Public Comments on SP 800-132, Recommendation for Password-Based Key Derivation: Part 1: Storage Applications

Comment period: January 6, 2023 – February 24, 2023

On January 6, 2023, NIST's Crypto Publication Review Board initiated a review of SP 800-132, *Recommendation for Password-Based Key Derivation: Part 1: Storage Applications* (December 2010). This document includes the public comments received during the comment period.

More details about this review are available from NIST's Crypto Publication Review Project site.

LIST OF COMMENTS

1. Comments from Levi Schuck, January 9, 2023

Hello,

I am Levi Schuck. My comments are submitted out of personal interest and not through any affiliation.

First, thank you for publishing accessible material. I often refer to NIST publications to learn the practical properties and consequences for cryptographic constructions.

With that in mind, I believe it would be helpful to define what "memory-hard" means. Additionally, I think "cache-hard" should be defined and considered in future publications.

In NIST SP 800-63B, BALLOON is cited as a memory-hard Password-based Key Derivation Function. I believe that SP 800-132 should have its scope increased to include other PBKDFs which are memory-hard and or cache-hard so that other publications like NIST SP 800-63B have one place to look for PBKDFs.

Also, I believe that BALLOON should not be referenced in any NIST publication. While it is an interesting academic publication, there is no specification, no test vectors, and the only existing code reference is an academic toy. There is ambiguity in what byte-order should be used inside and there is no means to reliably verify a compatible implementation in another software language or platform. Further, I am unable to find evidence that BALLOON is deployed anywhere of consequence due to its novelty and lack of availability. Please remove BALLOON.

Please consider referencing the community held "Password Hashing Competition" and its results. The Argon2 KDF has been standardized with the IETF as RFC 9106. There are implementations of Argon2 available across many software languages and platforms.

Password managers appear to use PBKDF2 in part due to platform limitations or compliance. If possible, please encourage adoption of a memory or cache-hard KDF in web platforms to replace PBKDF2. According to "Chick3nman" on twitter, PBKDF2-SHA256 with 100,100 iterations will run at 90,000 hashes per second on a single Nvidia RTX 4090, which is available to any individual on the market. With this speed, "Chick3nman" claims that a 14,000,000 password database can be tested in 165 seconds. See https://twitter.com/Chick3nman512/status/1606037277515325448

I encourage a re-evaluation of PBKDF2's iteration recommendations. Given the above, a minimum 1,000 iteration recommendation does not offer the protection today that it did in 2010. An example of the maximum iteration count is seen in the Apple "Backup Keybag" as documented on page 81 of https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf

The maximum count may need to be re-evaluated. Or if to be exceeded, the implementor is recommend to use an alternative memory-hard or cache-hard KDF.

Steve Thomas, a panelist for the "Password Hashing Competition", has provided some recommended hardness parameters on the web page https://tobtu.com/minimum-password-settings/ for Argon2 and PBKDF2. Should Argon2 be included in a NIST publication, please include recommended configurations for hardness parameters. Please consider reviewing Steve's recommendations when creating recommendations for NIST SP 800-132.

If other PBKDFs are considered, please take into account the complexity of the implementation. For example, scrypt is a complex construction that includes several different cryptographic primitives. This complexity increases the chances of a faulty or vulnerable implementation and deployment.

Thank you for reading my comments.

Levi Schuck

2. Comments from Mike Powers, January 20, 2023

Hello,

Regarding SP 800-132 PBKDF2, one “industry standard” usage of this KDF is within WPA2-PSK. In short, when a user sets a passphrase – if it’s not exactly 256 bits in length, then the PMK is produced via PBKDF2 as follows:

PMK = PBKDF2(HMAC−SHA1, passphrase, SSID, 4096, 256)

This is the method of deriving the PMK that is suggested in IEEE 802.11i-2004, section H.4, and it is used in many networking devices that implement/use WPA2-PSK.

