Comparison of Key Derivation Functions and Cipher Methods for Password Managers

14/06/2021

Lukasz Baldyga

LEEDS BECKETT UNIVERSITY

Table of Contents

1. Abstract....3
2. Key Derivation Functions....4
  2.1 PBKDF2....4
  2.2 Scrypt....4
  2.3 Argon2....4
3. Ciphers....6
  3.1 AES....6
  3.2 Blowfish....6
  3.3 Serpent....6
4. Asymmetric Cryptography....7
Conclusion....8
Bibliography....9

1. Abstract

This documents compares of different algorithms have been conducted between several encryption methods. It covers a vast range of citations and evaluates the effectiveness of key derivation functions and ciphers. It was concluded key derivation function will be Argon2 as the primary function with PBUDF2 as an alternative function.

2. Key Derivation Functions

In the password manager’s encryption, there are properties that key derivation functions need to have in order to be effective. The functions need to be expensive/slow enough so that they can prevent an adversary from cracking the encryption with brute force. This means also means being computationally expensive for other methods, like using Application Specific Integrated Circuits (also known as ASICs) and Graphical Processing Units (also known as GPUs).

2.1 PBKDF2

The PBUDF2 (B. Kaliski and RSA Laboratories, 2000) key derivation function is a widely used function used to encrypt disk data (cryptsetup, 2020), as well as password managers (Dan Callahan and Mozilla, 2014) (Bitwarden, 2021). It is the algorithm recommended by NIST (Meltem Sönmez Turan *et al.*, 2010).

The function’s cost can be updated and dynamically scaled depending on the usage. This means that as technology advances, the cost can be adjusted to be very expensive for the adversary, while keeping the user secure.

This fundamental flaw with this function is that it can be implemented as an ASIC and also accelerated by GPUs (Ondrej Mosnáˇcek, 2015). This is because of the low memory requirement by the algorithm.

However attractive this function might seem in terms of stability, it has fundamental flaws that cannot be mitigated effectively in a password manager, especially in the future.

2.2 Scrypt

Scrypt (Percival and Josefsson, 2016) is a recent algorithm. It is a “memory hard” (Colin Percival, 2009) algorithm, meaning that it requires a lot of memory to compute in order to try and be hardware resistant. The algorithm has been designed with ASIC computation threats in mind, whereas PBKDF2 is not. In the case of this algorithm, it is possible to adjust its memory and time costs simultaneously; as computational cost rises, memory costs also rise.

However, since Scrypt is a recent algorithm it is so not as heavily scrutinised as PBKDF2 or Bcrypt. It might not be as stable as the tried and tested PBKDF2.

Like PBKDF2, it shares the same fundamental flaw: it is not ASIC or GPU resistant (anymore). The problem with adjusting the memory and time cost simultaneously is that you cannot fine tune the algorithm.

2.3 Argon2

The winner of the Password Hashing competition in 2019 was the Argon2 algorithm (Password-Hashing.net, 2019). There exists a draft for the Argon2 function on the Internet Engineering Task Force (IETF)’s website (A. Biryukov *et al.*, 2021).

It allows for memory cost and time cost control independently, unlike Scrypt. This allows for greater control and also better ASIC and GPU resistance, which is the biggest threat to these algorithms.

The only problem that arises is that this is a very recent algorithm, much younger than Scrypt. This means that there hasn’t been much time for experts to give thorough scrutiny for this algorithm. There may be cryptanalysis that exists in the future for this specific algorithm. Regardless, this is also an algorithm worthy of consideration.

3. Ciphers

Ciphers need to have attributes that protect the users. A good cipher needs to produce an output that is indistinguishable from random, even though it is ciphertext. It also needs to have not been broken with cryptanalysis.

3.1 AES/Rijndael

The most well known and widely used, AES (also known as Advanced Encryption Standard and previously as Rijndael) (Nechvatal *et al.*, 1999) meets the criteria above. AES doesn’t show evidence of being fully cracked. It produces and output that is indistinguishable from random. Rijndael became the AES after the AES contest (Philip Bulman, 2000).

AES can take keys up to 32 bytes. Usually only 16 byte keys are secure enough for AES, however, due to the fact that quantum computers can half the brute force time in block chain ciphers (Lane Wagner, 2021) by using Grover’s algorithm (Grover, 1996), only using 32 byte keys is acceptable.

3.2 Blowfish

The Blowfish algorithm (B. Schneier, 1994) is a block chain cipher. It can take keys up to 48 bytes in size. Like AES, Blowfish does not show evidence of being fully cracked and also produces an output that is indistinguishable from random. Blowfish has been tried and tested given its age.

The fact that Blowfish can take 56 byte keys means that it might be more secure than AES when Grover’s algorithm (Grover, 1996) is applied.

3.3 Serpent

Serpent (Ross Anderson, Eli Biham, and Lars Knudsen, 1998) is a block chain cipher. It was a candidate for the Advanced Encryption Standard, and was ranked second to Rijndael. It was deemed to have a larger security margin than Rijndael (Nechvatal *et al.*, 1999).

Like any block chain cipher, Serpent can take keys up to 32 bytes and suffers from the same quantum attack as AES (Philip Bulman, 2000).

