Introduction and Shannon's one-time pad

Noah Stephens-Davidowitz

May 27, 2023

1 Cryptographic pre-history

The most basic cryptographic task is secure communication. In this task, Alice wants to send a message to Bob—via a letter or over the internet or whatever—without letting an eavesdropper named Eve learn the contents of the letter. We assume that Eve is at least a half-decent eavesdropper, so that she can probably get her hands on the letter itself if she wants to. So, Alice can't just write the message to Bob in plain English and hope to get away with it. (Maybe Alice and Bob are elementary school students passing notes in class and Eve is their teacher, or Alice and Bob are revolutionaries and Eve runs an authoritarian government. Anyway, the question isn't interesting unless we assume that Eve might have the ability to see the letter.) So, how can Alice ensure that Bob is able to decode her message from the letter but that Eve is not?

A famous classical method for encoding a message is known as the Caesar cipher, because Julius Caesar used it. The basic idea is just to replace the letter 'A' with 'B', 'B' with 'C', 'C' with 'D', and so on. (If you're writing a message with a 'Z' in it, you can replace it with an 'A'. Or, you can just avoid discussing zebras and zippers in your messages.) So, this *encryption* procedure takes the *plaintext* message 'ATTACK AT DAWN' to the *ciphertext* 'BUUBDL BU EBXO'. This can be easily *decrypted* if you know how the encryption was performed, but it looks like gibberish to the unitiated. There are a lot of variants of this simple idea, in which we swap letters or move them around in some simple systematic way (uchsay asay igpay atinlay). (When I was a kid, my friend and I had a not very good code in which we fust fort fof fut fan feff fat fe fart fof feach ford.)

These encryption schemes (often called *ciphers* in this context) aren't so bad if Eve is unlikely to put much effort into decoding the message. E.g., they might be fine for kids keeping secrets from their parents. But, if Eve is a government or large corporation, or even just a curious kid with some time on her hands, this won't be very effective. E.g., given the ciphertext above, one might quickly guess that each instance of 'B' in the ciphertext represents the same letter in the original plaintext, and the same holds for 'U', which already gives you a lot of information. (You might also guess that 'B' and 'U' represent relatively common letters, and that they are letters that often appear next to each other.) If you happen to know a little bit of context, like that the first word of the plaintext could be 'ATTACK', then it's pretty easy to see that the plaintext likely has the form 'ATTACK AT _A__', and you can be pretty sure that Alice wasn't telling Bob 'ATTACK AT DUSK'. Of course, even slightly more careful analysis will reveal the entire plaintext.

One can come up with far more complicated encryption schemes that try to avoid the flaws described above. But, by the middle of the 20th century, after nations had spent vast resources building and breaking codes during World War II, it was clear that we needed a more rigorous

understanding of encryption. (And, of course, with the internet, this became a civilian concern as well.)

2 Shannon security

Claude Shannon was the first to give a rigorous notion of the security of an encryption scheme [Sha49], and his definition is now typically called either *perfect security*, *Shannon security*, or *Shannon secrecy*. (Some people use these terms for formally different definitions, and then prove that these definitions are actually equivalent.) He also showed a very simple encryption scheme that satisfies his definition. His scheme is even optimal in a certain precise sense, as we will see. So, in some sense, he solved the problem of encryption completely all the way back in the 1940s. (We'll see that, in another sense, Shannon's scheme is quite unsatisfying. We'll then spend the next eight or nine lectures developing the basic cryptographic machinery necessary to build a better scheme.)

Before we see Shannon's definition of *security*, let's just define what we mean by an encryption scheme, even a possibly insecure one. (It's quite common in cryptography to have two parts to a definition. First, there's typically a notion of *correctness*, which is only concerned with making sure that your construction is useful—that, e.g., Alice and Bob learn what they're expected to learn—and doesn't worry at all about adversaries like Eve. Then, there's usually a notion of *security*, which considers an adversary like Eve.)