As it stands now, this is not an Approved usage of PBKDF2 for a few reasons:

1. FIPS 140-2 IG D.6 / FIPS 140-3 IG D.N; there is the following statement (which is consistent with the title of the special publication “Part 1: Storage Applications”):

a. “Further, the vendor shall indicate in the module’s Security Policy that keys derived from passwords, as shown in SP 800-132, may only be used in storage applications”

2. Also - In SP800-132, there is the following statement:

a. “All or a portion of the salt shall be generated using an approved Random Bit Generator (e.g., see [5]). The length of the randomly-generated portion of the salt shall be at least 128 bits.”

For (2) that could be enforced by implementations by forcing the SSID (which is used as the salt) to be randomly-generated. For (1), however, that seems to currently just be an inherent flaw in 802.11i-2004 where it conflicts with the intended usage of PBKDF2. Based on this, my comment is as follows:

-Consider updating the SP 800-132 standard to specifically allow for its usage in the context of WPA2-PSK

Thanks,

Mike Powers

Leidos CSTL Technical Director

3. Comments from Canadian Center for Cyber Security, January 26, 2023

Section 3.1:
Include definitions of a block (used in Iteration count), symmetric cryptographic key (used in Keying material and MAC), include MAC as abbreviation in its definition or in 3.2 or avoid using abbreviation in Authenticated Encryption definition.

Section 3.2:
GCM is never used in the text.

Section 5.3:
Since NIST recently retired SHA-1, a note here regarding that might be appropriate.
Function Int is defined with capital I, but appears with lower case i in Figure 2.

References need to be updated:
NIST SP 800-108 needs to change to be NIST SP 800-108 Rev 1, RFC 2898 was obsoleted by RFC 8018, NIST SP 800-90 should be replaced with NIST SP 800-90 Rev 1. Also, NIST SP 800-38D might need to be updated soon as it’s currently under review.

Appendix B: it should be mentioned that ACVP testing is available for this algorithm.

4. Comments from S. Arciszewski, E. Crocket, P.Kampanakis, AWS, February 21, 2023

Dear NIST Review Board,

We would like to provide some feedback regarding the Request for Public Comments on SP 800-132.

SP 800-132, as described, explains how to derive cryptographic keying material from a password or a passphrase to protect stored electronic data or data protection keys. Traditionally, these algorithms can also be used to store protected passwords for password authentication. This document should state that as well to benefit users which look for ways to store their password securely.

Additionally, SP800-63B is the NIST SP that addresses password hashing. We suggest that SP800-63B's section 5.1.1.2 should reference SP 800-132 and not include Balloon hashing. As far as we know, Balloon was never standardized, and there's very little academic research on it.

We also have some comments regarding industry needs for new password-based standards, including memory-hard password-based key derivation functions and password hashing schemes:

We want to suggest that NIST considers including scrypt in the SP. scrypt improves upon PBKDF2 by adding tunable memory hardness of launching a dictionary attack. This now-standard best-practice means that specialized hardware cannot be used to speed up dictionary attacks against scrypt. By contrast, PBKDF2, which is approved in the current SP, only has an iteration count parameter with constant memory footprint. Also, scrypt is built on PBKDF2 and the stream cipher Salsa20/8. Salsa20/8 could be replaced with another stream cipher if desired. Given that the SP already approves PBKDF2, scrypt would be relatively straightforward to approve.

We want to suggest NIST considers including Argon2 in the SP because Argon2 was declared the winner of a community-organized NIST-style competition for password hashing algorithms. It is based on the Blake2 hash algorithm. Blake2 could be substituted by another hash function in the SHA-3 family if desired by NIST. Argon2 allows users to scale the time and memory complexity of the hash which protects better than PBKDF2 from specialized hardware dictionary attacks. Argon2 also provides better resistance to side-channel attacks than scrypt.

Regards,

Scott Arciszewski, Eric Crocket, Panos Kampanakis

AWS

5. John Mattsson, Ericsson AB, February 22, 2023

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. Please find attached our comments SP 800-132.

