Collision and Pre-Image Attacks on Hashing Algorithms

Introduction

Hashing is the process of transforming a string into a different string to conceal its information and/or ensure its information hasn't been tampered with. Hashing is performed by applying a "hashing function" to the message using a random key. Said function is easy to perform, but the message is hard to decrypt without the given key. In addition to that, the original string can have any size, but the result of the hash function will have a predefined size regardless of the size of the original string. Thus, if the hash string has *n* bits, then there are a total of *2^n* possible hash strings to be returned from the hashing function. Since an unlimited number of strings map to a limited number of hashes, this means different strings will map to the same hash value. Attacks can be conducted on hash values to exploit this particular property of hash functions. In this document, collision attacks and pre-image attacks will be analyzed and compared in theory and in practice.

Implementation

The author implemented collision and pre-image attacks using different hash bitsizes, recording all values to compare with the theoretical performance. To increase confidence, the author performed 50 attacks at each bitsize selected for that attack, then average them to compare the result to its theoretical counterpart. All attacks were performed using python3.

Collision Attacks

Collision attacks try to find two input strings that map to the same hash value. In theory, this attack should be successful after *2^(n/2)* attempts, where *n* is the number of bits in the hash value. The author implemented the attack by generating random messages of 20 bytes, then storing its hash value in a dictionary (although it could be done in any number of data structures). The process continued in a loop until one of the random messages hashed to an already stored hash. The number of attempts was then recorded for each attack and averaged for comparison, which can be seen in figure 1. Additionally, figure 1 shows that the results of the author's implementation is extremely close to what is expected in theory. A larger difference between practice and theory can be observed at 29 bits. This difference, however, isn't abnormal enough to conclude a wrong implementation, as an average of 50 attacks can still produce significant variance.

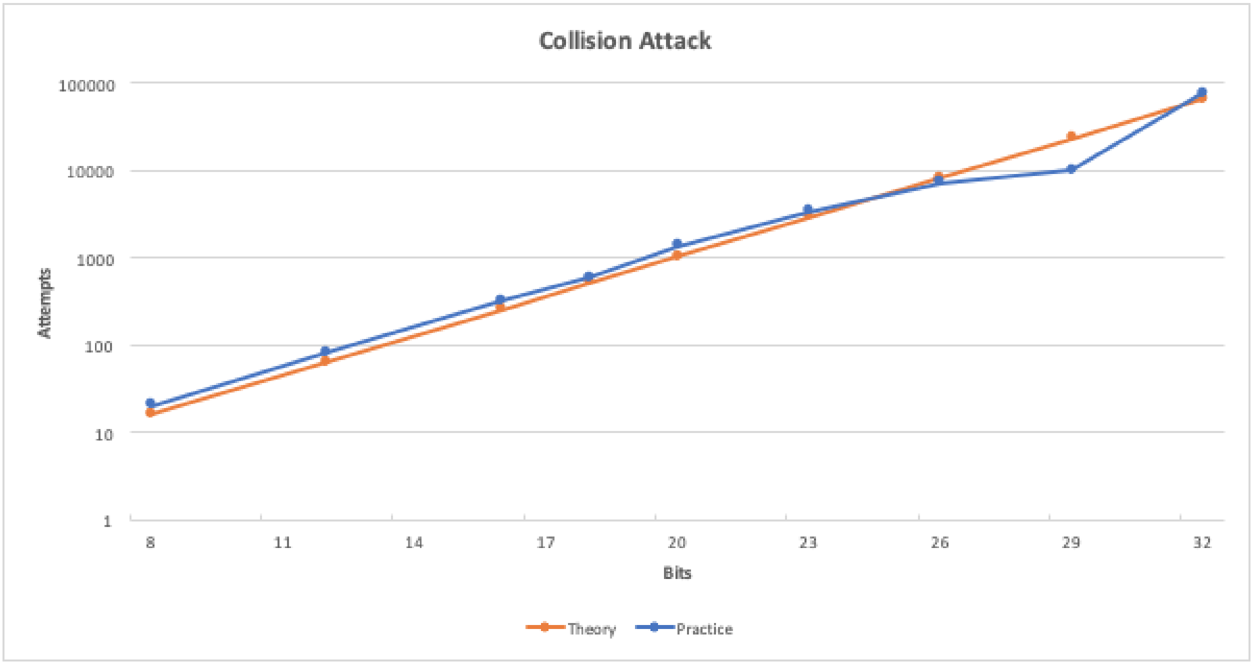

Figure 1

Pre-Image Attacks

Given an output hash value, pre-image attacks try to find a message that hashes to that value. In theory, this attack should be successful after *2^n* attempts, where *n* is the number of bits in the hash value. The author implemented the attack by generating a random hash at each bitsize, then generating new random messages in a loop and hashing them until one of the new hashes matched the original hash. Similarly to the collision attack, the result of the author's implementation is extremely close to what would be theoretically expected, as seen in figure 2. Notice that the pre-image attack implementation has less data points than the collision attack. This decrease in data points was a conscious decision. As the algorithm has a larger exponent, recording 50 attacks on a hash of over 20 bits would take a substantially larger amount of time to compute.

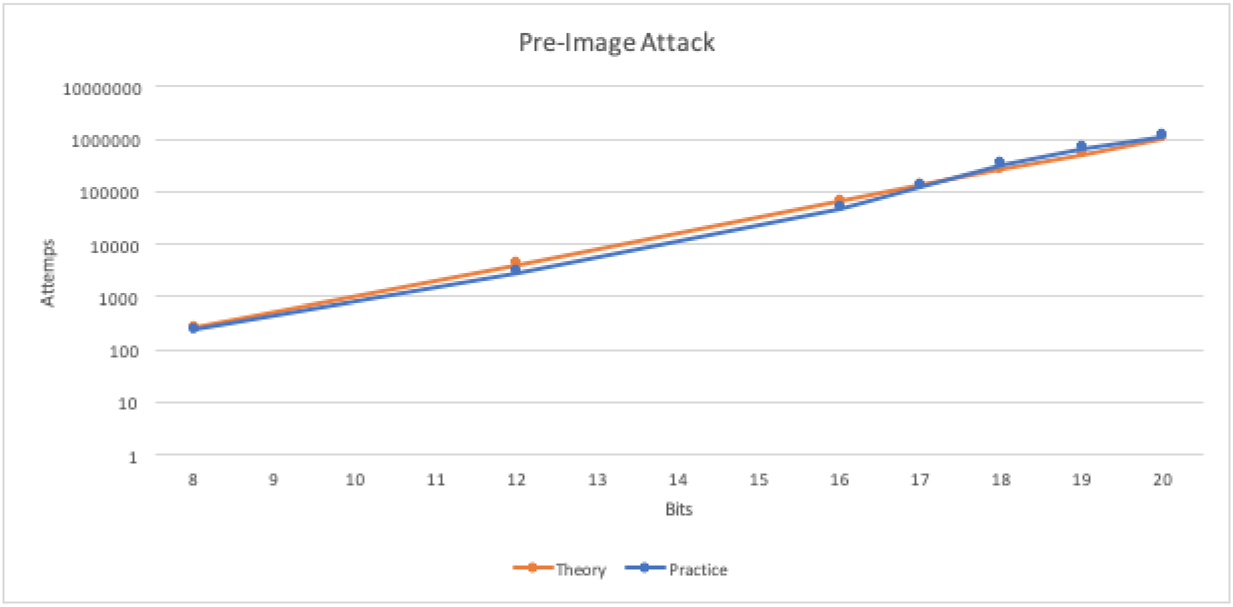

Figure 2