Presumably an encryption scheme should specify some way to encode and decode a message. But, we should be very careful. Indeed, the whole point of this exercise is to give a truly rigorous definition. In fact, cryptography—especially cryptography before Shannon—has a history of insufficiently rigorous definitions leading to broken schemes.[1] In fact, it should also probably define what the scheme counts as a "message" and what the scheme counts as an encoding of a message. (E.g., are messages and their encodings both bit strings? Is one written in ASCII?) Etc. Because the words "message" and "encoding" are horribly overloaded in this space, we use the words "plaintext" and "ciphertext" for the "unencoded message" and the "encoded message" respectively.

Most importantly, the *key* part of the definition is the notion of a secret key k, which encapsulates formally the information that Bob knows that will allow him to decrypt the message, which is presumably information that Eve does not have. The key will just kind of come along for the ride in our definition of correctness below, but it will be essential when we consider security.

Definition 2.1 (Encryption scheme)**.** *An* encryption scheme *consists of a plaintext space* $\mathcal{M}$*,*

[1] Even in this course, in which our whole goal is to make our definitions precise and to prove theorems rigorously, I will still not be quite fully formal. I'll try my best to hide the informality from you, and we'll see if you notice. The good news is that, where I am informal, you can rest assured that everything we cover in this course can be made fully formal. However, this is a dangerous precedent. It is quite common to, e.g., find an informal proof that a certain cryptographic scheme is secure only to later realize that it is totally broken. This is *much* more common than in closely related fields, like in the study of algorithms or complexity theory, and it has led to a culture among (good) cryptographers of being extremely careful. Indeed, a major flaw of this course is that I won't show you enough examples of cryptography done *wrong*. We will spend a lot of time very carefully and tediously proving the security of various schemes, but not much time seeing *why* we are so incredibly careful in this space. We are careful because it is very easy to make mistakes in this space, and bad proofs of security can have serious consequences—for our pride and our safety. Every professional cryptographer knows the feeling of having written up a beautiful "proof" that some beautiful construction of theirs is secure only to realize that the proof has some subtle flaw and the construction is actually trivially broken.

a ciphertext space $\mathcal{C}$, and a key space $\mathcal{K}$ together with three (possibly randomized) algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *that follow the satisfying basic properties.*

1. *The* key generation *algorithm* Gen *takes no input and outputs a key $k \in \mathcal{K}$, i.e., $k \leftarrow \mathsf{Gen}()$.*
2. *The* encryption *algorithm* Enc *takes as input a key $k \in \mathcal{K}$ and a plaintext $m \in \mathcal{M}$ and outputs a ciphertext $c \in \mathcal{C}$, i.e., $c \leftarrow \mathsf{Enc}(k, m)$.*
3. *The* decryption *algorithm* Dec *takes as input a key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$and outputs a plaintext $m \in \mathcal{M}$, i.e., $m \leftarrow \mathsf{Dec}(k, c)$.*
4. **Correctness:** *For any $k \in \mathcal{K}$ and $m \in \mathcal{M}$,*

$$\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m \ .$$

(Since the algorithms Enc and Dec are randomized, we should formally say that correctness holds with probability 1, but we adopt the common convention of writing $A = x$ when a random variable A equals x with probability 1.)

The key generation algorithm Gen is only really interesting if it is randomized (since otherwise its output is fixed, and therefore cannot really be viewed as hidden from Eve). Connecting this back to our story before, we think of Alice and Bob getting together before our story begins (say, before class or before the war started) and together running the Gen to obtain a shared secret key $k \in \mathcal{K}$. They can then use this key to communicate. I.e., if Alice wants to send a plaintext message m to Bob, she computes $c \leftarrow \mathsf{Enc}(k, m)$, and sends Bob c (perhaps via mail or over the internet). Bob reads Alice's message by computing $m \leftarrow \mathsf{Dec}(k, c)$. Correctness guarantees that Bob gets the correct message.

The intuitive idea of security should then be clear: no *adversary* Eve who "does not know the secret key k" should be able to "learn anything about the plaintext message m from the ciphertext c." (Here, and throughout this course, I am using *scare quotes* around text that provides some intuition, but is certainly not formal. One should be careful not to take such statements too seriously, and one should *never* view such use statements as justification for security. The intuitive concept that Eve "does not know the secret key k" is really captured formally by the fact that the key is generated randomly using the key-generation algorithm Gen, and the definition below is just one way to formalize the intuitive idea that Eve is unable to "learn anything about the plaintext m from the ciphertext c.") Here is one way to make this formal, which we will call Shannon security.