Best Regards,
John Preuß Mattsson

ERICSSON

Date: February 22, 2023

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on SP 800-132, Recommendation for Password-Based Key Derivation Part 1: Storage Applications

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents.

Please find below our comments on SP 800-132:

– We think SP 800-132 should be expanded with algorithms approved for hashing a password or passphrase for safe storage for authentication purposes. SP 800-63B [1] already refers to SP 800-132 for this use case.

– To mitigate against password cracking on GPUs, FPGAs, and ASICs we think SP 800-132 should be expanded with one or more NIST approved side-channel resistant memory-hard or cache-hard password-based key derivation function such as Argon2id [2], Balloon [3], or bscrypt [4]. Argon2id is the recommendation from the Open Worldwide Application Security Project (OWASP) [5] but does not use a NIST approved hash function. Balloon is suggested by SP 800-63B [1]. We think SP 800-132 should recommend the use of memory-hard or cache-hard algorithms instead of PBKDF2 [6] for all use cases.

– "*A minimum iteration count of 1,000 is recommended.*"

  This should be significantly increased. The current recommendation from OWASP [5] is to use an iteration count of 600,000 for PBKDF2 with SHA-256 and an iteration count of 1,300,000 for PBKDF2 with SHA-1.

– "*such as one of the modes that is defined in [3,4]*"

  In addition to AES-GCM and AES-CCM, we suggest to also add a reference to SP 800-38F [7] that defines AES-KW and AES-KWP.

– *"The following algorithm for the derivation of MKs from passwords is based on an algorithm specified in [6], where it was specified as PBKDF2 and used HMAC [1] with SHA-1 as a PRF. This Recommendation approves PBKDF2 as the PBKDF using HMAC with any approved hash function as the PRF."*

  RFC 2898 has been obsoleted by RFC 8018 [6] which introduces SHA-2 as an alternative.

The differences (if any) between the algorithm in Section 5.3 of SP 800-132 and the PBKDF2 algorithm specified in RFC 8018 should be described. We think NIST should remove the abbreviation PBKDF and only talk about PBKDF2. The abbreviation PBKDF is too associated with the PKCS #5 algorithms to be suitable as a general term.

- "*Dictionary attacks aim to recover passwords by trying the most-likely strings, such as the words in a dictionary. These attacks are less likely to succeed against passwords that include random combinations of upper/lowercase letters and numeric values. Such passwords can only be recovered using inefficient brute force attacks that try all possible passwords.*"

  We think this should be rewritten to better reflect modern methods for password cracking. Attackers typically generate likely passwords based on previous breach corpuses, details about the target, and sophisticated models on how humans create passwords.

- SP 800-132 should be aligned with SP 800-63B [1]:

  - SP 800-63B states that PBKDF2 as specified in SP 800-132 is a suitable algorithm for hashing a password for safe storage for authentication purposes. SP 800-132 states that PBKDF2 shall not be used for any other purposed than to generate data protection keys or intermediate keys to protect data protection keys. We suggest that the update to SP 800-132 approves the use of PBKDF2 for hashing a password or passphrase for safe storage for authentication purposes.

  - SP 800-63B states that the salt shall be at least 32 bits. SP 800-132 states that the randomly-generated portion of the salt shall be at least 128 bits.

  - In addition to a salt, SP 800-63B also recommends the use of a pepper, i.e., using an additional secret salt value known only to the verifier and stored separately (e.g., in an HSM). We think SP 800-132 should recommend peppering for all use cases. The pepper could be unlocked by multifactor authentication.

  - SP 800-63B requires that when processing requests to establish and change memorized secrets, the prospective secrets shall be compared against a list that contains values known to be commonly-used, expected, or compromised. We think SP 800-132 should require this as well. The U.S. Department of the Interior report [8] showed that between 20 and 40 percent of their passwords could be easily cracked.

