Luke Albin

Project #2

Hash Attack Experiment

Reviewed by: Abel Nelson

Purpose:

For Project #2, we structured and executed code to simulate the functionality involved in a "hash attack" where a value is found that has the same hash as another input. In practice, these techniques can be used to intercept messages and insert malicious information without the message being flagged as altered.

Structure:

For my experiment, the two methods of "attack" were a pre-image attack and a collision attack. The pre-image attack takes an original input and its hash and then generates new, random inputs until a matching hash is found. Once a matching hash is found, we will have two differing inputs that contain an identical hash.

The collision attack operates similarly but rather than trying to match a single hash, random inputs and their hashes are stored and compared until any two hashes are identical. In this method, collisions are found much faster, but the hashes found are random and less precise.

The programming language I used for the implementation of this experiment is Java. This is because it has decent speed over a language like python, garbage collection, and is a language I am very comfortable with.

The hashing algorithm being used is SHA-1 as it has strong security, but collisions and vulnerabilities exist which can be exploited for the purpose of this experiment.

Hashes will be truncated to a set number of bits in order to be completed in a reasonable manner for this experiment. This means we will be able to encounter a collision much more frequently than we normally would.

To see trends and compare our data with the expected data, we will be keeping track of the number of hashes that occur before we discover a matching hash. For the pre-image attack, we expect to find a collision in 2^n hashes where n is the number of bits in the hash. For the collision attack, we expect to fin a collision in 2^(n/2) hashes where n is again the number of bits in the hash.

This experiment will work by taking a certain bit size of hash, and then running a certain number of trial on each attack and comparing the theoretical number of hashes with the actual.

Results:

Pre-Image Test

Type	Bit Size	Num Hashes
PreImage	6	61.51
PreImage	8	259.96
PreImage	10	1057.39
PreImage	12	3717.71
PreImage	14	18780
PreImage	16	66912
PreImage	18	250027.2
PreImage	20	291187.6
PreImage	24	9844135
Expected	6	64
Expected	8	256
Expected	10	1024
Expected	12	4096
Expected	14	16384
Expected	16	65536
Expected	18	262144
Expected	20	1048576
Expected	24	16777216

Bit Sizes 6-16 are the calculated mean of 75 trials
Bit Sizes 18-24 are the calculated mean of 5 trials

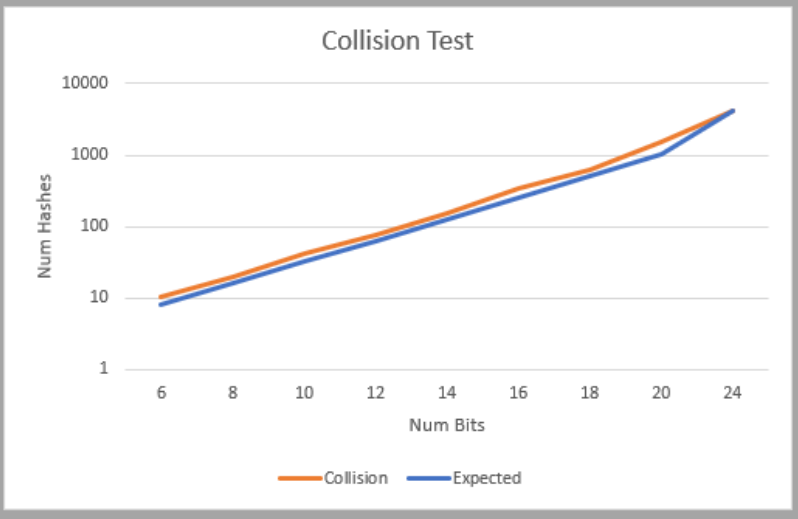
In the pre-image test, I ran tests on hash sizes ranging from 6 bits up to 24. Tests on sizes 6 through 16 were run on 75 trials in order to get a good average solution while tests on sizes 18 and above were run on 5 trials each. This decision was purely for the sake of time as running the tests already took over 2 hours.

Note that this graph is in a logarithmic scale. As we look at the data, we can see that our actual number of hashes is always less than our expected number of hashes. We can also tell that we have very little variance from this trend until we reach large bit sizes. This is almost certainly because less trials were run on these large bit hashes.

We can also notice how our hashes required to find a solution increases exponentially as we increase the number of bits being operated on. This is also a proof of concept on how difficult and expensive a pre-image attack would be on full-sized hashes which often reach bit sizes of 512.

Collision Test

Type	Bit Size	Num Hashes
Collision	6	10.45
Collision	8	20.05
Collision	10	42.32
Collision	12	75.12
Collision	14	153.96
Collision	16	346.09
Collision	18	635.2
Collision	20	1537
Collision	24	4197.4
Expected	6	8
Expected	8	16
Expected	10	32
Expected	12	64
Expected	14	128
Expected	16	256
Expected	18	512
Expected	20	1024
Expected	24	4096



Bit Sizes 6-16 are the calculated mean of 75 trials
Bit Sizes 18-24 are the calculated mean of 5 trials

In the collision test, similarly to the pre-image test, I ran tests on hash sizes ranging from 6 bits up to 24. Tests on sizes 6 through 16 were run on 75 trials in order to get a good average solution while tests on sizes 18 and above were run on 5 trials each.

Note that this graph is in a logarithmic scale. Looking at the data, our actual number of hashes actually exceeds our expected number of hashes quite consistently. This is most likely due to my implementation either comparing inefficiently, or the method I used to get random inputs. However, the trend varies very little so we can still observe the trends shown.

Notice how our hashes required to find a solution increases at a much slower rate than our pre-image test. This occurs because we are able to find a collision between any two inputs, not just a collision with a specific hash.