


How Hashing Algorithms Affect Password Security

Final Presentation

Ashley Jacobs, Alex Salmeron, Nathaniel (Nate) Kieffer and Sammuel Licea



● The Problem

- 
- Sensitive data is not always stored securely.
 - One of the basic security techniques used to store sensitive data is hashing.
 - However, just hashing this data isn't enough. We must choose a secure algorithm.
 - We seek to determine which hashing algorithms are secure enough to store sensitive data like passwords.
 - We focus on the MD5, SHA-1, and SHA-256 hashing algorithms throughout this project.

Related Work

Researching the Relationship between Time and Memory

Last Time we began to discuss why super fast and efficient hash functions are not secure.

- Hashing algorithms that are fast and require little memory are easily brute forced.

What are factors that make up a good Hashing algorithm

- No Collisions
 - When you can achieve collisions is when an algorithm is abandoned.
- Time Memory Trade off Defence (TMTO)
 - the concept that the speed of a hashing algorithm should be fixed if it meets the required memory to perform its hash but take more (exponentially preferably) if less than the fixed amount of memory is offered.
 - This makes performing dictionary attacks harder unless you have specialized expensive hardware (ASIC processor which is used for bitcoin mining)

Related Works

MD5, SHA-1, SHA-256 and SHA-512 speed performance - Lyudmil Latinov

- This was a paper that compared the speeds of the 3 standard hashing functions with different string lengths and we used as a reference for our work

Cryptanalytic time-memory trade-off for password hashing schemes - Donghoon Chang

- This paper talked about TMOC and memory hardness and gave us the background of the whys and hows slow hashing is more secure.

Adding Salt to Hashing: A Better Way to Store Passwords - Dan Arias

- This article took our understanding past what the interworking of hashing is and explored ways to protect your hashes even if users have weak passwords.

Ex. Basically consider you had the password: hotdog

- easily cracked with dictionary attack

With the salted value “Salt3d!” this password becomes: Salt3d!hotdog

- Much more secure
- More flavor

Our Solution

Our project centers on two focus areas: research about the security of hashing algorithms, and the implementation of a password cracker tool.