- SP 800-132 and 800-63B should add some discussion of when the mechanisms in SP 800-132 and 800-63B are acceptable to use instead of a password-authenticated key exchange (PAKE) such as the augmented PAKE OPAQUE [9] and the balanced PAKE CPace [10] that are recommended by the Crypto Forum Research Group (CFRG). As PAKEs are a better solution in many situations it would be good to have NIST approved PAKEs.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

[1] SP 800-63B, "Digital Identity Guidelines Authentication and Lifecycle Management"
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf

[2] RFC 9106, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications"
https://datatracker.ietf.org/doc/html/rfc9106

[3] Boneh et al., "Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks"
https://eprint.iacr.org/2016/027

[4] Steve Thomas, "bscrypt a cache hard password hash/KDF"
https://github.com/Sc00bz/bscrypt

[5] OWASP, "Password Storage Cheat Sheet"
https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[6] RFC 8018, "PKCS #5: Password-Based Cryptography Specification Version 2.1"
https://datatracker.ietf.org/doc/html/rfc8018

[7] SP 800-38F, "Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping"
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

[8] U.S. Department of the Interior, "P@s$w0rds at the U.S. Department of the Interior: Easily Cracked Passwords, Lack of Multifactor Authentication, and Other Failures Put Critical DOI Systems at Risk"
https://www.doioig.gov/sites/default/files/2021-migration/Final%20Inspection%20Report_DOI%20Password_Public.pdf

[9] IRTF, "The OPAQUE Asymmetric PAKE Protocol"
https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-opaque

[10] IRTF, "CPace, a balanced composable PAKE"
https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-cpace

6. Jeremiah Blocki, Purdue University, February 24, 2023

In the past decade data-breaches have exposed billions of user passwords to the dangerous threat of offline brute-force cracking. It has been well documented that a large fraction of user passwords have lower entropy and are potentially vulnerable to offline attacks. An offline attacker is limited only by the resources that s/he is willing to invest trying to crack each user password. Key-stretching is an important defense to help deter an offline attacker by increasing guessing costs.

While PBKDF2 does increase costs for an offline attacker the approach is outdated. We applied a decision theoretic framework to analyze several recent password breaches[1]. One of our findings was that PBKDF2 (and other hash iteration based techniques for key-stretching) provide insufficient protection for most user passwords. There are several reasons for this finding. First, a function such as PBKDF2 is not egalitarian meaning that an attacker using customized hardware (e.g., FPGAs or ASICs) could potential reduce guessing costs by several orders of magnitude. Second, the iteration count is constrained by user patience since we require that an honest party can evaluated the sequential function quickly during user authentication. Given that many user passwords are weak it was simply not possible to provide sufficient levels of key-stretching to deter a rational attacker without introducing an unacceptably long authentication delay. By contrast, we found that memory hard functions (e.g., SCRYPT, Argon2id, DRSample) could provide a meaningful deterrent to dissuade a rational offline password attacker from cracking many user passwords [1]. One of the reasons is that the amortized space-time complexity of a memory hard function can scale quadratically in the running time parameter n i.e., ideally we can force the attacker to lock up n blocks of memory for n sequential steps. By contrast, the cost of PBKDF2 scales linearly in the running time parameter n.

I would highly recommend that NIST make an effort to promote the usage of memory-hard functions for password hashing whenever possible. One candidate for standardization is Argon2, the winner of the password hashing competition in 2015. However, while Argon2 is certainly much stronger than PBKDF2 it is also likely that we can develop even stronger constructions. Scientific understanding of memory hard functions has advanced significantly since 2015 e.g., see [2-13]. We now have efficient pebbling attacks on Argon2i (the side-channel resistant version of Argon2 which was originally recommended for password hashing) demonstrating that Argon2i (along with several other iMHF candidates) is less memory hard than we might hope [2,3,13]. On the positive side we have also seen the development of improved iMHFs [6,7] with provable memory-hardness. Several of these provably secure iMHFs candidates already have proof-of-concept implementations [4,5].

