Hash Report

By Grant Alderson

My implementation of collision and preimage attacks correctly represents the expected outcome and difficulty of implementing such an attack. The bit sizes used are [8,10,12,14,16,18,20,22] with the number of iterations for a successful attack increasing with bit size. Each attack was conducted with 50 samples for each bit size.

Preimage

The preimage attack Is conducted by hashing a random string, using Hashlib and Random ,and truncating it to a specific bit size. A random binary number of the same size is then generated and checked against that hashed number. This process repeats many times until one of the pairs matches.

Collision

The collision attack Is conducted by hashing a random string, using Hashlib and Random ,and truncating it to a specific bit size. It then checks if it has seen that hash before and if so, it breaks out of the loop. If it has not seen the hash, it is added to an array of previously seen hashes that future hashes will be checked against.

Average Preimage iterations

[169.62, 996.38, 4590.72, 16537.94, 74980.4, 293069.22, 976498.64, 3804517.56]

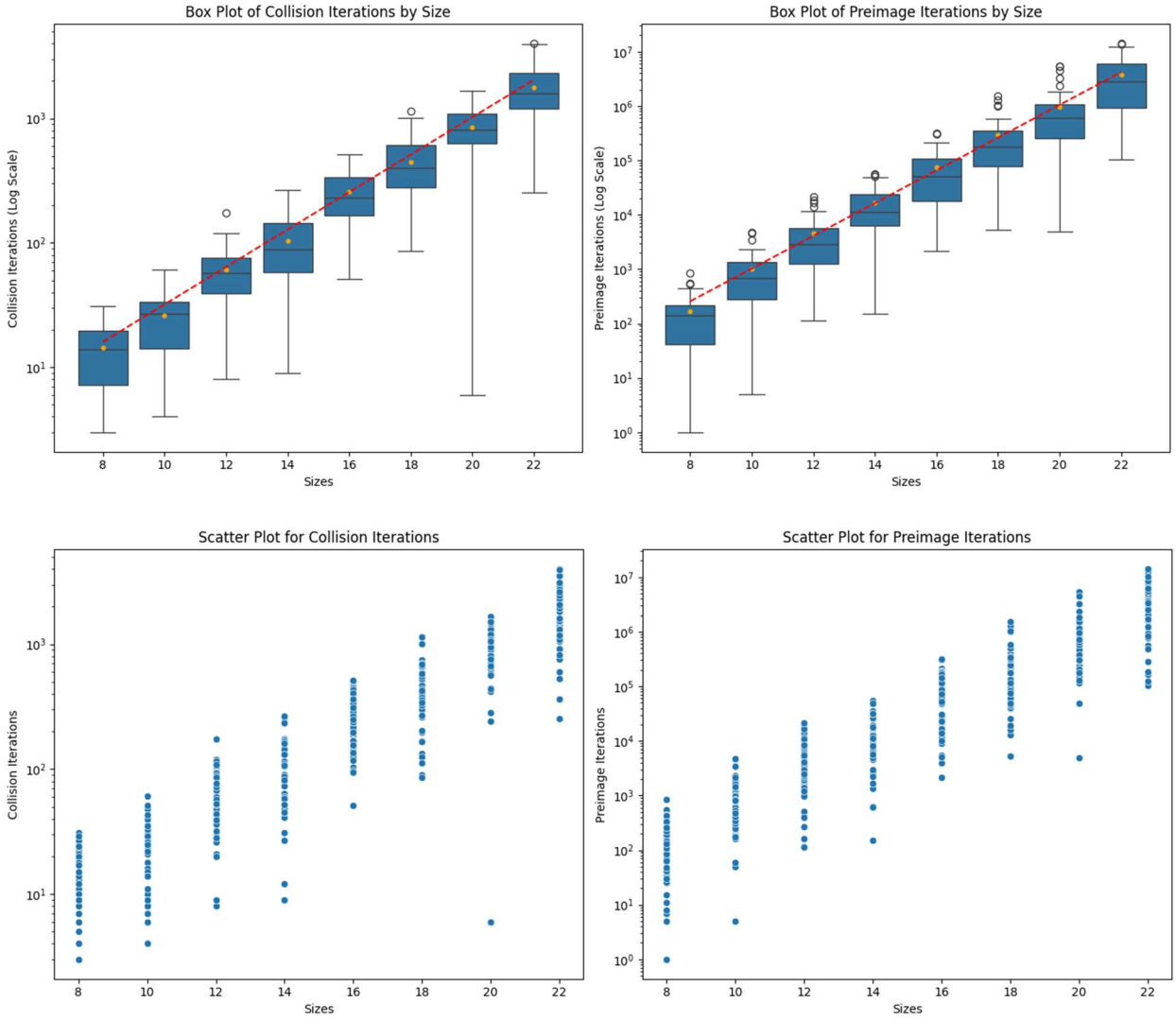
Average Collision iterations

[14.46, 26.0, 61.3, 103.58, 256.4, 447.46, 854.04, 1779.3]

As you can see from above, collision is greatly superior in number of iterations required for a successful attack. The average number of iterations for each bit size is shown from the orange dot in each bar.

The number of expected iterations is shown by the red line on the graphs. These lines were drawn based on the 2^n equation for preimage and 1.1774*(2^(n/2)) for collision which is their theoretical complexity.

The variance between my actual results and the theoretical results is rather minimal as you can see from the average compared to the expected in the graphs. Any further difference is most likely attributed to the sample size of 50 per bit size. With a greater number of samples, it would most likely be even closer to the expected value. It could also be a product of my implementation.



The code to create these graphs was made in Python using Matplotlib, pandas, and seaborn.

This Paper was reviewed by: Makenzie Johnson