Common Authentication Technology Next Generation
Internet-Draft
Intended status: Best Current Practice
Expires: 26 October 2026

S. Whited

24 April 2026

Best practices for password hashing and storage
draft-ietf-kitten-password-storage-10

Abstract

This document outlines best practices for handling user passwords and other authenticator secrets in client-server systems making use of SASL.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 26 October 2026.

Table of Contents

1. Introduction

Following best practices when hashing and storing passwords for use with SASL impacts a great deal more than just a user's identity. It also affects usability, backwards compatibility, and interoperability by determining what authentication and authorization mechanisms can be used.

1.1. Conventions and Terminology

Various security-related terms are to be understood in the sense defined in [RFC4949]. Some may also be defined in [DOI_10.6028_NIST.SP.800-63-4] Appendix B, [DOI_10.6028_NIST.SP.800-63B-4] Appendix D, and in [NIST_SP_800_132] section 3.1.

Throughout this document the term "password" is used to mean any password, passphrase, PIN, or other memorized secret.

Other common terms used throughout this document include:

Mechanism pinning A security mechanism which allows SASL clients to

resist downgrade attacks. Clients that implement mechanism pinning remember the perceived strength of the SASL mechanism used in a previous successful authentication attempt and thereafter only authenticate using mechanisms of equal or higher perceived strength.

Pepper A secret added to a password hash like a salt. Unlike a salt, peppers are secret and the same pepper may be reused for many hashed passwords.

Salt In this document salt is used as defined in [RFC4949].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. SASL Mechanisms

For clients and servers that support password based authentication using SASL [RFC4422] it is RECOMMENDED that the following mechanisms be implemented:

* SCRAM-SHA-256 [RFC7677]

* SCRAM-SHA-256-PLUS [RFC7677]

System entities SHOULD NOT invent their own mechanisms that have not been standardized by the IETF or another reputable standards body. Similarly, entities MUST NOT implement any mechanism with a usage status of "OBSOLETE", or "LIMITED", or "MUST NOT be used" in the IANA SASL Mechanisms Registry [IANA_sasl_mechanisms_sasl_mechanisms_1]. For example, entities MUST NOT implement DIGEST-MD5 (deprecated in [RFC6331]).

The SASL mechanisms discussed in this document do not negotiate a security layer. Because of this a strong security layer such as TLS [RFC8446] MUST be negotiated before SASL mechanisms can be advertised or negotiated.

3. Client Best Practices

3.1. Mechanism Pinning

Clients often maintain a list of preferred SASL mechanisms, generally ordered by perceived strength to enable strong authentication. To prevent downgrade attacks by a malicious actor that has successfully executed an in-the-middle attack on a connection, or compromised a trusted server's configuration, clients SHOULD implement "mechanism pinning". That is, after the first successful authentication with a strong mechanism, clients SHOULD make a record of the authentication and thereafter only advertise and use mechanisms of equal or higher perceived strength.

The following mechanisms are ordered by their perceived strength from strongest to weakest with mechanisms of equal strength on the same line. The remainder of this section is merely informative. In particular this example does not imply that mechanisms in this list should or should not be implemented.

1. EXTERNAL

2. SCRAM-SHA-256-PLUS

3. SCRAM-SHA-1-PLUS

4. SCRAM-SHA-256

5. SCRAM-SHA-1

6. PLAIN

The EXTERNAL mechanism defined in [RFC4422] appendix A is placed at the top of the list. However, its perceived strength depends on the underlying authentication protocol. In this example, we assume that TLS [RFC8446] services are being used.

The channel binding ("-PLUS") variants of SCRAM [RFC5802] are listed above their non-channel binding cousins, but may not always be available depending on the type of channel binding data available to the SASL negotiator.

Finally, the PLAIN mechanism sends the username and password in plain text and therefore requires a strong security layer such as TLS for the password to be protected in transit. However, if the server is trusted to know the password PLAIN does allow for the use of a strong key derivation function (KDF) for storing the authentication data at rest and provides for password hash agility.

3.2. Storage

Clients SHOULD always store authentication secrets in a trusted and encrypted keystore such as the system keystore, or an encrypted store created specifically for the clients use. They SHOULD NOT store authentication secrets as plain text.

If clients know that they will only ever authenticate using a mechanism such as SCRAM [RFC5802] where the original password is not needed after the first authentication attempt they SHOULD store the SCRAM bits or the hashed and salted password instead of the original password. However, if backwards compatibility with servers that only support the PLAIN mechanism or other mechanisms that require using the original password is required, clients MAY choose to store the original password so long as an appropriate keystore is used.