Definition 2.2 (Shannon security)**.** *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* Shannon secure *if for any probability distribution M over the plaintext space $\mathcal{M}$ and every fixed plaintext $m \in \mathcal{M}$ and ciphertext $c \in \mathcal{C}$,*[2]

$$\Pr_{M}[M = m] = \Pr_{M, k \leftarrow \mathsf{Gen}()}[M = m \mid \mathsf{Enc}(k, M) = c] \ .$$

The adversary Eve doesn't appear directly in this definition, but she is there implicitly. One way to think about this definition is as a sort of Bayesian statement about what Eve knows before she sees c and after she sees c.

[2]Formally, we should restrict our attention to $c \in \mathcal{C}$ such that $\Pr[\mathsf{Enc}(k, M) = c] > 0$, since otherwise the conditional probability is not defined. But, this is too pedantic even for these notes. We therefore adopt the common (silly) convention that equations (or inequalities) are always true when at least one side of the equation involves something that conditions on a zero-probability event.

E.g., the boring constant distribution in which $M = m^*$ with probability one corresponds to the case when Eve knows what plaintext m^* Alice plans to send before she sends it (which is very often the case in practice—e.g., maybe Alice quite kindly says "good morning" to Bob every morning). For this distribution, our security definition does not tell us anything, since the probability on the left and the probability on the right are equal to 1 no matter what the encryption scheme is. This is, of course, perfectly reasonable. There's not much point to using an encryption scheme when Eve already knows Alice's plaintext.

However, if Eve has some uncertainty about which plaintext Alice wishes to send, then we can capture this using a non-constant probability distribution M that represents Eve's knowledge. For example, maybe $M \in \{$'ATTACK AT DAWN', 'ATTACK AT DUSK'$\}$ is uniform, which corresponds to the case in which Alice is equally likely to send either of these messages, as far as Eve is concerned. (Going forward, we will use the notation $M \sim \{$'ATTACK AT DAWN', 'ATTACK AT DUSK'$\}$ for the uniform distribution.) If a scheme is Shannon secure, then this probability distribution remains unchanged after seeing the ciphertext $\mathsf{Enc}(k, M)$, so that in a very precise sense Eve knows exactly as much about the plaintext *after* seeing the ciphertext as she did before. So, she has learned nothing from the ciphertext.

More generally, this definition exactly formally captures the idea that "Eve does not gain any information about the plaintext m from the ciphertext c."

Here's another definition that is a bit easier to understand—at least for me. (Spoiler: we're going to prove that these definitions are equivalent.)

Definition 2.3 (Perfect indistinguishability)**.** *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* perfectly indistinguishable *if for any two messages* $m_0, m_1 \in \mathcal{M}$ *and any ciphertext* $c \in \mathcal{C}$*,*

$$\Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_0) = c] = \Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_1) = c] \ .$$

In words, the above definition requires that any *two* plaintexts yield the same distribution on ciphertexts. Intuitively, this captures the idea that "the ciphertext is useless for Eve in the task of distinguishing between two plaintexts" ('ATTACK' and 'RETREAT', say).[3]

We can now prove the equivalence of these definitions. This allows us to work with whichever definition we like in different contexts. This also suggests that we have the "right" definition(s). At the very least, there's no need to argue about which of the two definitions is better, since they are equivalent. (We will see below, however, that these definitions are in some sense far too strong.)

Theorem 2.4. *An encryption scheme is Shannon secure if and only if it is perfectly indistinguishable.*

Proof. The proof is straightforward, and is essentially just a single application of Bayes' theorem. But, since this is the first lecture, we will do it very carefully.

First, we show that any Shannon secure scheme is perfectly indistinguishable, i.e., we assume Shannon security and show that for any distinct two plaintexts $m_0, m_1 \in \mathcal{M}$ and ciphertext $c \in \mathcal{C}$, $\Pr[\mathsf{Enc}(k, m_0) = c] = \Pr[\mathsf{Enc}(k, m_1) = c]$. To that end, let M be the distribution defined by

[3] Whenever you see a definition like this, it is *always* a good idea to play with it a bit to understand it better. E.g., here are some basic questions to ask yourself that happen to have boring answers in this case: (1) What does the definition say when $m_1 = m_2$? (2) What if c is not in the image of Enc? One can ask more interesting questions as well. E.g., what happens if we replace m_0 and m_1 by random variables M_0 and M_1 (and take the probability over both k and M_b)?

