \mathcal{M}$ as input, and outputs a **ciphertext** $c \in C$.*
- Dec*: a deterministic algorithm that takes a key $k \in \mathcal{K}$ and ciphertext $c \in C$ as input, and outputs a plaintext $m \in \mathcal{M}$.*

*We call $\mathcal{K}$ the **key space**, $\mathcal{M}$ the **message space**, and C the **ciphertext space** of the scheme. When we use a single variable — say, Σ — to refer to the scheme as a whole and distinguish one scheme from another, we write Σ.KeyGen, Σ,Enc, Σ.Dec, $\Sigma.\mathcal{K}$, $\Sigma,\mathcal{M}$, and $\Sigma.C$ to refer to its components.*

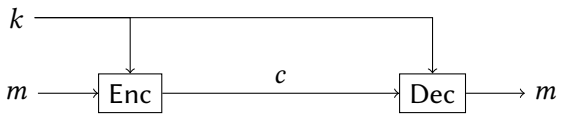
*The scheme satisfies **correctness** if for all $k \in \mathcal{K}$ and all $m \in \mathcal{M}$,*

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1,$$

where the probability is over the random choices (if any) made by Enc.

Encryption addresses the problem of secure communication in a very natural way:

- We imagine a sender and a receiver who wish to communicate. The sender encrypts the desired message/plaintext m using the encryption algorithm Enc and a key k that was chosen according to a the key generation algorithm KeyGen. The result is a ciphertext c, which is sent to the receiver. The receiver can then use the decryption algorithm Dec with the same key k to recover m.



- Because the same key is used for encryption and decryption, we refer to this style of encryption scheme as **symmetric-key**. It's also sometimes refered to as *secret-key* or *private-key* encryption; these terms are somewhat confusing because even other styles of encryption involve things that are called secret/private keys.

- The definition does not specify *how* the sender and receiver come to know a common key k. That problem is considered out of scope for encryption (it is known as *key distribution*). Rather, we are only concerned with what to do once the sender and receiver establish a shared key.

- The definition does not specify *what it means to be secure.* It is a *syntax* definition that specifies only what the *honest* parties (sender and receiver) are supposed to do, whereas security refers to a guarantee that holds in the presence of an adversary. We will actually spend a considerable amount of time in this course building up a good definition of encryption security, step by step.

1.2 One-Time Pad

Now with a clear definition of encryption syntax, we can give the specifics of **one-time pad (OTP)** encryption. The idea of one-time pad had historically been attributed to Gilbert Vernam, a telegraph engineer who patented the scheme in 1919. In fact, one-time pad is sometimes called "Vernam's cipher." However, an earlier description of one-time pad was recently discovered in an 1882 text on telegraph encryption by banker Frank Miller.[1]

Construction 1.2 (One-time pad)

$\mathcal{K} = \{0,1\}^\lambda$
$\mathcal{M} = \{0,1\}^\lambda$
$C = \{0,1\}^\lambda$

KeyGen:
$k \leftarrow \mathcal{K}$
return k

$\mathsf{Enc}(k,m)$:
return $k \oplus m$

$\mathsf{Dec}(k,c)$:
return $k \oplus c$

Here are a few observations about one-time pad to keep in mind:

[1] Steven M. Bellovin: "Frank Miller: Inventor of the One-Time Pad." *Cryptologia* 35 (3), 2011.

- Enc and Dec are essentially the same algorithm. This results in some small level of convenience when implementing one-time pad.
- One-time pad keys, plaintexts, and ciphertexts are all the same length. The encryption schemes we will see later in the course will not have this property.
- Although our encryption syntax allows the Enc algorithm to be randomized, one-time pad does not take advantage of this possibility and has a deterministic Enc algorithm.
- According to this definition, one-time pad keys must be chosen *uniformly* from the set of λ-bit strings.

The correctness property of one-time pad follows from the properties of XOR listed in Section 0. For all $k, m \in \{0,1\}^\lambda$, we have:

$$\begin{aligned}\mathsf{Dec}(k, \mathsf{Enc}(k, m)) &= \mathsf{Dec}(k, k \oplus m) \\ &= k \oplus (k \oplus m) \\ &= (k \oplus k) \oplus m \\ &= 0^\lambda \oplus m \\ &= m.\end{aligned}$$

Example *Consider using OTP encryption to encrypt the 20-bit plaintext m under the 20-bit key k given below:*

$$\begin{array}{cll} & \texttt{00110100110110001111} & (m) \\ \oplus & \texttt{11101010011010001101} & (k) \\ \hline & \texttt{11011110101100000010} & (c) \end{array}$$

The result is ciphertext c. Decrypting c using the same key k results in the original m:

$$\begin{array}{cll} & \texttt{11011110101100000010} & (c) \\ \oplus & \texttt{11101010011010001101} & (k) \\ \hline & \texttt{00110100110110001111} & (m) \end{array}$$

◆

1.3 Properties

A general-purpose security definition for encryption will be discussed in the next chapter. For now, we will show a simple but important property that one-time pad satisfies, and argue that it has some relevance to secure communication.