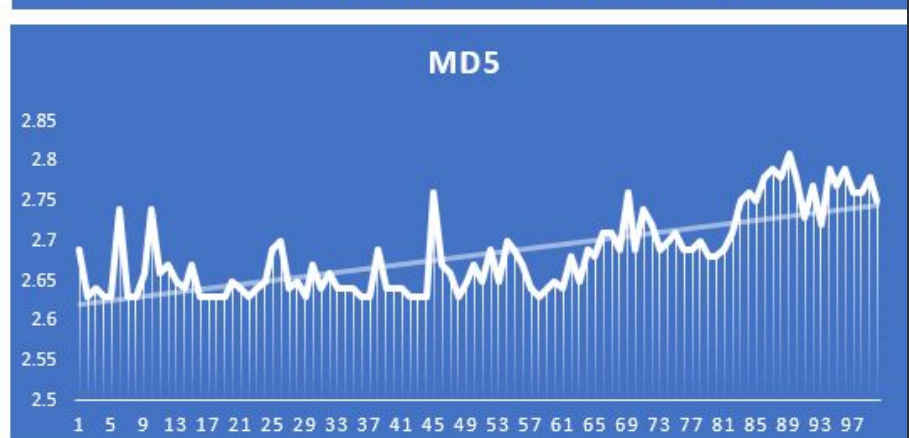
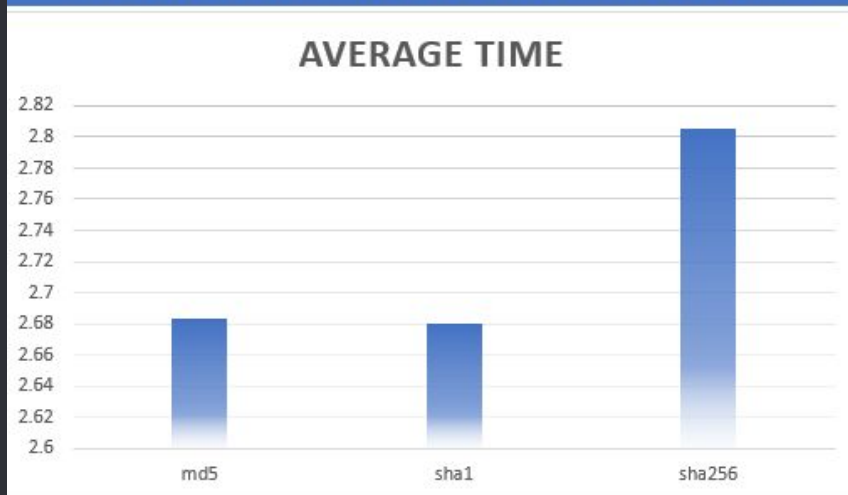
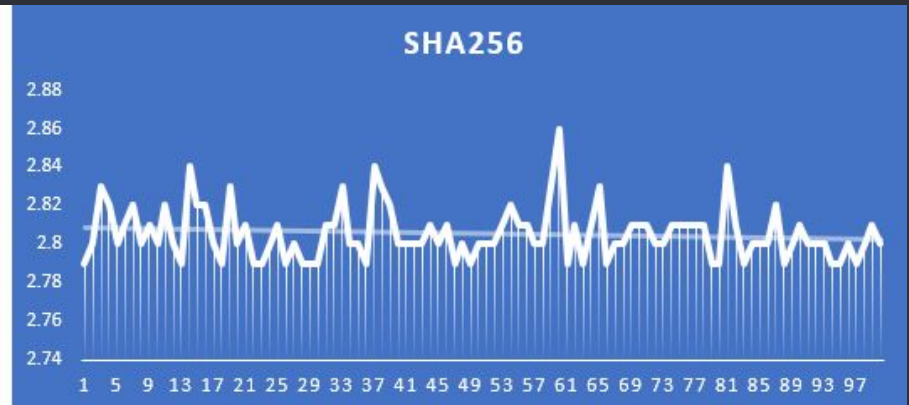
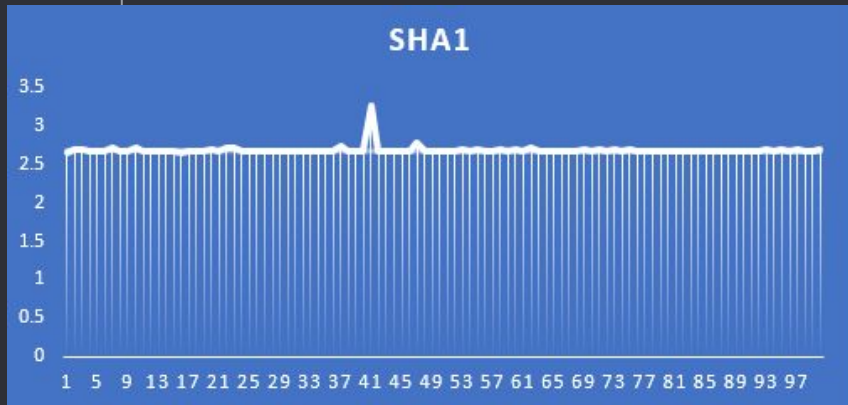
Research:

- MD5 and SHA-1 have both produced collisions.
- SHA-256 has not produced a known collision yet.
- Collisions pose a threat because an attacker may crack complex sensitive data without any guess as to what the data is.

Password Kraken:

- Dictionary attack - guess to crack
- Accommodates MD5, SHA-1, and SHA-256
- Displays statistics
- We aim to test whether or not passwords hashed in a secure algorithm take more time to crack.

● Our Findings - Nate



Conclusion - Sam

MD5 is considered broken for the use of security

- Collisions are likely to happen
- Using massive wordlist files is fast
- Computers are too powerful
- 1 Billion hashes in 100 seconds

SHA-1 -- Similar to MD5

- Uses similar resources compared to MD5
- One collision = Not a safe algorithm
- Google found one collision with a supercomputer in a few months
- 1 Billion hashes in 108 seconds

SHA-256

- Uses the most amount of resources
- Very efficient algorithm (same performance 1 Core vs. 8 cores)
- Computationally hard to crack
- 1 Billion hashes in 118 seconds

● TLDW

○ Secure hashing algorithm != use easy passwords

&

SHA-256 will only get us so far.



Q&A