$\Pr[M = m_0] = \Pr[M = m_1] = 1/2$, i.e., the uniform distribution on $\{m_0, m_1\}$. Then, by Shannon security, for any $b \in \{0, 1\}$, we have

$$1/2 = \Pr_M[M = m_b] = \Pr_{M,k \leftarrow \mathsf{Gen}()}[M = m_b \mid \mathsf{Enc}(k, M) = c] \ .$$

Applying Bayes' theorem yields

$$\Pr[M = m_b \mid \mathsf{Enc}(k, M) = c] = \frac{\Pr[M = m_b] \Pr[\mathsf{Enc}(k, m_b) = c]}{\Pr[\mathsf{Enc}(k, M) = c]} = \frac{\Pr[\mathsf{Enc}(k, m_b) = c]}{2 \Pr[\mathsf{Enc}(k, M) = c]} \ .$$

Putting these two equalities together, we see that

$$1/2 = \frac{\Pr[\mathsf{Enc}(k, m_b) = c]}{2 \Pr[\mathsf{Enc}(k, m) = c]} \ ,$$

which of course implies that

$$\Pr[\mathsf{Enc}(k, m_b) = c] = \Pr[\mathsf{Enc}(k, M) = c] \ ,$$

and in particular that $\Pr[\mathsf{Enc}(k, m_0) = c] = \Pr[\mathsf{Enc}(k, m_1) = c]$, as needed.

Now, suppose that our encryption scheme is perfectly indistinguishable, and let M be a distribution over the plaintext space with $\Pr[M = m_0] > 0$ and $c \in \mathcal{C}$. We have

$$\Pr_{M,k \leftarrow \mathsf{Gen}()}[M = m_0 \mid \mathsf{Enc}(k, M) = c] = \frac{\Pr_M[M = m_0] \cdot \Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_0) = c]}{\Pr_{M,k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, M) = c]} \ .$$

But, by perfect indistinguishability, we know that

$$\begin{aligned} \Pr[\mathsf{Enc}(k, M) = c] &= \sum_{m_1 \in \mathcal{M}} \Pr_M[M = m_1] \cdot \Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_1) = c] \\ &= \sum_{m_1 \in \mathcal{M}} \Pr_M[M = m_1] \cdot \Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_0) = c] \\ &= \Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_0) = c] \ , \end{aligned}$$

where the last equality just uses the fact that $\sum_{m_1 \in \mathcal{M}} \Pr[M = m_1] = 1$. Plugging this in to the above, we see that

$$\Pr[M = m_0 \mid \mathsf{Enc}(k, M) = c] = \Pr[M = m_0] \ ,$$

as needed. □

2.1 What about Eve?

Here, we quickly present yet another equivalent definition of security for an encryption scheme. This definition is not very important and rather unnatural in this context. However, it has the benefit that it directly refers to the adversary Eve, so that it makes the relationship to the original problem that we were trying to solve (letting Alice and Bob communicate without Eve learning the contents of their communication) explicit. It is also much more similar to most of the other definitions that we will see in this course.

To that end, we imagine Eve playing a game in which she is given a ciphertext, and her goal is to determine whether it is an encryption of m_0 or an encryption of m_1.

Definition 2.5 (Perfect security against an adversary)**.** *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* perfectly secure against an adversary *if for any adversary* $\mathcal{E} : \mathcal{C} \to \{0,1\}$ *and any pair of messages* $m_0, m_1 \in \mathcal{M}$*,*

$$\Pr_{b\sim\{0,1\},\ k\leftarrow\mathsf{Gen}()}[\mathcal{E}(\mathsf{Enc}(k, m_b)) = b] = 1/2\ .$$

Here, $\mathcal{E}$ is any function at all that maps ciphertexts to bits. (We could be more general here and allow for the possibility that Eve is randomized, but we will not bother with this.)

