Evaluating Performance and Security Trade-offs in Modern Password Hashing Algorithms

Christian Chung
cc72574@eid.utexas.edu
UT Austin, TX

***Abstract*—Password hashing algorithms play a critical role in protecting user credentials in modern authentication systems. As data breaches continue to escalate and attackers gain access to increasingly powerful hardware, the security of stored passwords depends heavily on the choice and configuration of the hashing algorithm. The core problem is that traditional hashing algorithms such as SHA-256 are designed for speed, making them vulnerable to rapid brute-force attacks, while modern slow-hash algorithms offer stronger protection but require higher computational costs. This project aims to experimentally compare the performance of SHA-256, bcrypt, scrypt, and Argon2 on a consumer laptop to illustrate their security and performance trade-offs. For the initial report, the focus will be on conducting controlled experiments to measure hashing time, memory usage, and CPU-based brute-force resistance. The goal is to determine how algorithm choice alone can significantly strengthen password storage practices.**

I. Introduction

The increasing reliance on digital services has led to widespread storage of sensitive user information across the web. As a result, password database breaches have become a frequent and damaging occurrence, exposing millions of hashed passwords to attackers. In an era where commodity hardware can compute billions of hashes per second, the security of users' accounts depends not only on password strength but also on the choice of hashing algorithm used by service providers.

The central problem is that many systems still rely on fast hashing algorithms such as SHA-256 because they are easy to implement and computationally efficient. Unfortunately, this efficiency enables attackers to perform rapid brute-force attacks with minimal cost. Although stronger algorithms like bcrypt, scrypt, and Argon2 exist and are widely recommended, developers often lack a clear, practical understanding of how these algorithms differ in security and performance. This gap results in insecure deployments that fail to leverage available defenses.

This project introduces an experimental approach to quantify the performance and security differences among common password hashing algorithms. The key insight is that security can be improved not only through user behavior (e.g., stronger passwords) but also through algorithmic cost parameters that intentionally slow down hashing operations. By empirically demonstrating how hashing cost scales with algorithm design, the project provides actionable guidance for secure password storage.

To achieve this, the project evaluates four hashing algorithms—SHA-256, bcrypt, scrypt, and Argon2—on a standard consumer laptop. The experiments will measure time per hash across various parameters, memory requirements for memory-hard algorithms, and CPU-based brute-force time estimates for breaking passwords. The initial report will focus on the setup, methodology, and draft figures, while the final report will present actual measured data, comparisons, and insights.

The remainder of this proposal describes the motivation and prior work behind secure password hashing, the proposed experimental architecture, and the evaluation plan. The paper concludes with expected findings and the next steps for completing the project.

This project builds directly on the insights of these prior works by empirically testing the practical performance differences among these algorithms on modern hardware. The goal is not to design a new hashing algorithm, but to extend existing research by demonstrating these differences in a small-scale, reproducible environment suitable for educational purposes. Additional related work—including GPU-accelerated cracking, rainbow tables, and hardware-specific attack optimizations—will be covered in a later Related Work section so as not to disrupt the main narrative.

II. Motivation

A significant body of work has explored the weaknesses of traditional hashing algorithms in password security. Studies have demonstrated that fast algorithms like MD5, SHA-1, and SHA-256 can be computed billions of times per second using GPUs, making them unsuitable for password storage. The introduction of bcrypt (1999) marked a shift toward computationally expensive hashing, but subsequent research identified the need for memory-hard functions to defend against parallel hardware attacks, leading to the development of scrypt and the Argon2 family.

This project builds directly on the insights of these prior works by empirically testing the practical performance differences among these algorithms on modern hardware. The goal is not to design a new hashing algorithm, but to extend existing research by demonstrating these differences in a small-scale, reproducible environment suitable for educational purposes. Additional related work—including GPU-accelerated cracking, rainbow tables, and hardware-specific attack optimizations—will be covered in a later Related Work section so as not to disrupt the main narrative.

