Hash Attack

Conner Brinkley

09.22.2020

Applied Cryptography
The University of Tennessee

Reviewed by: Austin Knight

INTRODUCTION

The purpose here is to investigate two different types of hash attacks, understand how they work, and then see if the theoretical time for each attack holds true in the real world. The two types of attacks being executed are pre-image and collision. Both attacks were performed on the SHA-1 algorithm, which was intended to produce a new hash value for each unique input. SHA-1 spits out a 160 bit hash every time it is called, which would take the 2015 MacBook Pro being used an unreasonable amount of time to find a match. Because of this, a series of smaller experiments were conducted in the range of 8-32 bits instead of 160 to see if the theoretical attack times scale comparatively.

PROCESS (& THEORY)

Pre-image attacks work by beginning with a random given hash and then trying to find a message that maps to that same starting hash, which theoretically should roughly scale around 2^n time, where n is the number of bits in the hash digest. Collision attacks are when any two different messages are found to hash to the same value, and in theory, they should scale at a slower rate than pre-image attacks. The time for a collision attack is around $2^{n/2}$ where n is also the number of bits in the digest.

To test this theory, a C++ program was written that generates random strings to give the SHA-1 algorithm. The program takes two command line arguments: the first being which attack to attempt and the second being the bit size to compare. The bit sizes being tested in this experiment are 8, 16, 24, and 32.

```
./Hash-Attack [Pre-Image|Collision] [Bit-Length]
```

For the collision attack, the program kept generating digests with random inputs, which were stored in a hash table until it generated one that was already in the table. Once it found a match, it stopped and saved how many iterations (or attempts) it took for that sample to break SHA-1 at that given bit size to a CSV file. Each of the four experiments generated a total of 50 samples, which were then averaged to find a more accurate number of attempts at each bit size.

For the pre-image attack, the program generates a random hash to start with, and then keeps generating them until it finds a match that was not generated with the same input. This is where problems started arising because of how much the number of attempts scaled up. Figure 1 below shows the output of the first attempt at running the 32-bit experiment.

```
Broke SHA-1 with a pre-image attack at iteration 4923105 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 20905826 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 1239749 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 6765911 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 37256142 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 35598331 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 4142361 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 19924884 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 76754835 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 18923703 with 24 bits.
Broke SHA-1 with a pre-image attack at iteration 87412466 with 32 bits.
Broke SHA-1 with a pre-image attack at iteration 1073741822 with 32 bits.
Broke SHA-1 with a pre-image attack at iteration 1073741822 with 32 bits.
Broke SHA-1 with a pre-image attack at iteration 1073741822 with 32 bits.
```

Figure 1. First sign of trouble with pre-image attacks at 32 bits.

After seeing that, it did not seem like a coincidence that the attack would succeed at 1,073,741,822 several times in a row. The data for the 32-bit experiment was then scrapped, and the program was modified to print out what random strings were being given to SHA-1 when it generated the target and found a match.

```
YplqXbInsjseMuwKNDig |  e3 fa 48 d3 (TARGET)
BPSkFjXrkCpubMfRUZyV |  e3 fa 48 d3 (MATCH!)
AiDpChdpNfpLnGZYzsBx |  93 37 d9 ea (TARGET)
HWQdbcPNXFnFgxzbkTrn |  93 37 d9 ea (MATCH!)
ZSjpgySyTJKRMxWccCXd |  20 ea ee 21 (TARGET)
ZSjpgySyTJKRMxWccCXd |  20 ea ee 21 (MATCH!)
pJmQHdICSYOnARPlQfaS |  86 c2 f4 bb (TARGET)
pJmQHdICSYOnARPlQfaS |  86 c2 f4 bb (MATCH!)
dZOomkiOhtyMcQXqCYnv |  d5 a0 81 ca (TARGET)
dZOomkiOhtyMcQXqCYnv |  d5 a0 81 ca (MATCH!)
uLdfgardewEhrYVrxpOB |  c2 a1 c0 84 (TARGET)
uLdfgardewEhrYVrxpOB |  c2 a1 c0 84 (MATCH!)
gBYcyuDySpCwaQWKvFwt |  43 54 18 56 (TARGET)
```

Figure 2. Comparing the inputs and outputs of the SHA-1 algorithm.

The output from the program in Figure 2 interestingly showed that for several of the matches, the program is generating the exact same “random” string at the same iteration (1,073,741,822) for multiple samples. After deep diving into why this was happening, it was discovered that all pseudo random number generators have a period before they start repeating. This issue occurred because the rand() function in C has a relatively short period, which happened to be around exactly 1,073,741,822 calls. This took some time to figure out, but once the bug was found, rand() was swapped out with the C++11 64-bit random number generator mt19937_64(). This simple fix made it possible to generate several billion attempts for each sample. Because of how long it took to debug, it was unreasonable to generate 50 pre-image samples at 32 bits, so only 9 were generated.

RESULTS

Despite the issues with the pseudo random number generator, the average results still stayed consistent with the theory. Figure 3 below plots the results against the theoretical number of attempts at each bit size for a collision attack.