In words, imagine Alice flipping a coin $b \sim \{0,1\}$ and using this to decide which message m_b to send. An encryption scheme is secure under this definition if no procedure $\mathcal{E}$ used by Eve will allow her to guess Alice's coinflip better than a random guess, even if she is given $\mathsf{Enc}(k, m_b)$. (We often do not distinguish between "Eve's algorithm $\mathcal{E}$" and Eve herself. Instead, we just refer to $\mathcal{E}$ as Eve.)

It is a good exercise to convince yourself that this definition is equivalent to perfect indistinguishability (and thus to Shannon security). Notice that perfect indistinguishability is actually a special case of the above definition in which we take $\mathcal{E}$ to be the particular function $\mathcal{E}^*$ that outputs 0 on input c if $\Pr_{k\leftarrow\mathsf{Gen}()}[\mathsf{Enc}(k, m_0) = c] \geq \Pr_{k\leftarrow\mathsf{Gen}()}[\mathsf{Enc}(k, m_1) = c]$, and otherwise outputs 1. So, the equivalence of these two definitions essentially boils down to arguing that, if $\mathcal{E}^*$ has no advantage in guessing b, then no adversary $\mathcal{E}$ does.

This is also a good opportunity to try modifying a definition to see what happens. It turns out that this definition is quite robust to changes. For example, we get an equivalent definition if we replace $\Pr[\cdots] = 1/2$ by $\Pr[\cdots] \leq 1/2$; if we give $\mathcal{E}$ m_0 and m_1 as input in addition to an encryption of m_b; if we use random variables $M_0, M_1 \in \mathcal{M}$ instead of fixed messages m_0, m_1; if we use the analogous definition for a triple of messages m_0, m_1, m_2; if b is chosen from a biased coin (and the probability $1/2$ is updated appropriately); etc. Less obviously, the definition remains the same even if we bound the computational power of $\mathcal{E}$. (Specifically, the definition remains unchanged if we restrict our attention to functions $\mathcal{E}$ that are computable with "just a bit more computational power than what is needed to run the Gen algorithm and compute $\mathsf{Enc}(k, m_b)$." See if you can figure out why!)

2.2 The key is key

A crucial aspect of the above definitions is that they formalize the notion of a secret key. Specifically, in our story about Alice, Bob, and Eve, the secret key is something that Alice and Bob know but Eve does not. Clearly, Bob must know *something* that Eve does not, since Bob has enough knowledge to decrypt the ciphertext sent to him by Alice, and we certainly don't want Eve to be able to do this as well.

When Alice and Bob use informal encryption schemes like the Ceasar cipher, they are implicitly assuming that Eve does not know *something* that Bob knows, such as the cipher itself. Such an assumption is referred to as "security by obscurity"—i.e., security under the assumption that the adversary "doesn't know what you're doing."

In contrast, our new formal definitions are very clear about what Eve does not know: the secret key k. To make this precise, we don't just say that "k is something that Eve does not know" (so that k could be a proof of the Riemann hypothesis or the password to your email account or your favorite color or whatever). Instead, we explicitly say that the key k is generated in a specific way—by the randomized algorithm Gen—and we formulate an experiment in which the secret key is a random variable. E.g., if in the definition of perfect security against an adversary, we gave $\mathcal{E}$

the key k as input, the definition would obviously be unachievable (at least for correct encryption schemes).

On the other hand, nothing other than the secret key is obscured here. In our story, Eve may know the algorithms Gen, Enc, and Dec. She may know the potential messages that we might send (e.g., the distribution M or the pair of messages m, m', depending on the definition). This notion is captured implicitly by our definitions, which apply to fixed schemes $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and quantify over all messages m, m' or distributions of messages M or algorithms $\mathcal{E}$. (Make sure you understand this. It's *very* common for students to get confused about "whether Eve knows the algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$." The short answer is that she does in fact know these algorithms. The more precise answer is that these are fixed algorithms, and we are quantifying over all possible adversaries $\mathcal{E}$, including adversaries $\mathcal{E}$ that, e.g., run Enc as a subprocedure.)

