Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 29 August 2025

W. Guo
L. Xia
J. Li
Huawei Technologies
25 February 2025

PAKE Usage in TLS 1.3
draft-guo-pake-in-tls-01

Abstract

This document provides a mechanism that uses password-authenticated key exchange (PAKE) as an authentication and key establishment in TLS 1.3, and that supports PAKE algorithms negotiation.

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 29 August 2025.

Table of Contents

1. Introduction

In some applications, it is desirable to use a pre-shared low-entropy secret, such as a password, to authenticate with each other between a client and a server. Although TLS 1.3 [RFC8446] itself provides an authentication mechanism of pre-shared keys (PSKs), PSKs need to be full-entropy secrets. If a low-entropy password is directly used in this authentication mechanism, then the PSK binder values can be used by a passive adversary to mount an offline dictionary attack on the password, and even if without the PSK binder and the PSK mechanism is combined with Diffie-Hellman (DH) key establishment, an active adversary can still offline enumerate the password through a man-in-the-middle (MITM) attack.

Note that there are already some early works about PAKE usage in TLS: the Secure Remote Password (SRP) PAKE protocol has been integrated in prior versions of TLS [RFC5054], but it does not extend to later TLS 1.3; the Dragonfly PAKE protocol has also been integrated in both prior versions of TLS and the latest TLS 1.3 [RFC8492]; the integration of the OPAQUE [I-D.draft-irtf-cfrg-opaque-18] PAKE protocol with TLS 1.3 has been presented in [I-D.draft-sullivan-tls-opaque-01]. However, they all do not consider PAKE algorithms negotiation, which is very important in practice, because some algorithms may be broken in the future and we need some way to negotiate new algorithms. Although PAKE algorithms negotiation has been mentioned in [I-D.draft-barnes-tls-pake-04], it only presents the integration of the SPAKE2+ [RFC9383] PAKE protocol

with TLS 1.3 assuming that the PAKE algorithm has been negotiated in advance. This document provides a mechanism that uses PAKE as an authentication and key establishment in TLS 1.3, which supports PAKE algorithms negotiation.

2. Terminology

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.

This document uses terminology such as client, server, connection, handshake, endpoint, and peer that are defined in Section 1.1 of [RFC8446].

3. PAKE Usage in TLS 1.3

This section describes details about PAKE usage in TLS 1.3.

3.1. TLS Extensions

To use PAKE as an authentication and key establishment mechanism in TLS 1.3, we define three TLS extensions: "pake_algorithms", "pake_share" and "pake_auth_modes".

* The "pake_algorithms" extension indicates all the PAKE algorithm suites supported by the client, it can only appear in ClientHello and EncryptedExtension messages, otherwise peers MUST abort the handshake with an "illegal_parameter" alert.

* The "pake_share" extension indicates the specific parameters of PAKE key exchange between both parties, it can only appear in ClientHello, ServerHello, and HelloRetryRequest messages, otherwise peers MUST abort the handshake with an "illegal_parameter" alert.

* The "pake_auth_modes" extension indicates the mode of PAKE authentication for the server, it can only appear in ClientHello and ServerHello messages, otherwise peers MUST abort the handshake with an "illegal_parameter" alert.

3.1.1. PAKE Algorithms

When sent by the client, the "pake_algorithms" extension indicates PAKE algorithms which the client supports for authenticated key exchange, ordered from most preferred to least preferred.