The most likely reasons the Serpent algorithm was not chosen is because it is significantly slower across all platforms (Nechvatal *et al.*, 1999) and the fact that it suffers from an attack when implemented in a smart cards (Chari *et al.*, 1999).

4. Asymmetric Cryptography

Methods of asymmetric cryptography would not be susceptible for this project's password manager. The password manager does not need to communicate with entities that they do not know. The data is stored locally and is never sent away. The requirement is for data to be secured **only** locally.

Furthermore, encrypting the data asymmetrically would add unneeded overhead and would be the ultimately the weakest link in the encryption for a capable attacker (for example: a government) with a quantum computer. This is because it is relatively trivial for quantum computers to factorise numbers into their prime factors (Lavor, Manssur and Portugal, 2003).

In summery, there is no sense in using asymmetric cryptography to encrypt things locally, and only brings downsides to security.

5. Conclusion

From the evidence presented here, the best hashing algorithm for this implementation is Argon2, as it allows for independent control of both memory and speed. It’s possible to benchmark the user’s device and find the best parameters for this algorithm. However, to be accommodate for more conservative or sceptical users, PBUDF2 will also be available as a hashing algorithm but will be discouraged.

For the cyphers, Serpent will be chosen as the default as this is not a smart card implementation and because AES has smaller security margins. The password manager will also allow for AES for conservative or sceptical users and Blowfish. Blowfish will be available due to the fact that it allows for bigger key sizes than AES or Serpent, making it more quantum computer resistant and more future proof.

6. Bibliography

A. Biryukov *et al.* (2021) ‘The memory-hard Argon2 password hash and proof-of-work function draft-irtf-cfrg-argon2-13’, 11 March. Available at: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-argon2 (Accessed: 12 June 2021).

B. Kaliski and RSA Laboratories (2000) ‘PKCS #5: Password-Based Cryptography Specification Version 2.0’, September. Available at: https://www.ietf.org/rfc/rfc2898.txt (Accessed: 13 June 2021).

B. Schneier (1994) ‘The Blowfish Encryption Algorithm’, *Dr. Dobb’s Journal*, 19(4), pp. 38–40.

Bitwarden (2021) ‘Encryption’. Available at: https://bitwarden.com/help/article/what-encryption-is-used/ (Accessed: 12 June 2021).

Chari, S. *et al.* (1999) ‘A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards’, in *In Second Advanced Encryption Standard (AES) Candidate Conference*, pp. 133–147.

Colin Percival (2009) ‘STRONGER KEY DERIVATION VIA SEQUENTIAL MEMORY-HARD FUNCTIONS’. Available at: https://www.tarsnap.com/scrypt/scrypt.pdf (Accessed: 13 June 2021).

cryptsetup (2020) ‘Cryptsetup and LUKS - open-source disk encryption’, *What the ...?*, 21 December. Available at: https://gitlab.com/cryptsetup/cryptsetup (Accessed: 14 January 2021).

Dan Callahan and Mozilla (2014) ‘Firefox Sync’s New Security Model’, *Mozilla Services*, 30 April. Available at: https://blog.mozilla.org/services/2014/04/30/firefox-syncs-new-security-model/ (Accessed: 12 June 2021).

Grover, L. K. (1996) ‘A Fast Quantum Mechanical Algorithm for Database Search’, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery (STOC ’96), pp. 212–219. doi: 10.1145/237814.237866.

Lane Wagner (2021) ‘Is AES-256 Quantum Resistant?’, *Qvault.io*, 10 June. Available at: https://qvault.io/cryptography/is-aes-256-quantum-resistant/ (Accessed: 14 June 2021).

Lavor, C., Manssur, L. R. U. and Portugal, R. (2003) ‘Shor’s Algorithm for Factoring Large Integers’, *arXiv e-prints*, p. quant-ph/0303175.

Meltem Sönmez Turan *et al.* (2010) ‘Recommendation for Password-Based Key Derivation Part 1: Storage Applications’. NIST. Available at: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf (Accessed: 12 June 2021).

Nechvatal, J. *et al.* (1999) ‘Status Report on the First Round of the Development of the Advanced Encryption Standard’, *Journal of Research of the National Institute of Standards and Technology*. 1999/10/01 edn, 104(5), pp. 435–459. doi: 10.6028/jres.104.027.

Ondrej Mosnáˇcek (2015) *Key derivation functions and their GPU implementation*. Masaryk University, Faculty of Informatics. Available at: https://is.muni.cz/th/409879/fi_b/?lang=en.

Password-Hashing.net (2019) ‘Password Hashing Competition’, *Password Hashing Competition and our recommendation for hashing passwords: Argon2*, 25 April. Available at: https://www.password-hashing.net/ (Accessed: 11 June 2021).

Percival, C. and Josefsson, S. (2016) *The scrypt Password-Based Key Derivation Function*. RFC Editor (Request for Comments, 7914). doi: 10.17487/RFC7914.

Philip Bulman (2000) ‘Commerce Department Announces Winner of Global Information Security Competition’, 2 October. Available at: https://www.nist.gov/news-events/news/2000/10/commerce-department-announces-winner-global-information-security (Accessed: 14 June 2021).

Ross Anderson, Eli Biham, and Lars Knudsen (1998) ‘Serpent: A Proposal for the Advanced Encryption Standard’. Available at: https://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf (Accessed: 14 June 2021).