Remark (Does Alice need the key?)**.** *The observant reader might have noticed that the above argument only shows that* Bob *must know something that Eve does not, since Bob must know how to* decrypt *a ciphertext. On the other hand, Alice only needs to know how to* encrypt *a plaintext, and it is not immediately clear whether giving this ability to Eve would be a problem. In particular, maybe there exist encryption schemes in which* anyone *can encrypt a message (without knowledge of any secret information), but only Bob can decrypt? This (crazy!) idea is what led to public-key encryption, which we will cover in detail later in the course. A public-key encryption scheme does not require Alice and Bob to somehow share a secret before communicating, which is why most internet traffic relies on public-key encryption.*

3 The one-time pad

So, we have a very nice, strong definition. But, it's useless if we can't find an encryption scheme satisfying it. Fortunately, Shannon discovered such a scheme: the *one-time pad*. It is quite simple and elegant.

In this scheme, the plaintext space, key space, and ciphertext space are all $\{0,1\}^n$ for some integer n. (We simply choose n large enough to accomodate the plaintexts that we'd like to send.) The key-generation algorithm Gen simply samples a uniformly random bit string for the secret key, $k \sim \{0,1\}^n$. The encryption algorithm then takes the bit-wise XOR of the key with the plaintext $c = \mathsf{Enc}(k, m) := k \oplus m$. (E.g., $0010 \oplus 1110 = 1100$.) And, of course, decryption simply undoes this operation, which in this case turns out to be the exact same operation $m = \mathsf{Dec}(k, c) := k \oplus c$ since XORing with the same string twice gets you back where you started. A bit more formally, we rely for correctness on the simple fact that $k \oplus (k \oplus m) = m$ for any $k, m \in \{0,1\}^n$.

The one-time pad is very elegant and simple, and *very* efficient as well. It's also very easy to prove that it's perfectly indistinguishable, which immediately implies that it is also Shannon secret (since we proved that the two definitions are equivalent).

Theorem 3.1. *The one-time pad is perfectly indistinguishable.*

Proof. For two plaintexts $m_0, m_1 \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^n$, notice that there are *unique* keys $k_0 := m_0 \oplus c$ and $k_1 := m_1 \oplus c$ satisfying $\mathsf{Enc}(k_b, m_b) = c$. Therefore,

$$\Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_b) = c] = \Pr[k = k_b] = 2^{-n} ,$$

which is independent of b. So, clearly $\Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_0) = c] = \Pr_{k \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k, m_1) = c]$, as needed. □

Corollary 3.2. *The one-time pad is Shannon secure.*

4 The one-time pad can only be used one time!

The one-time pad is used in many applications practice—e.g., by intelligence agencies.[4]

However, the scheme is not useful for most practical applications. For example, suppose Alice and Bob share, say, 128 uniformly random bits. Then they can only use this key to send a 128-bit plaintext. If Alice wants to send Bob a 3 GB video, then they need to exchange a 3 GB key!

A closely related issue is that Alice and Bob cannot reuse their key. E.g., if Alice sends Bob $c_1 = m_1 \oplus k$ and later sends him $c_2 = m_2 \oplus k$, then Eve can trivially learn $m_1 \oplus m_2 = c_1 \oplus c_2$. In particular, Eve can determine whether m_1 and m_2 are the same message or different, or whether their first bit is the same or different. That is really bad! This is why it is called the *one-time* pad. You should only use each key *once* when using the one-time pad.

This next theorem (also proven by Shannon) shows that the one-time pad is essentially the best that we can do. In particular, if we want perfect indistinguishability when sending a 3 GB plaintext, we need a 3 GB key! (Notice that this implies that we would need a fresh 3 GB key to encrypt another message.)

Theorem 4.1. *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a perfectly indistinguishable (and correct) encryption scheme, then* $|\mathcal{K}| \geq |\mathcal{M}|$.

The intuitive idea of the proof is as follows. "For every plaintext, there must be at least one key that maps it to every (valid) ciphertext— otherwise the scheme is not perfectly indistinguishable. Since two distinct plaintexts cannot map to the same ciphertext under the same key (because then we could not possibly have correctness), we must have at least as many keys as ciphertexts. Since there must be at least one distinct ciphertext for each plaintext, this implies that we must have as many keys as plaintexts." This argument actually goes through when Enc is deterministic. But, if Enc is randomized, then it gets tricky (because there could be many ciphertexts that correspond to the same key k and plaintext m, which means there could easily be more valid ciphertexts than keys). The solution is to study Dec instead of Enc, since the output behavior of Dec is fixed by the assumption that the scheme is correct. (Confession: After struggling a bit to make the above intuition go through, I had to look up this proof.)