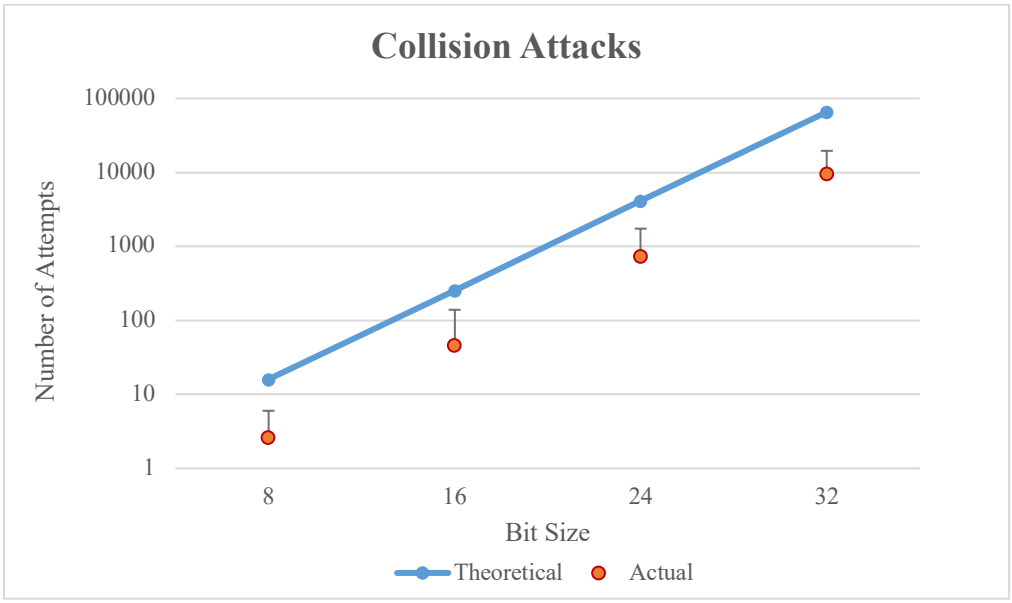

Figure 3. Number of actual attempts vs. theoretical to perform a collision attack on SHA-1 at each bit size.

Although it looks like the values were a bit lower than expected for collision attacks, they still grew at the same $2^{n/2}$ rate. Both Figure 3 and Figure 4 use a logarithmic scale to make the data easier to compare. Figure 4 below shows the same graph for pre-image attacks.

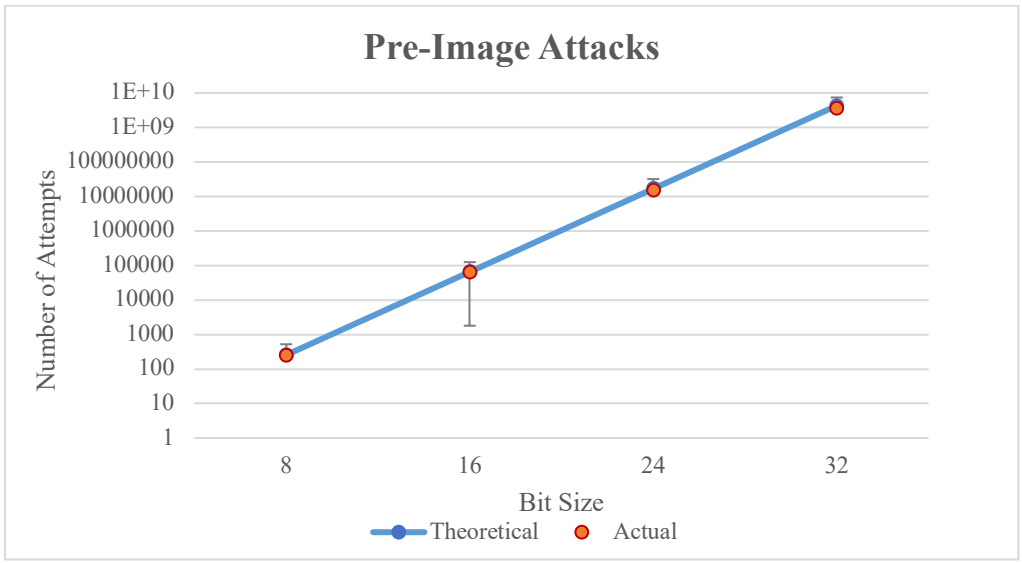

Figure 4. Number of actual attempts vs. theoretical to perform a pre-image attack on SHA-1 at each bit size.

Unlike the collision attacks, the number of actual pre-image attempts were extremely close to the theoretical values. Although only 9 samples were generated for the 32-bit experiment, they still seem to be consistent with the rest of the data.

CONCLUSION

This project had a major setback with generating the data, but once it was overcome, it was generally found that the theoretical time for both collision and pre-image hash attacks held true in practice. Looking back, a better approach would have been to use Python as the programming language, but at the time it seemed like C++ would be easier because it is what was most familiar. In the future for projects with a lot of data, Python will most likely be used.