478 Final Project Proposal: Password Strength and Hash Cracking Analysis

Team Members: Aiden Lucero and Byron Lopez

Due Date: November 16th, 2025

Problem Statement:

Modern authentication systems continue to rely heavily on passwords as the primary method of user verification. However, users frequently create weak, reused, or predictable passwords that significantly undermine system security. Even when passwords are hashed, the use of outdated algorithms such as MD5 or SHA-1 leaves systems vulnerable to offline cracking attacks. This project aims to investigate how password complexity and hashing algorithm choices affect resistance to cracking. Using fully synthetic passwords, we will hash them with several algorithms and attempt to recover them using brute-force and dictionary attacks. By comparing performance metrics such as cracking time and success rate, we will demonstrate how stronger passwords and modern hash functions improve overall security. The broader goal is to highlight persistent weaknesses in password management and promote the adoption of stronger cryptographic and password hygiene practices in real systems.

Threat Models:

The main assets for this project are going to be the synthetic passwords, the hashed versions of those passwords, and the measurements that are collected from those cracking attempts. Since all the data is going to be synthetic/artificial and contains no real user information, the asset is a controlled environment for testing. The attacker considered in this model is just a simulated offline attacker who has gained access to the password hashes. This attacker will be able to use common cracking techniques that include brute-force and dictionary attacks to gain the original unhashed password. The main attack surface is the password hash list itself. In real-world situations, attackers normally obtain the hashed passwords rather than plaintext because offline attackers aren’t limited by lockouts, which makes the hash file the vulnerable point. For this project, the attack surface will be limited by the synthetic data. Assumptions we can make are that the attacker doesn’t have access to the original plaintext passwords, the hashing algorithms used are implemented correctly, no other advanced projections are assumed unless included, and the environment is offline, controlled for testing purposes. We will evaluate the password strength by comparing how long it takes for two different cracking methods to break weak versus strong passwords.

<u>**Success Metrics**</u>

The overall success of this project will be measured by how effectively the cracking tests indicate the differences in password strength and the overall hashing flexibility. Indications that will be used are cracking time comparison, cracking success rate, algorithm flexibility, password strength validation, and the consistency of results.

- Cracking Time Comparison: The time it takes to crack weak versus strong synthetic passwords. A successful cracking outcome will show the time gap that will prove why stronger passwords mean better security.
- Cracking Success Rate: The number of passwords that are successfully cracked through either brute-force or dictionary attacks. Lower success rates will indicate better security.
- Algorithm Resistance: The comparison in difficulty between hashing algorithms. Modern algorithms will take more time to crack, which shows they provide stronger protection.
- Password Strength Validation: Confirm that the weak patterns are cracked quickly, while the longer or more complicated passwords are able to withstand cracking attempts.
- Consistency of Results: Tests can produce reliable outcomes. Constituency shows that the “methodology is sound”.

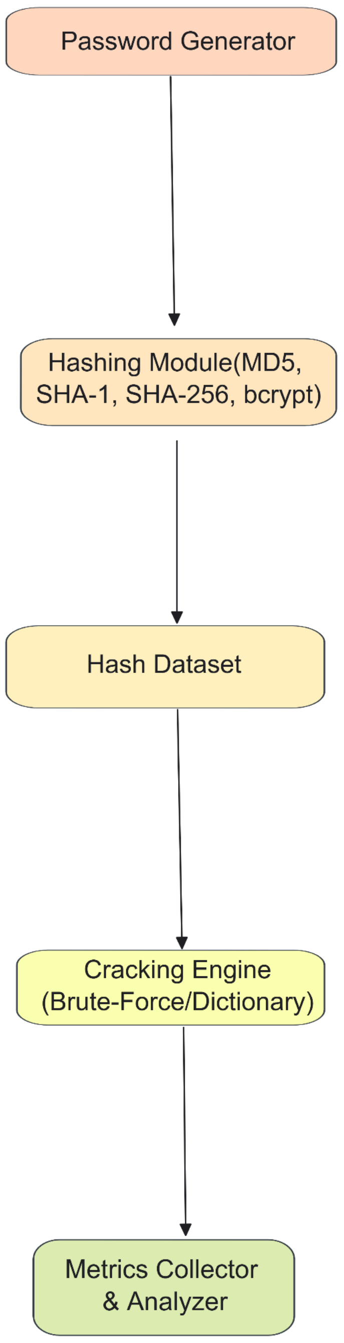
<u>**Dataset/PCAP plan:**</u>

All data used in this project will be synthetic to ensure ethical and legal compliance. A password generator script will produce passwords of varying complexity (weak, medium, strong) following defined patterns (like length, character diversity, randomness). These passwords will then be hashed using algorithms like SHA-1 and SHA-256 to compare legacy versus modern approaches. The dataset will contain plaintext–hash pairs and subsequent cracking results. This project focuses solely on offline password analysis rather than network traffic. Therefore, no PCAP files are necessary or will be collected. All generated data will be anonymized and stored securely within a controlled testing environment to ensure that the data is completely anonymized, safe to use, and acceptable for repeatable experiments.

Risk and Ethics:

This project is going to consist of password hashing and cracking techniques, which could lead to privacy violations, legal consequences, security risks, and ethical issues. To eliminate risks, all passwords and datasets are synthetic and generated exclusively for this project, and no real user credentials, leaked data, or third-party datasets will be used. All cracking activities will occur in an offline and isolated environment, ensuring no interaction with live systems. Results, hashes, and plaintexts will remain in a secure, controlled directory within the project repository. This approach adheres to ethical research principles and the ACM Code of Ethics, ensuring the project poses no privacy or security risks to individuals or systems. All this will ensure no one is harmed, no laws are violated, no privacy is violated, and no accounts are revealed.

Architecture Diagram:


Password Generator
Hashing Module(MD5, SHA-1, SHA-256, bcrypt)
Hash Dataset
Cracking Engine (Brute-Force/Dictionary)
Metrics Collector & Analyzer

Repository Skeleton Plan:

https://github.com/BIDL8588/478finalproj.git

Our repository will be structured to support reproducible experimentation:

- README.md: Project overview, setup instructions, and usage guide.
- LICENSE: Open-source license (MIT).
- CONTRIBUTING.md: Collaboration guidelines for both authors.
- docker-compose.yml: Defines separate containers for the hashing and cracking modules.