Proof. Fix any $c \in \mathcal{C}$ such that there exists an m and k with $\Pr[\mathsf{Enc}(k, m) = c] > 0$. Let

$$\mathcal{M}_c := \{m' \in \mathcal{M} \ : \ \exists k' \in \mathcal{K}, \ \mathsf{Dec}(k', c) = m'\}$$

be the set of all plaintexts that *can* be the result of decrypting c. Notice that $|\mathcal{M}_c| \leq |\mathcal{K}|$.[5]

Furthermore, perfect indistinguishability implies that

$$\Pr_{k' \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k', m') = c] = \Pr_{k' \leftarrow \mathsf{Gen}()}[\mathsf{Enc}(k', m) = c] > 0$$

[4] If you've ever seen the TV show The Americans, which is about Soviet spies in America, you'll notice that they spend a lot of time carefully encrypting and decrypting messages using the one-time pad. (It's what they're doing when they scribble a bunch of letters in notebooks in the laundry room.)

[5] Even here, I am cheating slightly. A fully rigorous proof would worry about the possibility that a randomized decryption algorithm sometimes outputs different plaintexts on input (k', c). Correctness rules this out for valid ciphertexts (though not for invalid ciphertexts, for which $\Pr[\mathsf{Enc}(k, m) = c] = 0$ for all (k, m)), but one must argue this carefully.

for *all* plaintexts $m' \in \mathcal{M}$. By correctness, this implies that there exists some k' with $\mathsf{Dec}(k', c) = m'$ for every plaintext $m' \in \mathcal{M}$, so that $\mathcal{M}_c = \mathcal{M}$, and the result follows. □

4.1 What about Eve (again)?

The proof above says that we cannot obtain a *very* strong notion of security unless $|\mathcal{K}| \geq |\mathcal{M}|$. This in particular implies that, if we wish to use the scheme to send many short messages, there is a restriction on how many short messages we can send (since we can think of many short messages as one very long message). This is not acceptable. (It essentially means that for *each* message Alice and Bob wish to send securely, they must somehow find a way to previously securely exchange a *fresh* random key of the same length.)

So, we want to weaken our security definition in order to avoid this issue. To that end, let's see how the above impossibility proof relies on the strength of the definition in order to see how we might weaken the definition to avoid the impossibility.

To do so, let's suppose that we have some encryption scheme for which $|\mathcal{K}| < |\mathcal{M}|$ and try to understand what the above proof tells us about Eve's ability to break this scheme. In fact, the proof quite explicitly suggests a way for Eve to attack the scheme. (Remember that Eve's goal is to take as input a ciphertext c and to guess whether it is an encryption of m_0 or m_1, with success probability better than 1/2.) Given some ciphertext $c \leftarrow \mathsf{Enc}(k, m_b)$, Eve can compute the set $\mathcal{M}_c := \{\mathsf{Dec}(k', c) \ : \ k' \in \mathcal{K}\}$. If $m_0 \in \mathcal{M}_c$ but $m_1 \notin \mathcal{M}_c$, then Eve can confidently output 0—i.e., she can confidently declare that c is an encryption of m_0, not of m_1, and therefore $b = 0$. Similar, if $m_1 \in \mathcal{M}_c$ but $m_0 \notin \mathcal{M}_c$, Eve can confidently output 1. If $m_0, m_1 \in \mathcal{M}_c$, then Eve can answer arbitrarily—e.g., always outputting 0, or flipping a coin and outputting the result. The above proof shows that, for at least one pair of messages $m_0, m_1 \in \mathcal{M}$, there is a non-zero probability $p > 0$ that one of the plaintexts will not lie in $\mathcal{M}_c$, in which case Eve will succeed with probability at least $(1 + p)/2 > 1/2$.