III. Proposed Design or Architecture

The system under evaluation consists of four password hashing implementations running on a single consumer laptop. The design centers around three core components:

1) Hashing Benchmark Module
   - Inputs: password strings, algorithm parameters
   - Outputs: average time per hash
   - Implements SHA-256 via hashlib, bcrypt, scrypt, and Argon2 via Python Libraries
2) Brute-force Simulation Module
   - Generates password candidates
   - Measures time required for a CPU-only brute force attempt
   - Supports both numeric and alphanumeric keyspaces
3) Analysis Engine
   - Aggregates results
   - Produces comparative metrics
   - Generates draft and final figures

Key Metrics to Optimize/Measure:

- Time per hash (ms)
- Scalability with increased cost parameters
- CPU and memory usage
- Estimated brute-force difficulty for each algorithm

These metrics will form the basis of the Evaluation section.

IV. Evaluation / Experimental Results

A. Hashing Time Comparison (Purpose)

The goal is to determine how hashing time scales across algorithms

- Axes:
  - X-axis: hashing algorithm
  - Y-axis: time per hash (ms)
- Lines:
  - Different cost parameters for bcrypt, scrypt, Argon2
- Trends (expected):
  - SHA-256 is near-instant
  - bcrypt increases logarithmically with cost
  - scrypt and Argon2 increase with memory/time settings
- Results: These trends highlight why slow hashing is necessary for password security.

Algorithm	Cost	Time per Hash (ms)
SHA-256	–	0.3
bcrypt	10	55
scrypt	N = 2^{14}	125
Argon2	m = 64MB	150

Table I
DRAFT TABLE.

B. CPU Brute-Force Resistance (Purpose)

The experiment estimates time required to brute-force 6-8 character passwords.

- Axes:
  - X-axis: algorithm
  - Y-axis: estimated crack time
- Lines:
  - Password length categories
- Trends (expected):
  - SHA-256 cracked in seconds/minutes
  - bcrypt, scrypt, Argon2 significantly slower
- Result: Algorithm choice is critical for real-world password security

Algorithm	Estimated Crack Time
SHA-256	Seconds
bcrypt	Hours
scrypt	Days
Argon2	Days/Weeks

Table II
DRAFT TABLE.

V. Related Work

Research on password hashing has primarily examined the design goals and theoretical properties of modern key-derivation functions. The Password Hashing Competition (PHC) final report provides the most comprehensive comparison of Argon2 against PBKDF2, bcrypt, and scrypt, emphasizing memory-hardness and resistance to parallel attacks rather than practical performance on consumer hardware [[BDK15]]. Earlier foundational works introduce PBKDF2 [[Kal00]], bcrypt [[PM99]], and scrypt [[Per09]], each focusing on security motivations and parameter choices, but offering limited empirical benchmarking across platforms.

More recent studies, such as Li's cross-platform library evaluation [[Li21]], explore performance differences between operating systems and implementations, though they focus on library inconsistencies rather than controlled algorithm-to-algorithm comparisons. Industry recommendations like the OWASP Password Storage Cheat Sheet [[Fou23]] provide parameter guidance but do not quantify runtime cost on typical laptops.

Overall, existing work provides theoretical and design-level insights but leaves a practical gap: how do widely used password hashing algorithms compare in performance under uniform, real-world conditions on commodity hardware? This project addresses that gap through direct benchmarking.

VI. Conclusions

What conclusions can you draw from your project – this is not a summary of the key results. Extrapolate into the future. Examine if your results lead to a re-evaluation of current tropes. The conclusions have to be interesting – ideally – instead of being a summary of some facts.

References

[BDK15] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. The password hashing competition: Final report. *PHC Technical Report*, 1(1):1–45, 2015.

[Fou23] OWASP Foundation. Owasp password storage cheat sheet. *OWASP Documentation Series*, 1(1):1–12, 2023.

[Kal00] Burt Kaliski. Pkcs #5: Password-based cryptography specification version 2.0. *RFC 2898*, 1(1):1–30, 2000.

[Li21] Xueyuan Li. Benchmarking password hashing libraries across platforms. *arXiv Preprint arXiv:2104.12356*, 1(1):1–10, 2021.

[Per09] Colin Percival. Stronger key derivation via sequential memory-hard functions. *scrypt Design Paper*, 1(1):1–20, 2009.

[PM99] Niels Provos and David Mazieres. A future-adaptable password scheme. *USENIX Annual Technical Conference Proceedings*, 1(1):81–91, 1999.