The "extension_data" field of this extension contains a "PAKEAlgorithmList" value:

```
enum {
  /* Balanced PAKE algorithms */
  0x0000~0x3FFF

  /* Augmented PAKE algorithms */
  0x4000~0x7FFF

  /* Post-Quantum Balanced PAKE algorithms */
  0x8000~0xBFFF

  /* Post-Quantum Augmented PAKE algorithms */
  0xC000~0xFFFF
  (0xFFFF)
} PAKEAlgorithm;

struct {
    PAKEAlgorithm supported_pake_algorithms<2..2^16-1>;
} PAKEAlgorithmList;
```

Here, each PAKE algorithm suite consists of two part, the first part is represented by the first byte and specifies different PAKE algorithms, and the second part is represented by the second byte and specifies different ciphersuites for some PAKE algorithm. For example, the first bytes "0x00" and "0x01" represent CPace [I-D.draft-irtf-cfrg-cpace-13] and SPAKE2 [RFC9382], respectively; and the first bytes "0x40" and "0x41" represent OPAQUE [I-D.draft-irtf-cfrg-opaque-18] and SPAKE2+ [RFC9383], respectively. For the SPAKE2+ algorithm with the first byte "0x41", the second bytes "0x00~0xFF" can be used to represent its different ciphersuites (see Section 4 of [RFC9383]).

As of TLS 1.3, servers are permitted to send the "pake_algorithms" extension to the client. Clients MUST NOT act upon any information found in "pake_algorithms" prior to successful completion of the handshake but MAY use the information learned from a successfully completed handshake to change what PAKE algorithms they use in their "pake_share" extension in subsequent connections. If the server has a PAKE algorithm it prefers to the ones in the "pake_share" extension but is still willing to accept the ClientHello, it SHOULD send "pake_algorithms" to update the client's view of its preferences; this extension SHOULD contain all PAKE algorithms the server supports, regardless of whether they are currently supported by the client.

3.1.2. PAKE Share

The "pake_share" extension contains the endpoint's PAKE parameters. The "PAKEShareEntry" value is defined as follows:

```
struct {
    PAKEAlgorithm pake_algorithm;
    opaque pake_message<1..2^16-1>;
} PAKEShareEntry;
```

* The "pake_algorithm" field indicates the PAKE algorithm used.

* The "pake_message" field indicates the PAKE key exchange information, its contents are determined by the specified PAKE algorithm, as shown below.

```
struct {
    select (pake_algorithm) {
    case CPace:
        opaque dh_share;
        opaque associated_data;
    case SPAKE2:
        opaque dh_share;
    case OPAQUE:
        opaque credential_message;
        opaque nonce;
        opaque dh_share;
    case SPAKE2+:
        opaque dh_share;
    };
} PAKEMessage;
```

* The "dh_share" field indicates a (EC)DH public element that is determined by the specified PAKE algorithm and the underlying group. 1) For CPace, the "dh_share" field indicates the value Ya or Yb in Section 6.2 of [I-D.draft-irtf-cfrg-cpace-13]; 2) for SPAKE2, this field indicates the value pA or pB in Section 3.3 of [RFC9382]; 3) for OPAQUE, this field indicates the value client_public_keyshare or server_public_keyshare in Section 6.1 of [I-D.draft-irtf-cfrg-opaque-18]; 4) for SPAKE2+, this field indicates the value shareP or shareV in Section 3.3 of [RFC9383].

* The "associated_data" field for CPace indicates the value ADa or ADb in Section 3.1 of [I-D.draft-irtf-cfrg-cpace-13].

* The "credential_message" field for OPAQUE indicates the value CredentialRequest or CredentialResponse in Section 6.3.1 of [I-D.draft-irtf-cfrg-opaque-18].

* The "nonce" field for OPAQUE indicates the value client_nonce or server_nonce in Section 6.1 of [I-D.draft-irtf-cfrg-opaque-18].

Note that the "random" value of the ClientHello message (see Section 4.1.2 of [RFC8446]) can be used as a unique session identifier sid of the CPace algorithm (see Section 3.1 of [I-D.draft-irtf-cfrg-cpace-13]).

(1) In the ClientHello message, the "extension_data" field of this extension contains a "PAKEShareClientHello" value:

```
struct {
    opaque client_identity<0..2^16-1>;
    PAKEShareEntry client_shares<0..2^16-1>;
} PAKEShareClientHello;
```

* The "client_identity" field indicates a client's identity used in the PAKE algorithm.