But, if Eve is meant to represent some entity in the real world—a person, or a computer, or a nation-state, or even an unimaginably advanced extraterrestrial civilization that has colonized an entire galaxy—then computing $\mathcal{M}_c$ might be a bit... challenging. Typically, we will have $|\mathcal{M}_c| \approx |\mathcal{K}|$, so that if our key is just, say, 256 bits long, then $|\mathcal{M}_c| \approx 2^{256} \approx 10^{77}$. This is significantly more than number of elementary particles contained in the entire Milky Way galaxy (ignoring dark matter and dark energy). So, if Eve wanted to write $\mathcal{M}_c$ down, she would need a lot of ink—so much that she'd probably have to borrow some ink from some neighboring galaxies! Of course, Eve does not actually need to write down $\mathcal{M}_c$ in order to perform the above attack. She just needs to iterate through it, which would take about 10^{30} years, even if she could try one message in a Planck time (which by some definition is the "smallest possible unit of physical time").[6]

This suggests that we should not model Eve using an *arbitrary* function mapping ciphertexts to bits (i.e., an arbitrary entity that "guesses" whether a ciphertext is an encryption of m_0 or m_1

[6] Notice that we have *not* proven that this is the *only* way to attack a particular encryption scheme with $|\mathcal{K}| < |\mathcal{M}|$. There can certainly be other ways. For example, consider the variant of the one-time pad in which we fix the first bit of the key to be zero, which has $|\mathcal{K}| = |\mathcal{M}|/2$. This can be trivially broken if m_0 and m_1 have different first bits. The adversary certainly doesn't need to enumerate through all possible keys to break this scheme—she only needs to look at the first bit of the ciphertext to figure out what the first bit of the plaintext is. More generally, one must always worry that our adversary is far more clever than we are, and one must carefully avoid *ever* assuming that "there is only one way to break a cryptographic scheme." Here, I am merely pointing out that the specific attack that is suggested by our impossibility proof is not efficient, which is very very different from saying that there is no efficient attack.

using whatever procedure she likes). Instead, we should model Eve as a *computationally bounded adversary*. We'll restrict Eve to have a gigantic but finite amount of computational power. I.e., we'll simply accept that someone with a computer the size of the galaxy might be able to figure out what Alice's plaintext is after computing for 10^{30} year. That's a sacrifice we're simply willing to make.

Another issue with the above attack is that, even if the adversary were powerful enough to carry it out, it might only succeed with vanishingly small probability. In particular, while we know that there exists some ciphertext $c \in \mathcal{C}$ such that, say, $m_0 \in \mathcal{M}_c$ but $m_1 \notin \mathcal{M}_c$, this choice of c might be exceedingly rare. For example, consider the variant of the one-time pad in which the key space is $\mathcal{K} = \{0,1\}^n \setminus (1\ldots,1)$, i.e., all strings except for the all-ones string. This scheme has $|\mathcal{K}| = |\mathcal{M}| - 1 < |\mathcal{M}|$. But, for any plaintext $m_1 \in \{0,1\}^n$, there is a single fixed ciphertext $c := m_1 \oplus (1,\ldots,1)$ such that $m_1 \notin \mathcal{M}_c$. So, for this scheme, the above attack will guess the correct value of b with probability $1/2+2^{-n}$. Again, if for example $n = 256$, then it's hard to imagine caring about the distinction between probability $1/2$ and probability $1/2+2^{-n}$. (Indeed, in anything close to a real-world situation, there will be many ways that things can go catastrophically wrong that have probabilities far higher than 2^{-256}. E.g., maybe there's a $1/2^{40} \approx 1/10^{12}$ chance that Alice accidentally uses the wrong secret key k; or that Bob misinterprets Alice's plaintext and attacks at the wrong time; or that Alice accidentally forgets to encrypt the plaintext at all and just sends it in the clear.)

So, though we've just proven a very strong impossibility result, the above discussion gives us a small glimmer of hope. Maybe we can't have *perfectly* secure encryption schemes with reusable keys. But, perhaps we can settle for such encryption schemes in which, e.g., any adversary with all of the resources of our galaxy would need at least 10^{30} years in order to guess Alice's plaintext with probability larger than $1/2 + 2^{-256}$. This turns out to be a *very* good idea.

References

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4), 1949.