The Humble (But Powerful) One Time Pad

Author	josh
Date	2022-08-07

Overview

One of the cornerstones of classic cryptography is the one-time pad. This type of encryption is powerful, and information-theoretically secure if done correctly! However, it is also quite impractical for most applications. Let's discuss how one-time pads work, why they are so powerful, and why they are also not the most practical cipher to use.

One-Time Workings

Properties of the One-Time Pad

This cipher is quite simple in its construction and use, and its simplicity is one of its strengths. First, we start with our *message* or *plaintext*. Our plaintext has a *length*. Let's say for example we wish to encrypt the message "cryptography is fun". This message is 19 characters in length, including the spaces.

The first important characteristic of the OTP is that the encryption *key* must be exactly equal in length to the *plaintext*. So since our plaintext is 19 bytes in length (one byte per character), our secret key must also be 19 bytes. Not one byte less or one byte more – exactly 19 bytes.

The second property of the OTP is that the key must be *truly random* in order to maintain perfect secrecy (more on that definition later). The key cannot be *pseudorandom* (just appear random) – we must use a truly random source such as radioactive emissions. There are many sources of randomness that are *cryptographically secure* for most purposes – CSPRNGS, fair dice rolls, etc. but are still, technically, not secure enough for a perfect-secrecy one-time pad.

The last property of the OTP that is critical is that the key be *only used once* – hence the name one-time pad. Reusing the key breaks the encryption scheme.

Constructing the OTP

If we have a certain *plaintext,* our truly random *key,* and a bit of software handy. One Time Pads are fairly straightforward to execute on computing devices – they make use of a simple XOR (exclusive or) logical operator.

Let's say for example the first character of our *plaintext* is "c" and we have the first byte of our *key* data – we can now *encrypt* to get our *ciphertext*.

01100011 (‘c’ in binary form) – our *plaintext* XOR (^)
00011001 (a random byte) – our *key*

_____
01111010 (plaintext ^ key) – our *ciphertext*

By XORing each *plaintext* bit with each *key* bit we get a resulting *ciphertext* bit.

To *decrypt*, we simply do the operation in reverse:

01111010 (plaintext ^ key) – our *ciphertext* XOR (^)
00011001 (a random byte) – our *key*

_____
01100011 (‘c’ in binary form) – our *plaintext*

Perfect Secrecy

In *theory*, a properly-constructed OTP is impossible to crack. This is due to the fact that all possible *keys* of the same size as the *plaintext* give all possible *ciphertexts*. For example, let’s say our “cryptography is fun” message XORed with some theoretical key encrypts to “zxveamlophtieombupi”. This certainly looks like an encrypted ciphertext to a human.

However, since all 19 byte keys are valid in this context, there also exists a key that encrypts “cryptography is fun” into “cryptography stinks!” and a key that encrypts it to “bitcoin donkey mud.”. If the key is truly random, there is no way for an attacker to try and *brute-force* a key that gives a sensible *decrypted plaintext.* All possible plaintexts exist, so an attacker can never tell what the real *plaintext* is.

Pad Pitfalls

Breaking the Pad

There are several ways in which an OTP can be broken. First off, is by reusing a key. There is an attack where if one reuses the *key* for two encryptions, we get the following fact:

ciphertext1 ^ ciphertext2 = plaintext1 ^ plaintext2

By retrieving the XORed together values of the first and second *plaintext*, one can do some pattern recognition to gather information about the plaintexts. This is, of course, insecure as we do not want our encryption scheme to leak any information about the plaintext other than its *length.* This is why truly random keys are so critical – they prevent attacks that rely on pattern recognition.

Impracticality

The second major issue is key distribution – it is difficult to generate, store, and share equal length *keys* needed for encrypting many applicable *plaintexts*. It’s totally reasonable to store and share a key to encrypt “cryptography is fun”. But imagine wanting to encrypt the entire contents

of a hard drive. You would have to have another equally or larger sized hard drive just to store the key! This is wildly impractical for most cases, especially with much of the data we enjoy working with today.

Not only that, but the contents of the drive must stay the same – because if you reuse that key for modified hard drive contents, you run into the security issue described above in “Breaking the Pad”.

One-Time Pads – Neat, but Impractical

One-time pads are a fascinating part of cryptography, a foundational construct. They have had their historical uses in intelligence and are still sometimes used in certain contexts. Their simplicity is interesting and straightforward to study for educational purposes. Unfortunately, they are just impractical for day-to-day use. Instead, we have secure ciphers such as AES for our symmetric encryption needs. One-time pads remain a neat concept to study!