* The "client_shares" field contains a list of offered PAKEShareEntry values in descending order of client preference.

The contents of "client_shares" field MAY be empty if the client is requesting a HelloRetryRequest. Each PAKEShareEntry value MUST correspond to a PAKE algorithm offered in the "pake_algorithms" extension and MUST appear in the same order. However, the values MAY be a non-contiguous subset of the "pake_algorithms" extension and MAY omit the most preferred PAKE algorithms. Such a situation could arise if the most preferred PAKE algorithms are new and unlikely to be supported in enough places to make pregenerating PAKE shares for them efficient.

Clients can offer as many PAKEShareEntry values as the number of supported PAKE algorithms it is offering, each representing a single set of PAKE key exchange parameters. For instance, a client might offer shares for several PAKE algorithms. The "pake_message" values for each PAKEShareEntry MUST be generated independently. Clients MUST NOT offer multiple PAKEShareEntry values for the same PAKE algorithm. Clients MUST NOT offer any PAKEShareEntry values for PAKE algorithms not listed in the client' s "pake_algorithms" extension. Servers MAY check for violations of these rules and abort the handshake with an "illegal_parameter" alert if one is violated. In addition, servers MUST verify that (1) the list of client shares to see if there is one with the provided "client_identity" value, and (2) if the "pake_message" contents in the "client_shares" field are valid (e.g., valid group elements). If either of these checks fails, then servers MUST abort the handshake with an "illegal_parameter" alert.

(2) In a HelloRetryRequest message, the "extension_data" field of this extension contains a "PAKEShareHelloRetryRequest" value:

```
struct {
    PAKEAlgorithm selected_pake_algorithm;
} PAKEShareHelloRetryRequest;
```

If a client receives a second HelloRetryRequest in the same connection (i.e., where the ClientHello was itself in response to a HelloRetryRequest), it MUST abort the handshake with an "unexpected_message" alert. Upon receipt of this extension in a HelloRetryRequest, the client MUST verify that (1) the "selected_pake_algorithm" field corresponds to a PAKE algorithm which was provided in the "pake_algorithms" extension in the original ClientHello, and (2) the "selected_pake_algorithm" field does not correspond to a PAKE algorithm which was provided in the "pake_share" extension in the original ClientHello. If either of these checks fails, then the client MUST abort the handshake with an "illegal_parameter" alert. Otherwise, when sending the new ClientHello, the client MUST replace the original "pake_share" extension with one containing only a new PAKEShareEntry for the PAKE algorithm indicated in the "selected_pake_algorithm" field of the triggering HelloRetryRequest.

(3) In a ServerHello message, the "extension_data" field of this extension contains a PAKEShareServerHello value:

```
struct {
    opaque server_identity<0..2^16-1>;
    PAKEShareEntry server_share;
} PAKEShareServerHello;
```

* The "server_identity" field indicates a server’s identity used in the PAKE algorithm.

* The "server_share" field contains a single PAKEShareEntry value that is in the same PAKE algorithm as one of the client’ s shares.

Upon receipt of the ClientHello message, servers use the "client_identity" value to determine if there is a PAKE registration record in their database, if not, servers MAY reply with a fake PAKE response to prevent active client enumeration attacks (see Section 3.1.4); if so, servers reply with a normal PAKE response containing a "pake_share" extension in the ServerHello message as described below.

