Authentication and Hash

Kyle Jennings, Zarek Lazowski, Evan Morris

# Questions

1. What do you observed based on Task 1b? How many bytes are different between the two digests?
   1. The two digests are pretty much completely different. As far as I can tell most if not all the bytes are completely different.
2. What is the maximum number of files you would ever need to hash to find a collision on an n-bit digest? Given the birthday bound, what is the expected number of hashes before a collision on a n-bit digest? Is this what you observed? Based on the data you have collected, speculate on how long it might take to find a collision on the full 256-bit digest.
   1. The maximum number for an n-bit digest is 2^n. So, a 10-bit digest would have 2^10 or 1024 unique values. To guarantee a collision you would do the unique values + 1 (1024 + 1).
   2. There is an equation to figure out the number of items to check to get a 50% chance of finding a collision.  
      So for a 10-bit digest we would plug n = 10 into the above equation and get 38.1 as our output. For our data we got 23, which is kind of close.  
        
      For a 30-bit digest we would plug n = 30 into the above equation and get 38582 as our output. For our data we got 65856. Again, not super close but this is because we are calculating for 50% and we may or may not get lucky in the actual results.
   3. For a 256-bit hash we could plug that into the equation, and we would get 4\*10^38. Very large.
3. Given an 8-bit digest, would you be able to break the one-way property (i.e. can you find any pre-image)? Do you think this would be easier or harder than finding a collision? Why or why not?
   1. Yes, there are only 2^8 (256) possible digests, so any operation would be trivial.
   2. This is more difficult than finding a collision because you have to check one value against all the other values instead of all the values against each other.
4. For Task 2, given your results, how long would it take to brute force a password that uses the format word1:word2 where both words are between 6 and 10 characters? What about word1:word2:word3? What about word1:word2:number where number is between 1 and 5 digits? Make sure to sufficiently justify your answers.
   1. The number of words in the corpus is around 135,000 words. With only word1 we only must check the 135,000 values. Every time we add another word we have to increase the exponent that value is raised to (135,000^1..2..3..etc.). This would effectively square the original time with one extra word and cube it with two.
   2. word1:word2 -> (135,000^2) values to search through
   3. word1:word2:word3 -> (135,000^3) values to search through
   4. As far as adding a number at the end goes, we only have to multiply the previous value by the number of possible numbers, in this case, with 5 digits, we would have to multiply it by 10^5 (100000).
   5. word1:word2:number -> (135,000^2 \* 10^5)