Imagine an eavesdropper who sees a single one-time pad ciphertext. The following algorithm describes what such an adversary will see if the plaintext happens to be m:

$$\boxed{\begin{array}{l}\underline{\textsc{ctxt}(m \in \{0,1\}^\lambda):} \\ \quad k \leftarrow \{0,1\}^\lambda \\ \quad \text{return } k \oplus m\end{array}}.$$

Since this algorithm involves random choices, its output on a particular m is not a fixed value but a random variable that follows some distribution. The important property of one-time pad has to do with this distribution:

Claim 1.3 *For every $m \in \{0,1\}^\lambda$, the output distribution $\text{CTXT}(m)$ is the uniform distribution over $\{0,1\}^\lambda$.*

Before proving the claim, let's break down why it should have any relevance to security:

- An eavesdropper who intercepts a one-time pad ciphetext sees one sample of the distribution $\text{CTXT}(m)$, where m was the plaintext that was encrypted.
- Claim 1.3 says that sampling from distribution $\text{CTXT}(m)$ is the same as sampling uniformly from $\{0,1\}^\lambda$. The two subroutines merely describe two different methods to sample from the same mathematical distribution.

  We can now say that, by intercepting a single ciphertext, an eavesdropper sees a uniform sample from $\{0,1\}^\lambda$.
- But the uniform distribution does not depend on m at all! Truly, the intercepted ciphertext (at least by itself, without the corresponding key) can carry *no information* about m if the same distribution can be achieved by ignoring m!

At this point you might be suspicious that I'm trying to talk you into a pardox: on one hand, the scheme satisfies the correctness property, so c can always be decrypted to reliably obtain m. On the other hand, I want you to be convinced that c carries *no information whatsoever* about m!

The answer to this riddle is that correctness is defined from the point of view of someone who knows the key k. Claim 1.3 is about the output distribution of the CTXT subroutine, which doesn't contain k (see Exercise 1.6). In short, if you know k, then c can be decrypted to obtain m; if you don't know k, then c might as well be uniformly distributed.

We now prove the claim:

Proof (of Claim 1.3) Fix $m, c \in \{0,1\}^\lambda$; what is the probability that $\text{CTXT}(m)$ produces output c? That event happens only when

$$c = k \oplus m \iff k = m \oplus c.$$

The equivalence follows from the properties of XOR given in Section 0. We have fixed m and c, and hence fixed the right-hand side of the last equation. In other words, after fixing m and c, there is *only one* value of k that encrypts m to c (that value being $m \oplus c$). Since CTXT chooses k uniformly from $\mathcal{K} = \{0,1\}^\lambda$, the probability of choosing the particular value $k = m \oplus c$ is $1/2^\lambda$. ∎

Limitations of One-Time Pad

The keys in one-time pad are as long as the plaintexts they encrypt. This is more or less unavoidable; see Exercise 2.4. Additionally, one-time pad keys can be used to encrypt only one plaintext (hence, "one-time" pad); see Exercise 1.5. Indeed, we can see that the CTXT subroutine in Claim 1.3 provides no way for a caller to guarantee that two plaintexts are

encrypted with the same key, so it is not clear how to use Claim 1.3 to argue about what happens in one-time pad when keys are reused in this way.

Despite these limitations, one-time pad is the conceptual core for *all* forms of encryption we will discuss in this course. The only way to hide information, throughout all of cryptography, is to mask it with a uniformly chosen one-time pad.

Exercises

1.1. The one-time pad encryption of plaintext "`mario`" (written in ASCII) under key k is:

$$\texttt{100001000000011101010100000111000001110 1}.$$

What is the one-time pad encryption of "`luigi`" under the same key?

1.2. Alice is using one-time pad and notices that when her key is the all-zeroes string $k = 0^\lambda$, then $\mathsf{Enc}(k, m) = m$ and her message is sent in the clear! To avoid this problem, she decides to modify KeyGen to choose a key uniformly from $\{0,1\}^\lambda \setminus \{0^\lambda\}$. In this way, her plaintext is never sent in the clear.

Does Claim 1.3 still hold with respect to this modifed one-time pad? Prove or disprove.

1.3. When Alice encrypts the key k itself using one-time pad, the ciphertext will always be the all-zeroes string! So if an eavesdropper sees the all-zeroes ciphertext, she learns that Alice encrypted the key itself. Does this contradict Claim 1.3? Why or why not?

1.4. Describe the flaw in this argument:

> Consider the following attack against one-time pad: upon seeing a ciphertext c, the eavesdropper tries every $k \in \mathcal{K}$ until she has found the one that was used, at which point she outputs the plaintext m. This contradicts the argument in Section 1.3 that the eavesdropper can obtain no information about m by seeing the ciphertext.

1.5. Suppose Alice encrypts two plaintexts m and m' using one-time pad with the same key k. What information about m and m' is leaked to an eavesdropper by doing this (assume the eavesdropper knows that Alice has reused k)? Be as specific as you can!

1.6. Suppose we modify the subroutine discussed in Claim 1.3 so that it also returns k:

$$\boxed{\begin{array}{l}\underline{\textsc{ctxt}'(m \in \{0,1\}^\lambda):} \\ \quad k \leftarrow \{0,1\}^\lambda \\ \quad \text{return } (k, k \oplus m)\end{array}}.$$

Is it still true that for every m, the output of $\textsc{ctxt}'(m)$ is distributed uniformly in $(\{0,1\}^\lambda)^2$?