If using PAKE key establishment, servers offer exactly one PAKEShareEntry in the ServerHello. This value MUST be in the same PAKE algorithm as the PAKEShareEntry value offered by the client that the server has selected for the negotiated PAKE key exchange. Servers MUST NOT send a PAKEShareEntry for any PAKE algorithm not indicated in the client's "pake_algorithms" extension. If using PAKE key establishment and a HelloRetryRequest containing a "pake_share" extension was received by the client, the client MUST verify that the selected PAKEAlgorithm in the ServerHello is the same as that in the HelloRetryRequest. If this check fails, the client MUST abort the handshake with an "illegal_parameter" alert. In addition, the client MUST verify that (1) the "server_identity" value to see if it is the target server's name, and (2) if the "pake_message" content in the "server_share" field is valid (e.g., a valid group element). If either of these checks fails, then the client MUST abort the handshake with an "illegal_parameter" alert.

3.1.3. PAKE Authentication Modes

In order to use PAKE, clients MUST also send a "pake_auth_modes" extension. The semantics of this extension are that the client wants to identify which authentication modes will be used, and its content MAY be multiple authentication modes. The "extension_data" field of this extension contains a "PAKEAuthModes" value:

```
enum {
    pake_auth (0),
    pake_certificate_auth (1),
    (255)
} PAKEAuthMode;

struct {
    PAKEAuthMode auth_modes<1..255>;
} PAKEAuthModes;
```

* The "pake_auth" mode indicates PAKE-only authentication. In this mode, the server MUST NOT send the Certificate, CertificateVerify and CertificateRequest messages.

* The "pake_certificate_auth" mode indicates PAKE authentication combined with certificate authentication. In this mode, the server MUST send the Certificate and CertificateVerify messages to conduct certificate-based server authentication, and optionally sends the CertificateRequest message if certificate-based client authentication is desired.

A client MUST provide a "pake_auth_modes" extension if it offers "pake_algorithms" and "pake_share" extensions. If clients offer "pake_algorithms" and "pake_share" without a "pake_auth_modes" extension, servers MUST abort the handshake with an "illegal_parameter" alert. Servers MUST provide only one authentication mode in the "pake_auth_modes" extension to unambiguously identify which mode was selected, but MUST NOT select an authentication mode that is not listed by the client. If servers offer "pake_share" without a "pake_auth_modes" extension, clients MUST abort the handshake with an "illegal_parameter" alert.

3.1.4. Enumeration Prevention

If there is not a PAKE registration record on the server and the server directly aborts the handshake with an "illegal_parameter" alert, then this will provide an attacker with an oracle for client identities, that allows the attacker to learn whether there exists a PAKE registration record for the given "client_identity" value. To prevent this active client enumeration attack, the server can generate a fake PAKE response as follows.

* Select the first pake_algorithm that is supported by both the client and the server.

* Offer a "pake_share" extension in the ServerHello message, which includes the "server_identity", the selected "pake_algorithm" and the associated "pake_message" values. Here, the "dh_share" field of the "pake_message" can be generated by picking a value at random from the set of all valid DH shares of the selected pake_algorithm; for OPAQUE, the "credential_message" field of the "pake_message" can be generated according to Section 6.3.2.2 of [I-D.draft-irtf-cfrg-opaque-18] to contain a fake CredentialResponse.

* Complete the remainder of the protocol as usual.

As a result, the normal and fake PAKE responses are indistinguishable from one another, and the client cannot figure out whether the given "client_identity" value has a PAKE registration record on the server upon receiving the ServerHello and server’s Finished messages. The verification of the server’s Finished message will fail with overwhelming probability, since the server’s DH share is a randomly picked value. However, the unsuccessful verification make the attacker only know whether a given (client_identity, password) pair is correct, and it cannot be used by the attacker to learn whether the client_identity is an unregistered client identity.

3.2. Key Schedule

Based on the PAKE parameters exchanged in the ClientHello and ServerHello, the client and server execute the negotiated PAKE protocol to derive a PAKE shared secret, which is denoted by pake_shared_secret.

