SHA-1 Hash Attack Analysis

Reviewed by Josh Fenwick

Introduction

In the world of computer science, hashes play an important role in many areas. They allow, among other things, a quick comparison of equality and a way to quickly verify if a message has been unaltered. The way that they do so is by taking an input of varying length and producing a value of fixed length that is practically unique to that input. As a simple example, there could be a hash that maps the string `Hello world!` to the value `112233` and the string `Goodbye world!` to the value `665544`.

There are a couple important properties that a hash function must have in order to be considered secure. This analysis focuses on the requirement that a hash must generate a *practically unique output* for every input. This means that it should be computationally infeasible to find another input that produces the same output as a given input.

Due to the fixed length of the output, there are a limited number of outputs that can be produced. For example, if a hash produces an output that is 8 bits in length, there are only 2ˆ8, or 256, different possible outputs. This means that, while the theoretical goal is to provide a unique output for every input, the practical goal is just to provide an output for every input such that it would take an impossibly long amount of time to find another input that produces the same output.

When two inputs produce the same output, this is known as a *collision*. If someone is able to derive a method to produce a collision that is computationally feasible, it renders the hash insecure. There are two methods (known as *attacks*) that this analysis will be exploring: the *classical collision attack* and the *pre-image attack*. The objective of this analysis is to test if the attack designed for this analysis takes the expected amount of time.

The objective of the *classical collision attack*, which we will refer to as just the *collision attack*, is to find two values that produce the same hash. This attack is expected to take `2^(n/2)` units of time, where `n` is the number of bits in the hash.

The *pre-image attack* takes a hash and finds some value that produces the exact same hash. This attack is expected to take `2^n` units of time, where `n` is the number of bits in the hash.

Methodology

This analysis has a few different elements from which it is built: the hashing function, the hash attacks, and the benchmark code.

Hashing

The hashing function is a wrapper for the SHA-1 hashing function. It takes an input string and a bit size `n` and returns the first `n` bits of the SHA-1 hash. This allows the benchmark to test against hash lengths of varying length.

Hash Attacks

The collision attack is implemented as follows. It takes a starting value as input. Beginning with that value, it converts the number to a string and hashes that string. Then, it stores the number and its hash in a hash map. Next, it increments the number and hashes the new number. It checks if the hash map already contains the hash. If it doesn't, the function adds it and continues. If it does, the function returns.

The reason that the function takes a starting number as input is so that the attack can be tested in different situations. One of the properties of a hash function is that it returns the same hash for the same input. Therefore, if the attack started from 0 each time, it would produce the exact same hashes and end up finding the exact same collision each time, so the results would be the same. In order to avoid this, the attack is passed a random number generated by a cryptographically secure random number generator.

The pre-image attack takes as input the string for which it is looking for a matching hash. (Alternatively, the attack could take the string's hash as input, this analysis takes the string itself as input for convenience). Then, starting from `0`, it converts the number to a string and hashes it. If the hash doesn't match the hash of the original string, the function increments the value and continues. If the hash matches, the function returns.

In this case, the randomness required to test the attack in different situations is generated by the input string, which is a string of 16 characters randomly generated by a cryptographically secure random string generator.

Benchmark

In order to test the attack functions, the benchmarks ran the attacks over a series of randomly generated numbers or strings, as mentioned previously. It also ran both attacks over varying bit sizes. The chosen bit sizes are 8, 11, 16, 19, and 24. Bit sizes 8, 16, and 24 are chosen because they are multiples of 8, while 11 and 19 were chosen because they were prime numbers that were relatively in the middle of the multiples of 8.

In addition, the benchmark runs the collision attack 1000 times and the pre-image attack 100 times. The reason for the difference in magnitude is that the collision attack runs much faster than the pre-image attack. 1000 rounds of the collision attack with a 24-bit-size hash takes about 5 seconds, while 100 rounds of the pre-image attack with a 24-bit-size hash takes about 15 minutes to run.

Thus, in order to complete the analysis in a reasonable amount of time, the pre-image attack was limited to 100 rounds.

Analysis

The benchmark was run, and the results are shown in the graphs below. Figure 1 represents the data collected in the collision attack benchmarks. Figure 2 represents the data collected in the pre-image attack benchmarks. Each of the graphs contain the data points for each bit size graphed as a vertical box plot in order to display the distribution of data points. As a note, the scale for the number of hashes is logarithmic in order to account for the fact that the time required by the attacks grows exponentially.

There are also two lines displayed on the graph. The blue dotted line represents the expected average amount of time the attack should take, while the red solid line represents the actual average amount of time the attack took. In these graphs, the average is represented by the median data point rather than the mean. This is because, due to the large dataset, there were some outliers that would have skewed the mean.

