German University in Cairo
الجامعة الألمانية بالقاهرة

Faculty of Management Technology

Information Security (|BINF 711|)

Phase 2: Cryptography and Encryption Techniques

Submitted by:

Name: Ahmed Hassan	ID: 58-0671	Tutorial: 7
Name: Ziad Ekramy	ID: 58-6936	Tutorial: 7

Supervised by:

Dr. Wagdy Anis

TA Nadeen Hamza

Submission Date:

29/11/2025

Part 1: Password Encryption

Analysis of Password Based Key Derivation Function 2 (PBKDF2):

Password Based Key Derivation Function 2 (PBKDF2) is a widely trusted and well established method for securing stored passwords. It is used across many real systems, including major operating systems and authentication frameworks, because it provides strong protection against brute force attacks. The algorithm first uses a unique, randomly generated salt for every password, which prevents attackers from using pre-computed tables such as rainbow tables. Salt is a unique random value added to the password before hashing. PBKDF2 then applies a hashing function many thousands of times, a process known as key stretching. In our code, these iterations are set at 120,000. This makes each password guess computationally expensive for an attacker, even if they have powerful hardware. We chose to implement PBKDF2 because it is easy to implement on Python by using a built-in library, hashlib. Moreover our function is a reliable industry standard due to its simplicity and wide support.

Part 2: Research & Analysis

As technology advances, encryption techniques like Playfair & Vigenère become too easy to bypass. However, these continuous advancement led to improved encryption methods. In Part 1, we already talked about Password Based Key Derivation Function 2. In short, it combines the password with a unique salt, random value, then applies a hashing function given an iteration count. Iteration counts are usually in the hundreds of thousands; this process is known as key stretching. Another method is bcrypt. Even though it is considered a legacy hashing technique, bcrypt is heavily tested, has a long deployment history, and has good practical properties. Compared to modern alternatives, it has lower memory usage which gives it an advantage. Bcrypt is derived from the blowfish cipher. Moreover, we have scrypt. It was designed to add memory hardness to password hashing, which forces attackers to use significantly more RAM and CPU. As a result, this made large-scale, hardware accelerated cracking much more expensive compared with PBKDF2 or bcrypt. The final method we will talk about is Argon2. Argon 2 is the most recently studied winner of the password hashing competition in 2015. It was specifically

designed to be memory hard and tunable given three parameters, time cost, memory cost, and parallelism. When configured correctly, it provides the best protection against GPU or ASIC attackers. Argon2 is commonly recommended for new systems.

Let's talk about the limitations for each of these methods from the perspectives of computational overhead, susceptibility to attacks, and ease of implementation.

1. PBKDF2
    1.1. Computational overhead: It is configurable through adjustable iteration counts, allowing the hashing process to be intentionally slowed down on a CPU. By increasing the number of iterations, the algorithm requires more computation time for each hash.
    1.2. Susceptibility to attacks: It is not memory hard, so it is more vulnerable to GPU/ASIC acceleration.
    1.3. Ease of implementation: Easily implemented with built in Python libraries.
2. Bcrypt
    2.1. Computational overhead: Tunable via cost parameter; generally moderate CPU cost.
    2.2. Susceptibility to attacks: It is still not fully memory hard. Moreover, there are implementation vulnerabilities.
    2.3. Ease of implementation: Widely available across languages; simple APIs. Good real-world support but somewhat older design
3. Scrypt:
    3.1. Computational overhead: Tunable for both CPU and memory; can be configured to be expensive in both dimensions.
    3.2. Susceptibility to attacks: It can be susceptible to certain side-channel attacks and is vulnerable to misconfiguration.
    3.3. Ease of implementation: Slightly more complex than PBKDF2/bcrypt; libraries exist in most languages but it’s not in the standard library for many languages.

4. Argon2:
    4.1. Computational overhead: Tunable with time (iterations), memory, and parallelism; can be configured to be heavy in memory and CPU.
    4.2. Susceptibility to attacks: Best resistance to GPU/ASIC parallel attacks when parameters are chosen well.
    4.3. Ease of implementation: Good library support exists, but more parameters mean more care is required when selecting them; integration must be done carefully.

References:

Florencio, D., & Herley, C. (2007, May). A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web* (pp. 657-666).

Komanduri, S., Shay, R., Kelley, P. G., Mazurek, M. L., Bauer, L., Christin, N., ... & Egelman, S. (2011, May). Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 2595-2604).

Shay, R., Komanduri, S., Durity, A. L., Huh, P., Mazurek, M. L., Segreti, S. M., ... & Cranor, L. F. (2016). Designing password policies for strength and usability. *ACM Transactions on Information and System Security (TISSEC)*, *18*(4), 1-34.

Temoshok, D., Fenton, J., Choong, Y. Y., Lefkovitz, N., Regenscheid, A., & Richer, J. (2022). *Digital Identity Guidelines: Authentication and Lifecycle Management* (No. NIST Special Publication (SP) 800-63B-4 (Withdrawn)). National Institute of Standards and Technology.

Ur, B., Segreti, S. M., Bauer, L., Christin, N., Cranor, L. F., Komanduri, S., ... & Shay, R. (2015). Measuring {Real-World} accuracies and biases in modeling password guessability. In *24th USENIX Security Symposium (USENIX Security 15)* (pp. 463-481).

Weir, M., Aggarwal, S., De Medeiros, B., & Glodek, B. (2009, May). Password cracking using probabilistic context-free grammars. In *2009 30th IEEE symposium on security and privacy* (pp. 391-405). IEEE.