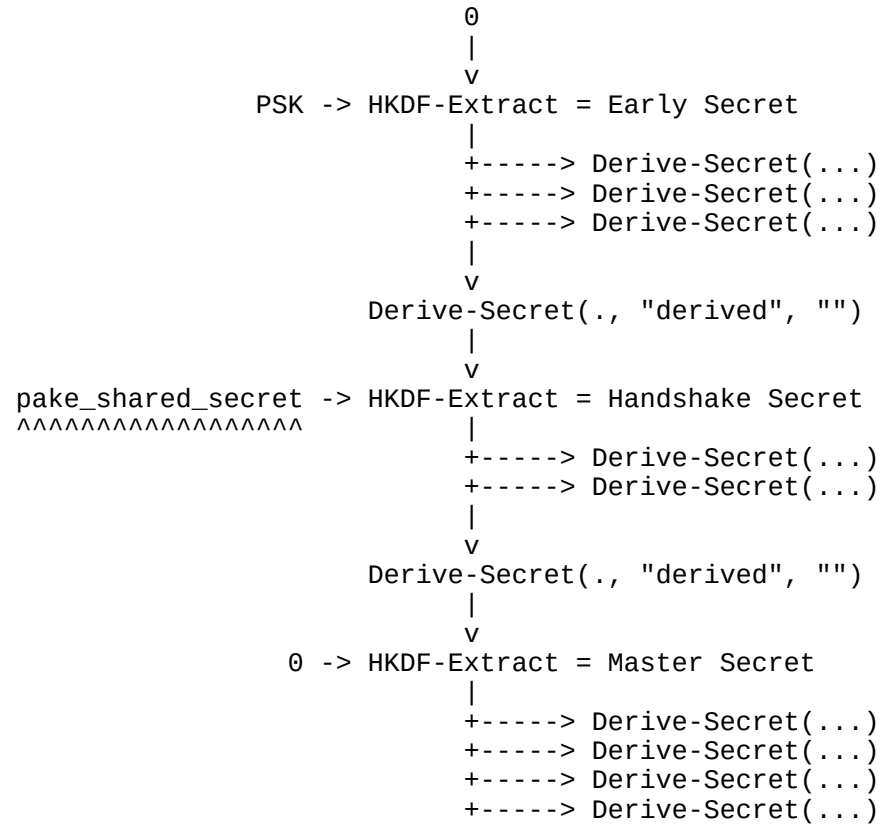
* For CPace, this secret indicates the value ISK in Section 6.2 of [I-D.draft-irtf-cfrg-cpace-13].

* For SPAKE2, this secret indicates the concatenated value Ke || Ka in Section 3.3 of [RFC9382].

* For OPAQUE, this secret indicates the concatenated value Km2 || Km3 || session_key in Section 6.4.3 and Section 6.4.4 of [I-D.draft-irtf-cfrg-opaque-18].

* For SPAKE2+, this secret indicates the value K_main in Section 3.3 of [RFC9383].

As shown below, the "pake_shared_secret" value is used as the "(EC)DHE" input to the TLS 1.3 key schedule, and the remaining process of the key schedule is unchanged.



3.3. Authentication Messages

In this mechanism, PAKEs are used for both password authentication and key establishment, the derived PAKE shared secret is fed into the TLS 1.3 key schedule, and PAKE-based authentication is finally confirmed by the validation of the Finished message.

Note that PAKE-based authentication is compatible with server-side certificate authentication. If the client specifies a "pake_auth" mode in the "pake_auth_modes" extension, then the server does not need to send Certificate and CertificateVerify messages, and only needs to send a Finished message to perform server-side PAKE authentication. If the client specifies a "pake_certificate_auth" mode in the "pake_auth_modes" extension, or specifies "pake_auth" and "pake_certificate_auth" modes and the server selects the latter one, then the server needs to send Certificate and CertificateVerify messages to conduct server-side certificate authentication, and send a Finished message to perform server-side PAKE authentication.