[1] Blocki, J., Harsha, B. and Zhou, S. On the Economics of Offline Password Cracking. IEEE S&P 2018.

[2] Alwen, J. and Blocki, J. Efficiently Computing Data-Independent Memory Hard Functions. CRYPTO 2022.

[3] Alwen, J. and Blocki, J. Towards Practical Attacks on Argon2i and Balloon Hashing. EURO S&P 2017.

[4] Alwen, J., Blocki, J. and Harsha, B. Practical Graphs for Optimal Side-Channel Resistant Memory-Hard Functions. CCS 2017.

[5] Blocki, J., Harsha, B., Kang, S., Lee, S., Xing, L. and Zhou, S. Data-Independent Memory Hard Functions: New Attacks and Stronger Constructions. CRYPTO 2019.

[6] Blocki, J. and Holman, B. Sustained Space and Cumulative Complexity Trade-Offs for Data-Dependent Memory-Hard Functions. CRYPTO 2022.

[7] Alwen, J., Blocki, J. and Pietrzak, K. Depth-Robust Graphs and Their Cumulative Memory Complexity. EUROCRYPT 2017

[8] Alwen, J., Blocki, J. and Pietrzak, K. Sustained Space Complexity. EUROCRYPT 2017.

[9] Ren, L. and Devadas, S. Bandwidth-Hard Functions for ASIC Resistance. TCC 2017.

[10] Blocki, J., Ren, L., and Zhou, S. Bandwidth-Hard Functions: Reductions and Lower Bounds. CCS 2018.

[11] Chen, B. and Tessaro, S. Memory-Hard Functions from Cryptographic Primitives. CRYPTO 2019.

[12] Alwen, J., Chen, B., Pietrzak, K., Reyzin, L. and Tessaro, S. SCRYPT is Maximally Memory Hard. EUROCRYPT 2017.

[13] Blocki, J. and Zhou, S. On the Depth-Robustness and Cumulative Pebbling Cost of Argon2i. TCC 2017.

7. Sophie Schmieg, Stefan Kölbl, February 25, 2023

We strongly suggest the addition of memory hard password hash functions to NIST SP 800-132, given the current state of the art. In particular, inclusion of Argon2 (RFC 9106), specifically the primary variant argon2id, and potentially scrypt (RFC 7914) in this standard would strengthen NIST's portfolio when it comes to password derived keys/slow hash functions. It would also address the fact that PBKDF2 is no longer considered appropriate for password hashing in most cases due to its lack of resistance against ASICs.

8. Steve Thomas, February 25, 2023

We strongly suggest the addition of memory hard password hash functions to NIST SP 800-132, given the current state of the art. In particular, inclusion of Argon2 (RFC 9106), specifically the primary variant argon2id, and potentially scrypt (RFC 7914) in this standard would strengthen NIST's.

Main points to consider:

1) Deprecate PBKDF2 in favor of a similar computationally-hard algorithm that allows for a defender to use SIMD and threads (such as "Parallel PBKDF2" https://github.com/Sc00bz/ppbkdf2).

2) Recommend a memory-hard algorithm (such as Argon2 or scrypt) and/or a cache-hard algorithm (such as bscrypt). But don't recommend using Balloon Hashing until the details of the algorithm are finalized and there are test vectors.

3) Base minimum PBKDF settings on attacker's cracking speed (see https://tobtu.com/minimum-password-settings/ for my current minimum settings which will be updated based on the latest GPUs).

4) State that the PBKDF should only run once for each time a user enters their password. Also when outputting a large amount with PBKDF2 then splitting that up into separate keys is equivalent to running PBKDF2 multiple times and should not be allowed.

