Ng Jia Sing 1006648

Part 1: Hashing using MD5.

In a python script, we created a list of plaintext and encoded it with the *hashlib* library.

A screenshot of a computer program

Description automatically generated

* **How does the length of the hash correspond to the input string?**

As we can see, the length of the hash will always be 32 regardless of the input string.

* **Are there any visible correlations between the hash and the input string?**

Comparing the difference in hashes of ‘a’, ‘aa’, and ‘aaa’, we can see no visible correlations between the hash and the input string.

* **What are the issues related to the cryptographic weakness of MD5?**

As the hashes are relatively short in length, hence lowering the resistance against brute force attack and collision attacks.

Part II: Break Hashes with Brute Force

A screenshot of a computer program

Description automatically generated 

* **How much time did you take in total?**

It took 70.9867 seconds in total

* **How much time does it take to crack each string, on average?**

4.73244 second

* **Is it possible to amortize (gradually write off the initial cost of) the brute forcing attempts?**

Yes, for each of the generated hashes, we compare to the entire list of valid hashes. This way, a missed hash reduce the total number of tries before reaching a successful break of the hash for all 15 hashes.

Part III: Salt

**• What are the observed differences between your ease of cracking the salted vs the unsalted plaintexts?**

With the salted plaintexts, the time taken to break the hash using brute force method was significantly longer than the time taken to break the hashes for the unsalted plaintext (70.9867s vs 1103.9885 second).

**• Report the difference in time taken to crack the salted and the unsalted hash values.**

The large difference in time difference is likely due to the larger number of possible hash values when salting is added. With the unsalted hashes, there exists only 36^5 = **60,466,176** combinations but there are 36^6 = **2,176,782,336** combinations for the salted hashes.

**• Explain any differences between salted and un-salted crack strategies.**

After breaking the hashes with the salted messages, more effort has to be put into deciphering the original plaintext as we do not know which characters were used for salting.

Part IV: Hash Breaking Competition

* **What is the approach you used to crack the hashes**

Utilising online databases namely [hash.com](https://hashes.com/en/decrypt/hash).

* **How you decided or designed your approach**

As I have already created the code to break the hashes through brute force, I ran the code from ex3.py in hopes of breaking a few hashes. Afterwards, the remaining hashes that was not broken I ran it using a third party software called hash.com.

* **Main challenges and limitations of your approach**

Brute force method would give insignificant results due to significantly larger set of possible plaintexts. Furthermore, we’ll have to trust the plaintext given by the hash.com regarding to the rest of the hashes.

* **How many hashes out of the total did you manage to crack** (there is no prize)

100 hashes.