PAKE-based authentication is also compatible with client-side certificate authentication, because the server can optionally send a CertificateRequest message to the client. If the client receives a CertificateRequest message, it needs to send Certificate and CertificateVerify messages to conduct client-side certificate authentication, and send a Finished message to perform client-side PAKE authentication; otherwise only the Finished message needs to be sent.

4. Security Considerations

4.1. Client Enumeration

Client enumeration denotes attacks where the attacker attempts to learn whether a given client identity has a PAKE registration record on a server. These attacks are migrated by requiring servers to process non-existent client identities in a way that is indistinguishable from their behavior with registered clients.

4.2. Key Confirmation

Because the key exchange transcript specified in TLS 1.3 is a superset of the transcript as defined in PAKEs, the explicit key confirmation of PAKEs can be achieved by the TLS 1.3 Finished messages without generating and verifying the PAKEs' own key confirmation MACs additionally.

5. IANA Considerations

This document defines three new TLS extension types "pake_algorithms", "pake_share" and "pake_auths" with the following contents (see Table 1), and requests that IANA add the three values to the "TLS ExtensionType Values" Registry defined in [RFC8446] and [RFC8447].

Value	Extension Name	TLS 1.3	Reference
TBD1	pake_algorithms	CH, EE	PRF XXXX
TBD2	pake_share	CH, SH, HRR	PRF XXXX
TBD3	pake_auth_modes	CH, SH	PRF XXXX

Table 1: New TLS Extension Types

6. References

6.1. Normative References

[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/rfc/rfc8446>.

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, DOI 10.17487/RFC5054, November 2007, <https://www.rfc-editor.org/rfc/rfc5054>.

[RFC8492] Harkins, D., Ed., "Secure Password Ciphersuites for Transport Layer Security (TLS)", RFC 8492, DOI 10.17487/RFC8492, February 2019, <https://www.rfc-editor.org/rfc/rfc8492>.

[RFC9383] Taubert, T. and C. A. Wood, "SPAKE2+, an Augmented Password-Authenticated Key Exchange (PAKE) Protocol", RFC 9383, DOI 10.17487/RFC9383, September 2023, <https://www.rfc-editor.org/rfc/rfc9383>.

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

[RFC9382] Ladd, W., "SPAKE2, a Password-Authenticated Key Exchange", RFC 9382, DOI 10.17487/RFC9382, September 2023, <https://www.rfc-editor.org/rfc/rfc9382>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/rfc/rfc8447>.

6.2. Informative References

[I-D.draft-irtf-cfrg-opaque-18] Bourdrez, D., Krawczyk, H., Lewi, K., and C. A. Wood, "The OPAQUE Augmented PAKE Protocol", Work in Progress, Internet-Draft, draft-irtf-cfrg-opaque-18, 21 November 2024, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-opaque-18>.

[I-D.draft-sullivan-tls-opaque-01]
Sullivan, N., Krawczyk, H., Friel, O., and R. Barnes, "OPAQUE with TLS 1.3", Work in Progress, Internet-Draft, draft-sullivan-tls-opaque-01, 22 February 2021, <https://datatracker.ietf.org/doc/html/draft-sullivan-tls-opaque-01>.

[I-D.draft-barnes-tls-pake-04]
Barnes, R. and O. Friel, "Usage of PAKE with TLS 1.3", Work in Progress, Internet-Draft, draft-barnes-tls-pake-04, 16 July 2018, <https://datatracker.ietf.org/doc/html/draft-barnes-tls-pake-04>.

[I-D.draft-irtf-cfrg-cpace-13]
Abdalla, M., Haase, B., and J. Hesse, "CPace, a balanced composable PAKE", Work in Progress, Internet-Draft, draft-irtf-cfrg-cpace-13, 14 October 2024, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-cpace-13>.

Authors' Addresses

Wei Guo
Huawei Technologies
Email: guowei90@huawei.com

Liang Xia
Huawei Technologies
Email: frank.xialiang@huawei.com

Ji Li
Huawei Technologies
Email: liji100@huawei.com