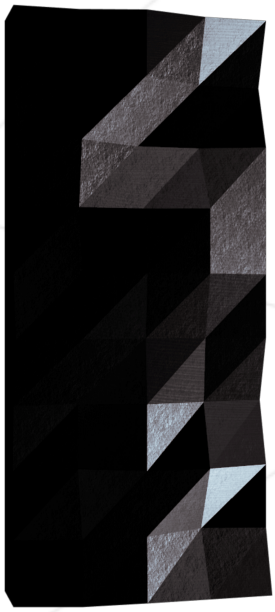

My First Cryptographic Hash

ft_ssl [md5]

42 staff staff@42.fr

Summary: This project is a continuation of the previous encryption project. You will recode part of the OpenSSL program, specifically the MD5 Hashing Algorithm

Contents

Chapter I

Foreword

[Insert something about the Hash Slinging Slasher]

Chapter II

Introduction

This is what Wikipedia has to say on the subject of cryptographic hash functions:

A cryptographic hash function is a special class of hash function that has certain properties which make it suitable for use in cryptography. It is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size. It is designed to also be a `one-way function`, that is, a function which is infeasible to invert. The only way to recreate the input data from an ideal cryptographic hash function's output is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Bruce Schneier has called one-way hash functions "the workhorses of modern cryptography". The input data is often called the message, and the output (the hash value or hash) is often called the message digest or simply the digest.

The ideal cryptographic hash function has five main properties:

- it is deterministic, so the same message always results in the same hash
- it is quick to compute the hash value for any given message
- it is infeasible to generate a message from its digest except by trying all possible messages
- a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value
- it is infeasible to find two different messages with the same hash value

Cryptographic hash functions have many information-security applications, notably in `digital signatures`, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as `checksums` to detect accidental data corruption. Indeed, in information-security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums, or just hash values, even though all these terms stand for more general functions with rather different properties and purposes.

Chapter III

Objectives

This is the second `ft_ssl` project on the path of `Encryption and Security`. You will recode some security technologies you may have already been using from scratch.

This project will focus specifically on cryptographic hashing algorithms.

If you have ever downloaded a file of significance from the internet (an operating system installation image, for example), you have probably noticed a section with either SHA-1 or MD5 followed by gibberish. If you were smart, you have looked up what the string meant and verified your download with the given algorithm to verify the hash. If you haven't, I advise you do some research to learn more about digital fingerprinting so you can better protect yourself from malicious downloads.

You will expand your `ft_ssl` executable from the previous project to include the `MD5 Hashing Algorithm` to solidify your understanding of the bitwise operations, integer overflow, and one-way functions.

Chapter IV

General Instructions

- This project will only be corrected by other human beings. You are therefore free to organize and name your files as you wish, although you need to respect some requirements below.
- The executable file must be named `ft_ssl`.
- You must submit a Makefile. The Makefile must contain the usual rules and compile the project as necessary.
- Your project must be written in accordance with the Norm.
- You have to handle errors carefully. In no way can your program quit unexpectedly (Segfault, bus error, double free, etc). If you are unsure, handle the errors like `OpenSSL`.
- You'll have to submit an author file at the root of your repository. You know the drill.
- You are allowed the following functions:
  - `open`
  - `close`
  - `read`
  - `write`
  - `malloc`
  - `free`
  - any functions required to complete previous exercises
- You are allowed to use other functions as long as their use is justified. (Although they should not be necessary, if you find you need `strerror` or `exit`, that is okay, though `printf` because you are lazy is not)
- You can ask your questions on slack in the channel `#ft_ssl`

Chapter V

Mandatory Part

You must build upon the program `ft_ssl` that you created in the previous exercise.

```
 > ft_ssl
usage: ft_ssl command [command opts] [command args]
```

For this project, you will implement the options of the `md5` hashing function.

`man openssl`

V.0.1 Mangled Data 5

`MD5` is a cryptographic hash function. It is short for "Message Digest 5". It has found to contain several vulnerabilities, but can still be safely used as a checksum to verify data integrity against unintentional corruption.

Create a command for your `ft_ssl` program that we can use to create a one-way hash using `md5`:

```
 > echo "pickle rick" | openssl md5
c5e433c1dbd7ba01e3763a9483e74b04
 >
 > echo "pickle rick" | md5
c5e433c1dbd7ba01e3763a9483e74b04
 >
 > echo "pickle rick" | ft_ssl md5
c5e433c1dbd7ba01e3763a9483e74b04
 >
 > diff <(echo "pickle rick" | md5) <(echo "pickle rick" | ft_ssl md5)
 >
```

- You must format your output the same as either the `md5` executable or the `openssl md5` command, but be consistent with your decision across the whole program.
- You must implement the following flags from the `md5` unix executable: `-q` and `-s`.

`man md5`

Chapter VI

Bonus part

- You may add the following flags from the `md5` executable for bonus: `-p` and `-r`.
- You may also add to your `ft_ssl` program additional one-way hash functions.

`SHA-256, SHA-512, and Whirlpool are all good options.`

`A hash function that is weaker than MD5 will not grant a bonus.`

As you should expect by now, the bonus will not be considered unless the mandatory part is complete and perfect.

Chapter VII

Turn-in and peer-evaluation

Submit your code to your `Git` repository as usual. Only the work in the repository will be considered for the evaluation. Any extraneous files will count against you unless justified.