4. Server Best Practices

4.1. Additional SASL Requirements

Servers MUST NOT support any mechanism that would require authentication secrets to be stored in such a way that they could be recovered in plain text from the stored information. This includes mechanisms that store authentication secrets using reversable encryption, obsolete hashing mechanisms such as MD5 or hashing mechanisms that are cryptographically secure but designed for speed such as SHA256.

4.2. Storage

Servers MUST NOT store passwords in plain text. Instead, servers MUST always store passwords only after they have been salted and hashed using a strong KDF. A distinct salt SHOULD be used for each user, and, in the case of SCRAM, for each SCRAM family supported. Salts and peppers SHOULD be generated using a cryptographically secure random number generator. The salt MAY be stored in the same datastore as the password. A pepper SHOULD be combined with the password before hashing if possible with the given authentication mechanism. Peppers MAY be re-used for all passwords, and SHOULD have a rotation mechanism in case of compromise. The pepper MUST NOT be stored in the same datastore as the hashed passwords or salts and SHOULD be stored in an appropriate secret store such as a keystore or HSM. Similarly, peppers SHOULD NOT be combined with the salt because the salt is not secret and may appear in the final hash output.

The following restrictions MUST be observed when generating salts and peppers:

Parameter	Value
Minimum Salt Length	16 bytes
Minimum Pepper Length	14 bytes

Table 1: Common Parameters

It is common practice to prefix the randomly generated portion of a salt with a human-readable statement of purpose. The minimum salt length requirement above applies only to the randomly generated portion of the salt, not to the entire value.

4.3. Authentication and Rotation

When authenticating using PLAIN or similar mechanisms that involve transmitting the original password to the server the password MUST be hashed and compared against the salted and hashed password in the database using a constant time comparison.

Each time a password is changed a new random salt MUST be created and the iteration count and pepper (if applicable) MUST be updated to the latest value required by server policy.

If a pepper is used, consideration should be taken to ensure that it can be easily rotated. For example, multiple peppers could be stored. New passwords and reset passwords would use the newest pepper and a hash of the pepper using the same KDF that was used on the password could then be stored in the database next to the salt so that future logins can identify which pepper in the list was used. This is just one example, pepper rotation schemes are outside the scope of this document.

5. KDF Recommendations

When properly configured, the following commonly used KDFs create suitable password hash results for server side storage. The recommendations in this section may change depending on the hardware being used and the security level required for the application.

With all KDFs proper tuning is required to ensure that it meets the needs of the specific application or service. For persistent login an iteration count or work factor that adds approximately a quarter of a second to login may be an acceptable tradeoff since logins are relatively rare. By contrast, verification tokens that are generated many times per second may need to use a much lower work factor. The

recommendations in the following tables represent a minimum iteration count and SHOULD be set as high as can be tolerated for the application. For example, the iteration count required to verify a user unlocking an encrypted harddrive during a cold boot of a computer may take over a second without causing adverse delays for the user, while a user logging in to an instant messaging application may begin to notice the delay if a full second were taken to log in.

The recommendations in this section are likely to change over time as CPUs and GPUs become more powerful. These recommendations are meant to be the current best practices at the time of this documents publication, but additional tuning will likely be required to ensure safety going forward.

5.1. Argon2

Argon2 is the 2015 winner of the Password Hashing Competition and the current OWASP recommendation for secure password storage. Security considerations, test vectors, and parameters for tuning argon2 can be found in [RFC9106]. Its use is RECOMMENDED for all new password storage implementations.

Parameter	Value
Degree of parallelism (p)	1
Minimum memory size (m)	2 GiB
Minimum number of iterations (t)	2
Algorithm type (y)	Argon2id (2)
Minimum output length	32

Table 2: Argon Parameters

5.2. PBKDF2

PBKDF2 [RFC8018] is used by the SCRAM [RFC5802] family of SASL mechanisms.

For other password storage systems its use is RECOMMENDED only when FIPS-140 [NIST_FIPS_140_3] compliance is required.

Parameter	Value
Minimum iteration count (c)	600,000
Hash	HMAC-SHA256
Output length (dkLen)	min(hLen, 32) (where hLen is the length of the chosen hash)

Table 3: PBKDF2 Parameters

When PBKDF2 is used with HMAC such as in the SCRAM [RFC5802] family of SASL mechanisms the password is pre-hashed if it is longer than the block size of the hash function (hLen, or 64 bytes for SHA-256). Care should be taken to ensure that the implementation of PBKDF2 does this before the iterations, otherwise long hashes may become significantly more expensive than expected, possibly resulting in a Denial-of-Service (DOS).