Collision Attack

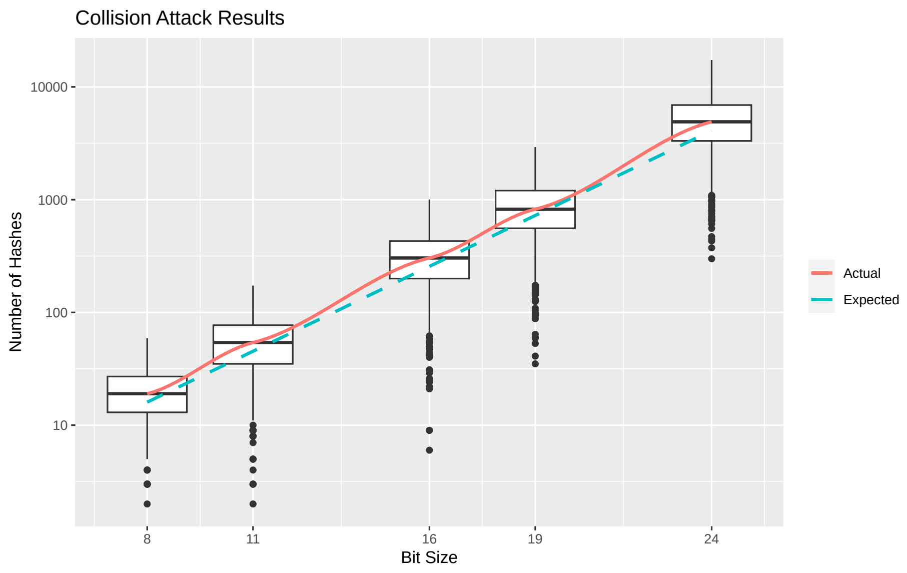

Figure 1: Data collected in the collision attack benchmarks

In the collision attack, it can be seen that the actual average followed the expected average pretty consistently. Although the average itself was just above the expected average, the expected average remained within the interquartile range of the data.

Pre-Image Attack

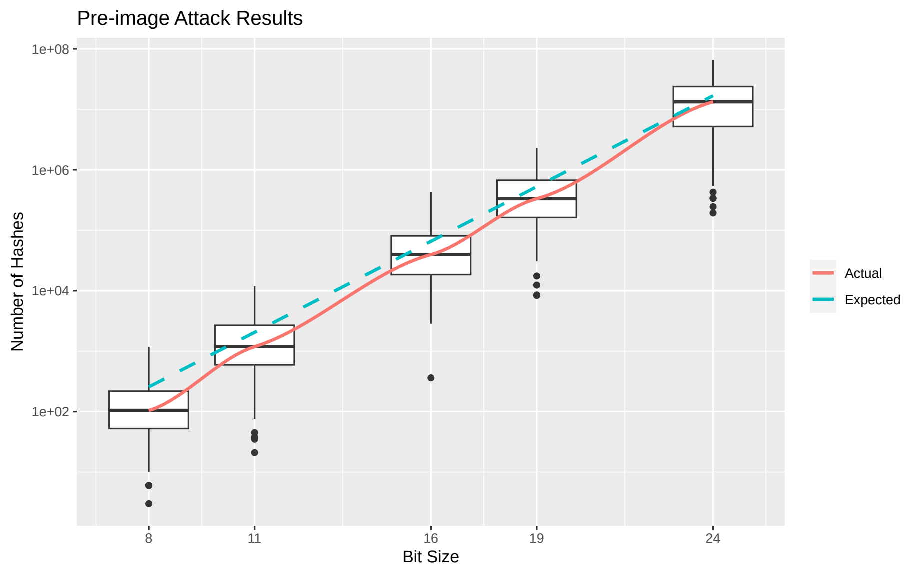

Figure 2: Data collected in the pre-image attack benchmarks

In the pre-image attack, the story is mostly the same as the collision attack. For the most part the actual average followed the expected average. With a size of 8 bits, the difference between actual and expected seems a lot greater than the difference with a size of 24 bits. However, this is accounted for by the logarithmic scale. In addition, due to the lower number of trials run, the data isn't going to match the expected average as tightly. If more trials were run, the average would likely be closer.

Conclusion

Understanding the practical limits of a hash is important to understanding its security. In this analysis, we verified the claims made about attacks on a hash. For the classical collision attack, we compared the actual data with the expected time of `2^(n/2)` and found that it matched pretty well. For the pre-image attack, we compared the actual data with the expected time of `2^n`. This didn't follow as closely as the classical collision attack, but it was still close. This was likely due to the fact that we were unable to perform as many trials due to the fact that the pre-image attack takes a longer amount of time. Thus, the expected values are an accurate model for the attack.