----

1) PBKDF2 should be deprecated in favor of a similarly computationally-hard algorithm that allows for a defender to use SIMD (Single Instruction Multiple Data) and threads. The reason we need an approved parallel computationally-hard algorithm is all new x86 CPUs have at least AVX2 which can do 8, 32 bit operations in parallel and most have at least 4 cores. That's a 32x speed up that PBKDF2 can't take advantage of. This means a defender can run the equivalent of PBKDF2-HMAC-SHA-256 with 320,000 iterations in the same amount of time as PBKDF2-HMAC-SHA-256 with 10,000 iterations. Also GPUs and FPGAs can be used by a defender which increases speed by ~10x vs a CPU.

"Parallel PBKDF2" takes advantage of PBKDF2's footgun. PBKDF2's footgun is that each block of hash length bytes of output is iterated and calculated independently of other blocks. Thus can be calculated with SIMD and threads. For more info see https://github.com/Sc00bz/ppbkdf2.

2) Recommend a cache-hard algorithm (such as bscrypt) and/or a memory-hard algorithm (such as Argon2 or scrypt). The difference between cache-hard and memory-hard algorithms is cache-hard algorithms' cracking speed is limited by memory transactions and memory-hard algorithms' cracking speed is limited by memory bandwidth. Memory-hard algorithms also can use enough memory that the attacker can't utilize all the available memory bandwidth. Cache-hard algorithms start out ahead of memory-hard algorithms because GPUs access memory with large bus widths usually 256 bits or more. Reading only 64 bits wastes most of the available memory bandwidth for an attacker. But there is a point where memory-hard algorithms use enough memory that they beat cache-hard algorithms.

All cache-hard algorithms, Argon2d, Argon2id, and scrypt do secret dependant reads which means NIST will likely disapprove these. That leaves Argon2i which doesn't do secret dependant reads. For Argon2i, time cost should be at least 3 to prevent some attacks. Balloon Hashing also doesn't do secret dependant reads but this is more of an idea of how to do a memory-hard algorithm and leaves the details up to the implementer. Thus Balloon Hashing should not be recommended until the details of the algorithm are finalized and there are test vectors.

3) The minimum iterations for PBKDF2 are stated as 1000 irrespective of the hash function. The minimum PBKDF settings should be based on attacker's cracking speed (see https://tobtu.com/minimum-password-settings/ for limiting an attacker's cracking speed to less than 10,000 guesses/second/GPU). NIST should pick a maximum speed an attacker should be able to achieve with the current best GPU for password cracking. The origins of less than 10,000 guesses/second/GPU came from password cracking expert Jeremi Gosney (https://arstechnica.com/information-technology/2015/06/hack-of-cloud-based-lastpass-exposes-encrypted-master-passwords/):

"On an NVIDIA GTX Titan X, which is currently the fastest GPU for password cracking, an attacker would only be able to make fewer than 10,000 guesses per second for a single password hash. That is proper slow! Even weak passwords are fairly secure with that level of protection (unless you're using an absurdly weak password.)"

4) In section 4, state that the PBKDF should only run once for each time a user enters their password. Note the statement "The MK shall not be used for other purposes." would require the PBKDF to be run multiple times for encrypted cloud storage. Once for generating an authentication key and again for encryption key(s). NIST should add the ability to use the MK to derive keys for authenticating to a server. Also when outputting a large amount with PBKDF2 then splitting that up into separate keys is equivalent to running PBKDF2 multiple times and should not be allowed.

It is a common mistake to run the PBKDF twice when storing password encrypted data in the cloud: once to authenticate to the server and once for an encryption key. Also NIST should state that the best way to authenticate to the server is with a PAKE (such as "(strong) AuCPace" or BS-SPEKE), but this is probably out of scope for SP 800-132.

An example of running the PBKDF multiple times is "Nigori: Storing Secrets in the Cloud" (see https://www.cl.cam.ac.uk/~drt24/nigori/nigori-protocol.html#anchor6). Nigori was deployed at Google for Chrome's Sync. This ran PBKDF2 four times and an attacker only needed to run it once. Thus they could of increased the settings by 4x with no slow down to the user.

Steve Thomas

https://tobtu.com/