5.3. Scrypt

The SCRYPT KDF is designed to be memory-hard and sequential memory-hard to prevent against custom hardware based attacks. Its use is RECOMMENDED only when Argon2 is not available.

Security considerations, test vectors, and further notes on tuning scrypt may be found in [RFC7914].

Parameter	Value
Minimum CPU/Memory cost parameter (N)	131072 (N=2^17)
Blocksize (r)	8 (1024 bytes)
Parallelization parameter (p)	1
Minimum output length (dkLen)	32

Table 4: Scrypt Parameters

5.4. Bcrypt

bcrypt [BCRYPT] is a Blowfish-based hashing function. Though it is not a KDF like the other options in this list, it has been commonly used for password storage in the past. Its use is NOT RECOMMENDED for new password storage implementations, but for legacy systems the following parameters SHOULD be used:

Parameter	Value
Minimum Recommended Cost	12
Maximum Password Length	50-72 bytes depending on the implementation

Table 5: Bcrypt Parameters

6. Internationalization Considerations

The PRECIS (Preparation, Enforcement, and Comparison of Internationalized Strings) framework [RFC8264] is used to enforce internationalization rules on strings and to prevent common application security issues arrising from allowing the full range of Unicode codepoints in usernames, passwords, and other identifiers. The "OpaqueString" profile of [RFC8265], Section 4.2 SHOULD be applied to passwords to ensure that codepoints in passwords are treated carefully and consistently. This ensures that users using multiple keyboards that enter different versions of the same character will still be able to log in. For example, some keyboards may output the full-width version of a character while other keyboards output the half-width version of the same character. The "OpaqueString" profile addresses this, among other issues, and ensures that comparison succeeds and the claimant is able to be authenticated.

When enforcing a minimum password length the authentication server SHOULD NOT count bytes as single Unicode scalar values may comprise multiple bytes. Similarly, a single emoji may be constructed from many Unicode scalar values, so it may not be appropriate to count scalar values or code points. Instead, password length SHOULD be calculated by counting Grapheme Clusters as defined in [UAX29].

7. Password Complexity Requirements

The recommendations from Section 6 of this document SHOULD be applied before any other password complexity requirements are checked. Entities SHOULD enforce a minimum length of at least 8 characters for user passwords when MFA is always in use, and a minimum length of at least 15 characters for user passwords when MFA is not in use. If using a mechanism such as PLAIN where the server performs hashing on the original password, a maximum length between 64 and 128 characters MAY be imposed to prevent denial of service (DoS) attacks. Entities SHOULD NOT apply any other password restrictions.

In addition to these password complexity requirements, servers SHOULD maintain a password blocklist and reject attempts by a claimant to use passwords on the blocklist during registration or password reset. The contents of this blocklist are a matter of server policy. Some common recommendations include lists of common passwords that are not otherwise prevented by length requirements, and passwords present in known breaches.

8. Security Considerations

This document contains recommendations that are likely to change over time. It should be reviewed regularly to ensure that it remains accurate and up to date. Many of the recommendations in this document were taken from [OWASP.CS.passwords], [DOI_10.6028_NIST.SP.800-63-4], [DOI_10.6028_NIST.SP.800-63B-4], and [NIST_SP_800_132].

The "-PLUS" variants of SCRAM [RFC5802] support channel binding to their underlying security layer, but lack a mechanism for negotiating what type of channel binding to use. In [RFC5802] the tls-unique [RFC5929] channel binding mechanism is specified as the default, and it is therefore likely to be used in most applications that support channel binding. However, in the absence of the TLS extended master secret fix [RFC7627] and the renegotiation indication TLS extension [RFC5746] the tls-unique and tls-server-endpoint channel binding data can be forged by an attacker that can MITM the connection. Before advertising a channel binding SASL mechanism, entities MUST ensure that both the TLS extended master secret fix and the renegotiation indication extension are in place and that the connection has not been renegotiated.

For TLS 1.3 [RFC8446] the commonly used tls-unique channel binding mechanism is not defined. Instead, tls-external [RFC9266] SHOULD be used as the default channel binding mechanism for TLS 1.3 and above.

Current best practices for password based authentication require using multiple authenticating factors such as a TOTP [RFC6238] or HOTP [RFC4226] token, biometric devices, or smart card posession. Work is ongoing to make multi-factor authentication possible with SASL, but until such a time authentication with only a password remains behind the state of the art.

9. IANA Considerations

This document has no actions for IANA.

10. Normative References

[IANA_sasl_mechanisms_sasl_mechanisms_1]
IANA, "SASL Mechanisms", <https://www.iana.org/assignments/sasl-mechanisms>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <https://www.rfc-editor.org/info/rfc5746>.

[RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, DOI 10.17487/RFC5929, July 2010, <https://www.rfc-editor.org/info/rfc5929>.

[RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <https://www.rfc-editor.org/info/rfc7627>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9266] Whited, S., "Channel Bindings for TLS 1.3", RFC 9266, DOI 10.17487/RFC9266, July 2022, <https://www.rfc-editor.org/info/rfc9266>.

11. Informative References

[BCRYPT] Provos, N. and D. Maziティres, "A Future-Adaptable Password Scheme", USENIX 1999 https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf, June 1999.

[DOI_10.6028_NIST.SP.800-63-4] Temoshok, D., Proud-Madruga, D., Choong, Y., Galluzzo, R., Gupta, S., LaSalle, C., Lefkovitz, N., Regenscheid, A., and National Institute of Standards and Technology (U.S.), "NIST SP 800-63-4:", DOI 10.6028/nist.sp.800-63-4, 1 August 2025, <https://doi.org/10.6028/nist.sp.800-63-4>.

[DOI_10.6028_NIST.SP.800-63B-4] Temoshok, D., Fenton, J. L, Choong, Y., Lefkovitz, N., Regenscheid, A., Galluzzo, R., Richer, J. P, and National Institute of Standards and Technology (U.S.), "NIST SP 800-63B-4:", DOI 10.6028/nist.sp.800-63b-4, 1 August 2025, <https://doi.org/10.6028/nist.sp.800-63b-4>.

[NIST_FIPS_140_3] NIST, "Security requirements for cryptographic modules", NIST Federal Information Processing Standards Publications 140-3, DOI 10.6028/NIST.FIPS.140-3, April 2019, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf>.

[NIST_SP_800_132] Turan, M S., Barker, E B., Burr, W E., Chen, L., and NIST, "Recommendation for password-based key derivation :part 1: storage applications", NIST Special Publications (General) 800-132, DOI 10.6028/NIST.SP.800-132, 2010, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf>.

[OWASP.CS.passwords] Manico, J., Saad, E., Maト☒owski, J., and R. Bailey, "Password Storage", OWASP Cheat Sheet Password Storage, April 2020, <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html>.

[RFC4226] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226, DOI 10.17487/RFC4226, December 2005, <https://www.rfc-editor.org/info/rfc4226>.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <https://www.rfc-editor.org/info/rfc4422>.

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, DOI 10.17487/RFC5802, July 2010, <https://www.rfc-editor.org/info/rfc5802>.

[RFC6238] M'Raihi, D., Machani, S., Pei, M., and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, DOI 10.17487/RFC6238, May 2011, <https://www.rfc-editor.org/info/rfc6238>.

[RFC6331] Melnikov, A., "Moving DIGEST-MD5 to Historic", RFC 6331, DOI 10.17487/RFC6331, July 2011, <https://www.rfc-editor.org/info/rfc6331>.

[RFC7677] Hansen, T., "SCRAM-SHA-256 and SCRAM-SHA-256-PLUS Simple Authentication and Security Layer (SASL) Mechanisms", RFC 7677, DOI 10.17487/RFC7677, November 2015, <https://www.rfc-editor.org/info/rfc7677>.

[RFC7914] Percival, C. and S. Josefsson, "The scrypt Password-Based Key Derivation Function", RFC 7914, DOI 10.17487/RFC7914, August 2016, <https://www.rfc-editor.org/info/rfc7914>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8264] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 8264, DOI 10.17487/RFC8264, October 2017, <https://www.rfc-editor.org/info/rfc8264>.

[RFC8265] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 8265, DOI 10.17487/RFC8265, October 2017, <https://www.rfc-editor.org/info/rfc8265>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9106] Biryukov, A., Dinu, D., Khovratovich, D., and S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications", RFC 9106, DOI 10.17487/RFC9106, September 2021, <https://www.rfc-editor.org/info/rfc9106>.

[UAX29] Davis, M. and C. Chapman, "Unicode Text Segmentation", February 2020, <https://www.unicode.org/reports/tr29/>.

Appendix A. Acknowledgments

The author would like to thank the civil servants at the National Institute of Standards and Technology for their work on the Special Publications series. U.S. executive agencies are an undervalued national treasure, and they deserve our thanks.

Thanks also to Cameron Paul, Thomas Copeland, Robbie Harwood, Jim Fenton, Alexey Melnikov, and Simon Josefsson for their reviews and suggestions.

Author’s Address

Sam Whited
Atlanta, GA
United States of America
Email: sam@samwhited.com
URI: https://blog.samwhited.com/