towardsdatascience.com

Cryptography Crash Course for the Intimidated - Towards Data Science

Rebecca Weng

8–10 minutes

Conceptual overview and suggestions for reading/watching

tds

Rotor Cipher Machine via Pixabay

In my last post, I talked about the importance of being mindful when you handle data. Who has access? How can you protect the information? How are some laws changing? What is and isn't covered in academia vs. industry.

For this post, I decided to give myself a quick crash course in cryptography. The concept of cryptography can sound really intimidating, and the security of data and the secure transfer of data is certainly something to take seriously. But we create data every day. I think it's important for anyone these days to have a basic grasp on what cryptography is.

Objectives

- Try to demystify cryptography to a certain extent.
- Explain some of the basic concepts that underlie cryptography.

- Explain a couple of examples in which cryptography is used in our everyday lives.
- Share resources!

Basic Concepts

To start understanding cryptography, I went for the basics. Here are some basic concepts that come up over and over again when you look up cryptography online. I have linked a YouTube video below, if you are more of a visual person.

First, what is cryptography? **Crytography** is a field of study focused on communication and data storage that is protected from an unwanted third-party. For example, before cellphones, two kids might try to pass a physical note to each other in the middle of class. They don't want anyone else (a third-party), such as a teacher to read the note. So they might want to figure out a way to write in a made up language or to scramble the message (**encrypt** the message) so that even if the teacher can "read" the message, they can't understand it.

There are different ways to **encrypt** (scramble) and **decrypt** (unscramble) information. They generally fall into two buckets:

1. **Symmetric:** encryption and decryption **keys** are the SAME, uses a private key
2. **Asymmetric:** encryption and decryption **keys** are DIFFERENT, uses a private key (metaphorical key) , and a public key (metaphorical lock), every public key only has one private key

   In order to do the actual encryption and decryption, the sender and receiver need keys. **Keys** are essentially sequences of characters that allow the user to randomly scramble a message, a file, a folder, data, a hard drive, etc. any information that you're trying to protect and/or send. A matching key, whether identical (symmetric encryption/decryption) or not (asymmetric encryption/decryption), is needed to decrypt the information.

   The information you are trying to encrypt can be called **plaintext**. Once it has been encrypted with a key and encryption program, it can be called

ciphertext, which is then what is stored and/or transported. It is when the data is **at rest** (stored) or **in transit** that it can be modified or stolen. When the **ciphertext** is then decrypted with a key and decryption program, you hope that it is the same **plaintext** that you sent.

The last concept I will go over is a **cryptographic hash function**. Some may be familiar with the concept of a hash function. We can think of hash functions as any kind of function (think of machine, you put in x, and you get y). Hash functions specifically output a value of a previously specified length. For example, a popular cryptographic hash function is SHA256 — SHA stands for Secure Hashing Algorithm — which given an input, outputs a 256-bit hash value. The difference between a regular hash function and a cryptographic hash function is that that latter attempts to minimize collisions — when two inputs result in the same output. We can see how a collision would be a huge issue if you had encrypted data.

Real-Life Examples You Might Be Not

Even Notice

Now that we've gone over some basic cryptographic concepts, I want to run through a couple examples of encryption/decryption that we encounter every day.

1. **HTTPS (Hyper-Text Transfer Protocol Secure)** is how your local device securely communicates with a web browser or remote site. When you go to an HTTPS-secured server (a secure website that starts with https rather than http), the web browser checks a certificate that lets the browser know that the site is legitimate vs. is a dangerous site that is trying to pose as a different one. HTTPS-secured server prevents information from leaking during the connection. The connection is encrypted using **TLS (Transport Layer Security)**, not to be confused with its outdated predecessor SSL (Secure Sockets Layer).
2. We're pretty used to signing up for things, logging in, and needing to use a **password**. The idea that people are really bad at coming up with and

remembering passwords has already been around for a while. As it turns out, encryption is a big part of what makes password systems work. In general, most systems will encrypt the password that you type in, and just keep the ciphertext. The next time you log in, it will encrypt whatever you've typed in, and then see if that matches with the ciphertext on file. If it doesn't, you can't log in. This is why most accounts and sites can only reset the password, but can't send you your old password.

Further Considerations

- How can you securely store sensitive information such as passwords or cryptographic keys?
- Are you using a secure connection?
- How can you transport the cryptographic keys to parties that need them?
- How do you know that the person that sent the data is who they said they are? How do you verify your own identity? **GPG Key Signing**, **Chains of Signatures**

BONUS: From Caesar Cipher to One-Time Pad

In the process of reading for this blogpost, I stumbled on an interesting series of Khan Academy lessons entitled “Journey to Cryptography,” which can be found here.

This section is for the historically minded. The lessons start with the **Caesar Cipher**, which may be something we read about as little kids without understanding cryptography as a concept, or maybe even came up with ourselves.

Caesar Cipher

This encryption method originated thousands of years ago. You just shift the letters in your message by a certain number of letters, and tell the person that you want to decode the message, the number to shift by. For example, let’s say that the shift is 7, and the message is in English (unclear how this cipher could work for character-based languages).

A -> H

B -> I

C -> J

D -> K

E -> L

.

.

.

V -> C

W -> D

X -> E

Y -> F

Z -> G

So if your message is “Cryptography is cool.” This becomes “JYFWAVNYHWOF PZ JVVS.” The weakness of this cipher is that we can see some letters repeat, and we know that all those letters are encoded to the same original letter. As it turns out, the Caesar Cipher can be broken easily using frequency analysis. For example, in the English language, certain letters appear more frequently like s, a, i, e, etc. Based on the frequently

encrypted letters we can then work backwards to determine what the shift may be.

Polyalphabetic Shift Cipher

An improvement on the Caesar Cipher is to give your friend not 1 shift, but n shifts, which then repeat themselves. For example if you gave your friend the shifts 5, 8, 13, 0, 1, 1, 2, 3 (this is not a random order). Then the first letter is shifted 5 letters, the second letter is shifted 8 letters, the next is shifted 13 letters, the next shifted 0 letters, … the eighth letter is shifted 3 letters, then the sequence repeats. So the ninth letter is shifted 5 letters, and on and on. The longer the sequence of shifts, the harder it is to break the cipher. But still this cipher does not perfect secrecy.

One-Time Pad

The one-time pad, however, provides perfect secrecy, as every letter is assigned a random shift. However, this comes with a cost — a lot of data — you need a shift for every letter in every

message/file/whatever information you're conveying. Generating random shifts was essentially what the Enigma machine that the Germans designed in World War II was doing. However, there was some human error that allowed for patterns to be determined and the Enigma machine to be broken.

References & Resources